pTHINC: A Thin-Client Architecture for Mobile Wireless Web

Joeng Kim, Ricardo A. Baratto, and Jason Nieh
Department of Computer Science
Columbia University, New York, NY, USA
{jk2438, ricardo, nieh}@cs.columbia.edu

ABSTRACT

Although web applications are gaining popularity on mobile wireless PDAs, web browsers on these systems can be quite slow and often lack adequate functionality to access many web sites. We have developed pTHINC, a PDA thin-client solution that leverages more powerful servers to run full-function web browsers and other application logic, then sends simple screen updates to the PDA for display. pTHINC uses server-side screen scaling to provide high-fidelity display and seamless mobility across a broad range of different clients and screen sizes, including both portrait and landscape viewing modes. pTHINC also leverages existing PDA control buttons to improve system usability and maximize available screen resolution for application display. We have implemented pTHINC on Windows Mobile and evaluated its performance on mobile wireless devices. Our results compared to local PDA web browsers and other thin-client approaches demonstrate that pTHINC provides superior web browsing performance and is the only PDA thin client that effectively supports crucial browser helper applications such as video playback.

Categories and Subject Descriptors: C.2.4 Computer-Communication-Networks: Distributed Systems – client/server

General Terms: Design, Experimentation, Performance

Keywords: thin-client computing, remote display, mobility, pervasive web

1. INTRODUCTION

The increasing ubiquity of wireless networks and decreasing cost of hardware is fueling a proliferation of mobile wireless handheld devices, both as standalone wireless Personal Digital Assistants (PDA) and popular integrated PDA/cell phone devices. These devices are enabling new forms of mobile computing and communication. Service providers are leveraging these devices to deliver pervasive web access, and mobile web users already often use these devices to access web-enabled information such as news, email, and localized travel guides and maps. It is likely that within a few years, most of the devices accessing the web will be mobile.

Users typically access web content by running a web browser and associated helper applications locally on the PDA. Although native web browsers exist for PDAs, they deliver subpar performance and have a much smaller feature set and more limited functionality than their desktop computing counterparts [10]. As a result, PDA web browsers are often not able to display web content from web sites that leverage more advanced web technologies to deliver a richer web experience. This fundamental problem arises for two reasons. First, because PDAs have a completely different hardware/software environment from traditional desktop computers, web applications need to be rewritten and customized for PDAs if at all possible, duplicating development costs. Because the desktop application market is larger and more mature, most development effort generally ends up being spent on desktop applications, resulting in greater functionality and performance than their PDA counterparts. Second, PDAs have a more resource constrained environment than traditional desktop computers to provide a smaller form factor and longer battery life. Desktop web browsers are large, complex applications that are unable to run on a PDA. Instead, developers are forced to significantly strip down these web browsers to provide a usable PDA web browser, thereby crippling PDA browser functionality.

Thin-client computing provides an alternative approach for enabling pervasive web access from handheld devices. A thin-client computing system consists of a server and a client that communicate over a network using a remote display protocol. The protocol enables 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 applications from the server to the client. Using a thin-client model for mobile handheld devices, PDAs can become simple stateless clients that leverage the remote server capabilities to execute web browsers and other helper applications.

The thin-client model provides several important benefits for mobile wireless web. First, standard desktop web applications can be used to deliver web content to PDAs without rewriting or adapting applications to execute on a PDA, reducing development costs and leveraging existing software investments. Second, complex web applications can be executed on powerful servers instead of running stripped down versions on more resource constrained PDAs, providing greater functionality and better performance [10]. Third, web applications can take advantage of servers with faster networks and better connectivity, further boosting application performance. Fourth, PDAs can be even simpler devices since they do not need to perform complex application logic, potentially reducing energy consumption and extend-

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

ing battery life. Finally, PDA thin clients can be essentially stateless appliances that do not need to be backed up or restored, require almost no maintenance or upgrades, and do not store any sensitive data that can be lost or stolen. This model provides a viable avenue for medical organizations to comply with HIPAA regulations [6] while embracing mobile handhelds in their day to day operations.

Despite these potential advantages, thin clients have been unable to provide the full range of these benefits in delivering web applications to mobile handheld devices. Existing thin clients were not designed for PDAs and do not account for important usability issues in the context of small form factor devices, resulting in difficulty in navigating displayed web content. Furthermore, existing thin clients are ineffective at providing seamless mobility across the heterogeneous mix of device display sizes and resolutions. While existing thin clients can already provide faster performance than native PDA web browsers in delivering HTML web content, they do not effectively support more display-intensive web helper applications such as multimedia video, which is increasingly an integral part of available web content.

To harness the full potential of thin-client computing in providing mobile wireless web on PDAs, we have developed pTHINC (PDA THin-client InterNet Computing). pTHINC builds on our previous work on THINC [1] to provide a thin-client architecture for mobile handheld devices. pTHINC virtualizes and resizes the display on the server to efficiently deliver high-fidelity screen updates to a broad range of different clients, screen sizes, and screen orientations, including both portrait and landscape viewing modes. This enables pTHINC to provide the same persistent web session across different client devices. For example, pTHINC can provide the same web browsing session appropriately scaled for display on a desktop computer and a PDA so that the same cookies, bookmarks, and other meta-data are continuously available on both machines simultaneously. pTHINC's virtual display approach leverages semantic information available in display commands, and client-side video hardware to provide more efficient remote display mechanisms that are crucial for supporting more display-intensive web applications. Given limited display resolution on PDAs, pTHINC maximizes the use of screen real estate for remote display by moving control functionality from the screen to readily available PDA control buttons, improving system usability.

We have implemented pTHINC on Windows Mobile and demonstrated that it works transparently with existing applications, window systems, and operating systems, and does not require modifying, recompiling, or relinking existing software. We have quantitatively evaluated pTHINC against local PDA web browsers and other thin-client approaches on Pocket PC devices. Our experimental results demonstrate that pTHINC provides superior web browsing performance and is the only PDA thin client that effectively supports crucial browser helper applications such as video playback.

This paper presents the design and implementation of pTHINC. Section 2 describes the overall usage model and usability characteristics of pTHINC. Section 3 presents the design and system architecture of pTHINC. Section 4 presents experimental results measuring the performance of pTHINC on web applications and comparing it against native PDA browsers and other popular PDA thin-client systems. Section 5 discusses related work. Finally, we present some concluding remarks.

2. PTHINC USAGE MODEL

pTHINC is a thin-client system that consists of a simple client viewer application that runs on the PDA and a server that runs on a commodity PC. The server leverages more powerful PCs to to run web browsers and other application logic. The client takes user input from the PDA stylus and virtual keyboard and sends them to the server to pass to the applications. Screen updates are then sent back from the server to the client for display to the user.

When the pTHINC PDA client is started, the user is presented with a simple graphical interface where information such as server address and port, user authentication information, and session settings can be provided. pTHINC first attempts to connect to the server and perform the necessary handshaking. Once this process has been completed, pTHINC presents the user with the most recent display of his session. If the session does not exist, a new session is created. Existing sessions can be seamlessly continued without changes in the session setting or server configuration.

Unlike other thin-client systems, pTHINC provides a user with a persistent web session model in which a user can launch a session running a web browser and associated applications at the server, then disconnect from that session and reconnect to it again anytime. When a user reconnects to the session, all of the applications continue running where the user left off, so that the user can continue working as though he or she never disconnected. The ability to disconnect and reconnect to a session at anytime is an important benefit for mobile wireless PDA users which may have intermittent network connectivity.

pTHINC's persistent web session model enables a user to reconnect to a web session from devices other than the one on which the web session was originally initiated. This provides users with seamless mobility across different devices. If a user loses his PDA, he can easily use another PDA to access his web session. Furthermore, pTHINC allows users to use non-PDA devices to access web sessions as well. A user can access the same persistent web session on a desktop PC as on a PDA, enabling a user to use the same web session from any computer.

pTHINC's persistent web session model addresses a key problem encountered by mobile web users, the lack of a common web environment across computers. Web browsers often store important information such as bookmarks, cookies, and history, which enable them to function in a much more useful manner. The problem that occurs when a user moves between computers is that this data, which is specific to a web browser installation, cannot move with the user. Furthermore, web browsers often need helper applications to process different media content, and those applications may not be consistently available across all computers. pTHINC addresses this problem by enabling a user to remotely use the exact same web browser environment and helper applications from any computer. As a result, pTHINC can provide a common, consistent web browsing environment for mobile users across different devices without requiring them to attempt to repeatedly synchronize different web browsing environments across multiple machines.

To enable a user to access the same web session on different devices, pTHINC must provide mechanisms to support different display sizes and resolutions. Toward this end, pTHINC provides a zoom feature that enables a user to zoom in and out of a display and allows the display of a web

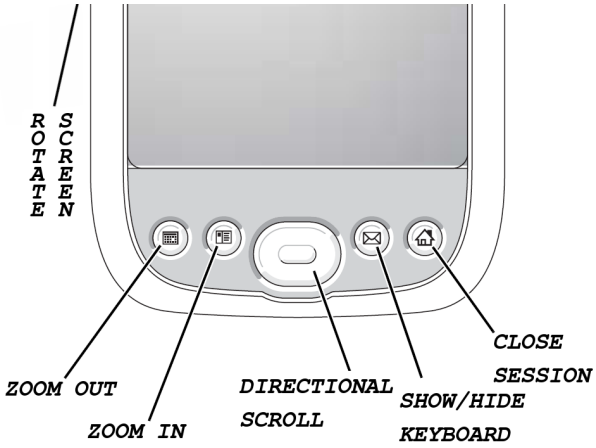

Figure 1: pTHINC shortcut keys

session to be resized to fit the screen of the device being used. For example, if the server is running a web session at 1024×768 but the client is a PDA with a display resolution of 640×480, pTHINC will resize the desktop display to fit the full display in the smaller screen of the PDA. pTHINC provides the PDA user with the option to increase the size of the display by zooming in to different parts of the display. Users are often familiar with the general layout of commonly visited websites, and are able to leverage this resizing feature to better navigate through web pages. For example, a user can zoom out of the display to view the entire page content and navigate hyperlinks, then zoom in to a region of interest for a better view.

To enable a user to access the same web session on different devices, pTHINC must also provide mechanisms to support different display orientations. In a desktop environment, users are typically accustomed to having displays presented in landscape mode where the screen width is larger than its height. However, in a PDA environment, the choice is not always obvious. Some users may prefer having the display in portrait mode, as it is easier to hold the device in their hands, while others may prefer landscape mode in order to minimize the amount of side-scrolling necessary to view a web page. To accommodate PDA user preferences, pTHINC provides an orientation feature that enables it to seamless rotate the display between landscape and portrait mode. The landscape mode is particularly useful for pTHINC users who frequently access their web sessions on both desktop and PDA devices, providing those users with the same familiar landscape setting across different devices.

Because screen space is a relatively scarce resource on PDAs, pTHINC runs in fullscreen mode to maximize the screen area available to display the web session. To be able to use all of the screen on the PDA and still allow the user to control and interact with it, pTHINC reuses the typical shortcut buttons found on PDAs to perform all the control functions available to the user. The buttons used by pTHINC do not require any OS environment changes; they are simply intercepted by the pTHINC client application when they are pressed. Figure 1 shows how pTHINC utilizes the shortcut buttons to provide easy navigation and improve the overall user experience. These buttons are not device specific, and the layout shown is common to widely-used PocketPC devices. pTHINC provides six shortcuts to support its usage model:

- *Rotate Screen*: The record button on the left edge is used to rotate the screen between portrait and landscape mode each time the button is pressed.
- *Zoom Out*: The leftmost button on the bottom front is used to zoom out the display of the web session providing a bird's eye view of the web session.
- *Zoom In*: The second leftmost button on the bottom front is used to zoom in the display of the web session to more clearly view content of interest.
- *Directional Scroll*: The middle button on the bottom front is used to scroll around the display using a single control button in a way that is already familiar to PDA users. This feature is particularly useful when the user has zoomed in to a region of the display such that only part of the display is visible on the screen.
- *Show/Hide Keyboard*: The second rightmost button on the bottom front is used to bring up a virtual keyboard drawn on the screen for devices which have no physical keyboard. The virtual keyboard uses standard PDA OS mechanisms, providing portability across different PDA environments.
- *Close Session*: The rightmost button on the bottom front is used to disconnect from the pTHINC session.

pTHINC uses the PDA touch screen, stylus, and standard user interface mechanisms to provide a user interface point-and-click metaphor similar to that provided by the mouse in a traditional desktop computing environment. pTHINC does not use a cursor since PDA environments do not provide one. Instead, a user can use the stylus to tap on different sections of the touch screen to indicate input focus. A single tap on the touch screen generates a corresponding single click mouse event. A double tap on the touch screen generates a corresponding double click mouse event. pTHINC provides two-button mouse emulation by using the stylus to press down on the screen for one second to generate a right mouse click. All of these actions are identical to the way users already interact with PDA applications in the common PocketPC environment. In web browsing, users can click on hyperlinks and focus on input boxes by simply tapping on the desired screen area of interest. Unlike local PDA web browsers and other PDA applications, pTHINC leverages more powerful desktop user interface metaphors to enable users to manipulate multiple open application windows instead of being limited to a single application window at any given moment. This provides increased browsing flexibility beyond what is currently available on PDA devices. Similar to a desktop environment, browser windows and other application windows can be moved around by pressing down and dragging the stylus similar to a mouse.

3. PTHINC SYSTEM ARCHITECTURE

pTHINC builds on the THINC [1] remote display architecture to provide a thin-client system for PDAs. pTHINC virtualizes the display at the server by leveraging the video device abstraction layer, which sits below the window server and above the framebuffer. This is a well-defined, low-level, device-dependent layer that exposes the video hardware to the display system. pTHINC accomplishes this through a simple virtual display driver that intercepts drawing commands, packetizes, and sends them over the network.

While other thin-client approaches intercept display commands at other layers of the display subsystem, pTHINC's display virtualization approach provides some key benefits in efficiently supporting PDA clients. For example, intercepting display commands at a higher layer between applications and the window system as is done by X [17] requires replicating and running a great deal of functionality on the PDA that is traditionally provided by the desktop window system. Given both the size and complexity of traditional window systems, attempting to replicate this functionality in the restricted PDA environment would have proven to be a daunting, and perhaps unfeasible task. Furthermore, applications and the window system often require tight synchronization in their operation and imposing a wireless network between them by running the applications on the server and the window system on the client would significantly degrade performance. On the other hand, intercepting at a lower layer by extracting pixels out of the framebuffer as they are rendered provides a simple solution that requires very little functionality on the PDA client, but can also result in degraded performance. The reason is that by the time the remote display server attempts to send screen updates, it has lost all semantic information that may have helped it encode efficiently, and it must resort to using a generic and expensive encoding mechanism on the server, as well as a potentially expensive decoding mechanism on the limited PDA client. In contrast to both the high and low level interception approaches, pTHINC's approach of intercepting at the device driver provides an effective balance between client and server simplicity, and the ability to efficiently encode and decode screen updates.

By using a low-level virtual display approach, pTHINC can efficiently encode application display commands using only a small set of low-level commands. In a PDA environment, this set of commands provides a crucial component in maintaining the simplicity of the client in the resource-constrained PDA environment. The display commands are shown in Table 1, and work as follows. COPY instructs the client to copy a region of the screen from its local framebuffer to another location. This command improves the user experience by accelerating scrolling and opaque window movement without having to resend screen data from the server. SFILL, PFILL, and BITMAP are commands that paint a fixed-size region on the screen. They are useful for accelerating the display of solid window backgrounds, desktop patterns, backgrounds of web pages, text drawing, and certain operations in graphics manipulation programs. SFILL fills a sizable region on the screen with a single color. PFILL replicates a tile over a screen region. BITMAP performs a fill using a bitmap of ones and zeros as a stipple to apply a foreground and background color. Finally, RAW is used to transmit unencoded pixel data to be displayed verbatim on a region of the screen. This command is invoked as a last resort if the server is unable to employ any other command, and it is the only command that may be compressed to mitigate its impact on network bandwidth.

pTHINC delivers its commands using a non-blocking, server-push update mechanism, where as soon as display updates are generated on the server, they are sent to the client. Clients are not required to explicitly request display updates, thus minimizing the impact that the typical varying network latency of wireless links may have on the responsiveness of the system. Keeping in mind that resource constrained PDAs and wireless networks may not be able to keep up with a fast server generating a large number of updates, pTHINC is able to coalesce, clip, and discard updates automatically if network loss or congestion occurs, or the client cannot keep up with the rate of updates. This type of behavior proves crucial in a web browsing environment, where for example, a page may be redrawn multiple times as it is rendered on the fly by the browser. In this case, the PDA will only receive and render the final result, which clearly is all the user is interesting in seeing.

Command	Description
COPY	Copy a frame buffer area to specified coordinates
SFILL	Fill an area with a given pixel color value
PFILL	Tile an area with a given pixel pattern
BITMAP	Fill a region using a bit pattern
RAW	Display raw pixel data at a given location

Table 1: pTHINC Protocol Display Commands

pTHINC prioritizes the delivery of updates to the PDA using a Shortest-Remaining-Size-First (SRSF) preemptive update scheduler. SRSF is analogous to Shortest-Remaining-Processing-Time scheduling, which is known to be optimal for minimizing mean response time in an interactive system. In a web browsing environment, short jobs are associated with text and basic page layout components such as the page's background, which are critical web content for the user. On the other hand, large jobs are often lower priority "beautifying" elements, or, even worse, web page banners and advertisements, which are of questionable value to the user as he or she is browsing the page. Using SRSF, pTHINC is able to maximize the utilization of the relatively scarce bandwidth available on the wireless connection between the PDA and the server.

3.1 Display Management

To enable users to just as easily access their web browser and helper applications from a desktop computer at home as from a PDA while on the road, pTHINC provides a resize mechanism to zoom in and out of the display of a web session. pTHINC resizing is completely supported by the server, not the client. The server resamples updates to fit within the PDAs viewport before they are transmitted over the network. pTHINC uses Fant's resampling algorithm to resize pixel updates. This provides smooth, visually pleasing updates with properly antialiasing and has only modest computational requirements.

pTHINC's resizing approach has a number of advantages. First, it allows the PDA to leverage the vastly superior computational power of the server to use high quality resampling algorithms and produce higher quality updates for the PDA to display. Second, resizing the screen does not translate into additional resource requirements for the PDA, since it does not need to perform any additional work. Finally, better utilization of the wireless network is attained since rescaling the updates reduces their bandwidth requirements.

To enable users to orient their displays on a PDA to provide a viewing experience that best accommodates user preferences and the layout of web pages or applications, pTHINC provides a display rotation mechanism to switch between landscape and portrait viewing modes. pTHINC display rotation is completely supported by the client, not the server. pTHINC does not explicitly recalculate the ge-

ometry of display updates to perform rotation, which would be computationally expensive. Instead, pTHINC simply changes the way data is copied into the framebuffer to switch between display modes. When in portrait mode, data is copied along the rows of the framebuffer from left to right. When in landscape mode, data is copied along the columns of the framebuffer from top to bottom. These very fast and simple techniques replace one set of copy operations with another and impose no performance overhead. pTHINC provides its own rotation mechanism to support a wide range of devices without imposing additional feature requirements on the PDA. Although some newer PDA devices provide native support for different orientations, this mechanism is not dynamic and requires the user to rotate the PDA's entire user interface before starting the pTHINC client. Windows Mobile provides native API mechanisms for PDA applications to rotate their UI on the fly, but these mechanisms deliver poor performance and display quality as the rotation is performed naively and is not completely accurate.

3.2 Video Playback

Video has gradually become an integral part of the World Wide Web, and its presence will only continue to increase. Web sites today not only use animated graphics and flash to deliver web content in an attractive manner, but also utilize streaming video to enrich the web interface. Users are able to view pre-recorded and live newscasts on CNN, watch sports highlights on ESPN, and even search through large collection of videos on Google Video. To allow applications to provide efficient video playback, interfaces have been created in display systems that allow video device drivers to expose their hardware capabilities back to the applications. pTHINC takes advantage of these interfaces and its virtual device driver approach to provide a virtual bridge between the remote client and its hardware and the applications, and transparently support video playback.

On top of this architecture, pTHINC uses the YUV colorspace to encode the video content, which provides a number of benefits. First, it has become increasingly common for PDA video hardware to natively support YUV and be able to perform the colorspace conversion and scaling automatically. As a result, pTHINC is able to provide fullscreen video playback without any performance hits. Second, the use of YUV allows for a more efficient representation of RGB data without loss of quality, by taking advantage of the human eye's ability to better distinguish differences in brightness than in color. In particular, pTHINC uses the YV12 format, which allows full color RGB data to be encoded using just 12 bits per pixel. Third, YUV data is produced as one of the last steps of the decoding process of most video codecs, allowing pTHINC to provide video playback in a manner that is format independent. Finally, even if the PDA's video hardware is unable to accelerate playback, the colorspace conversion process is simple enough that it does not impose unreasonable requirements on the PDA.

A more concrete example of how pTHINC leverages the PDA video hardware to support video playback can be seen in our prototype implementation on the popular Dell Axim X51v PDA, which is equipped with the Intel 2700G multimedia accelerator. In this case, pTHINC creates an offscreen buffer in video memory and writes and reads from this memory region data on the YV12 format. When a new video frame arrives, video data is copied from the buffer to

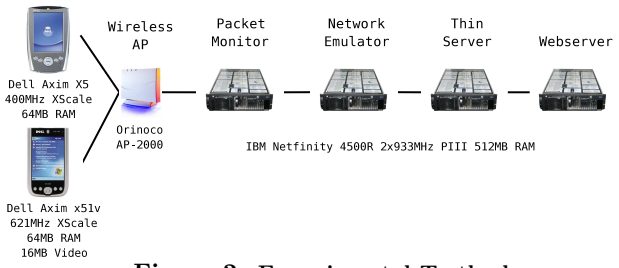

Figure 2: Experimental Testbed

an overlay surface in video memory, which is independent of the normal surface used for traditional drawing. As the YV12 data is put onto the overlay, the Intel accelerator automatically performs both colorspace conversion and scaling. By using the overlay surface, pTHINC has no need to redraw the screen once video playback is over since the overlapped surface is unaffected. In addition, specific overlay regions can be manipulated by leveraging the video hardware, for example to perform hardware linear interpolation to smooth out the frame and display it fullscreen, and to do automatic rotation when the client runs in landscape mode.

4. EXPERIMENTAL RESULTS

We have implemented a pTHINC prototype that runs the client on widely-used Windows Mobile-based Pocket PC devices and the server on both Windows and Linux operating systems. To demonstrate its effectiveness in supporting mobile wireless web applications, we have measured its performance on web applications. We present experimental results on different PDA devices for two popular web applications, browsing web pages and playing video content from the web. We compared pTHINC against native web applications running locally on the PDA to demonstrate the improvement that pTHINC can provide over the traditional fat-client approach. We also compared pTHINC against three of the most widely used thin clients that can run on PDAs, Citrix Meta-FrameXP [2], Microsoft Remote Desktop [3] and VNC (Virtual Network Computing) [16]. We follow common practice and refer to Citrix MetaFrameXP and Microsoft Remote Desktop by their respective remote display protocols, ICA (Independent Computing Architecture) and RDP (Remote Desktop Protocol).

4.1 Experimental Testbed

We conducted our web experiments using two different wireless Pocket PC PDAs in an isolated Wi-Fi network testbed, as shown in Figure 2. The testbed consisted of two PDA client devices, a packet monitor, a thin-client server, and a web server. Except for the PDAs, all of the other machines were IBM Netfinity 4500R servers with dual 933 MHz Intel PIII CPUs and 512 MB RAM and were connected on a switched 100 Mbps FastEthernet network. The web server used was Apache 1.3.27, the network emulator was NISTNet 2.0.12, and the packet monitor was Ethereal 0.10.9. The PDA clients connected to the testbed through a 802.11b Lucent Orinoco AP-2000 wireless access point. All experiments using the wireless network were conducted within ten feet of the access point, so we considered the amount of packet loss to be negligible in our experiments.

Two Pocket PC PDAs were used to provide results across both older, less powerful models and newer higher performance models. The older model was a Dell Axim X5 with

Client	1024×768	640×480	Depth	Resize	Clip
RDP	no	yes	8-bit	no	yes
VNC	yes	yes	16-bit	no	no
ICA	yes	yes	16-bit	yes	no
pTHINC	yes	yes	24-bit	yes	no

Table 2: Thin-client Testbed Configuration Setting

a 400 MHz Intel XScale PXA255 CPU and 64 MB RAM running Windows Mobile 2003 and a Dell TrueMobile 1180 2.4Ghz CompactFlash card for wireless networking. The newer model was a Dell Axim X51v with a 624 MHz Intel XScale XPA270 CPU and 64 MB RAM running Windows Mobile 5.0 and integrated 802.11b wireless networking. The X51v has an Intel 2700G multimedia accelerator with 16MB video memory. Both PDAs are capable of 16-bit color but have different screen sizes and display resolutions. The X5 has a 3.5 inch diagonal screen with 240×320 resolution. The X51v has a 3.7 inch diagonal screen with 480×640.

The four thin clients that we used support different levels of display quality as summarized in Table 2. The RDP client only supports a fixed 640×480 display resolution on the server with 8-bit color depth, while other platforms provide higher levels of display quality. To provide a fair comparison across all platforms, we conducted our experiments with thin-client sessions configured for two possible resolutions, 1024×768 and 640×480. Both ICA and VNC were configured to use the native PDA resolution of 16-bit color depth. The current pTHINC prototype uses 24-bit color directly and the client downsamples updates to the 16-bit color depth available on the PDA. RDP was configured using only 8-bit color depth since it does not support any better color depth. Since both pTHINC and ICA provide the ability to view the display resized to fit the screen, we measured both clients with and without the display resized to fit the PDA screen. Each thin client was tested using landscape rather than portrait mode when available. All systems run on the X51v could run in landscape mode because the hardware provides a landscape mode feature. However, the X5 does not provide this functionality. Only pTHINC directly supports landscape mode, so it was the only system that could run in landscape mode on both the X5 and X51v.

To provide a fair comparison, we also standardized on common hardware and operating systems whenever possible. All of the systems used the Netfinity server as the thin-client server. For the two systems designed for Windows servers, ICA and RDP, we ran Windows 2003 Server on the server. For the other systems which support X-based servers, VNC and pTHINC, we ran the Debian Linux Unstable distribution with the Linux 2.6.10 kernel on the server. We used the latest thin-client server versions available on each platform at the time of our experiments, namely Citrix MetaFrame XP Server for Windows Feature Release 3, Microsoft Remote Desktop built into Windows XP and Windows 2003 using RDP 5.2, and VNC 4.0.

4.2 Application Benchmarks

We used two web application benchmarks for our experiments based on two common application scenarios, browsing web pages and playing video content from the web. Since many thin-client systems including two of the ones tested are closed and proprietary, we measured their performance in a noninvasive manner by capturing network traffic with a packet monitor and using a variant of slow-motion benchmarking [13] previously developed to measure thin-client performance in PDA environments [10]. This measurement methodology accounts for both the display decoupling that can occur between client and server in thin-client systems as well as client processing time, which may be significant in the case of PDAs.

To measure web browsing performance, we used a web browsing benchmark based on the Web Text Page Load Test from the Ziff-Davis i-Bench benchmark suite [7]. The benchmark consists of JavaScript controlled load of 55 pages from the web server. The pages contain both text and graphics with pages varying in size. The graphics are embedded images in GIF and JPEG formats. The original i-Bench benchmark was modified for slow-motion benchmarking by introducing delays of several seconds between the pages using JavaScript. Then two tests were run, one where delays where added between each page, and one where pages where loaded continuously without waiting for them to be displayed on the client. In the first test, delays were sufficiently adjusted 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 run of the benchmark, then used the timestamps of the first and last packet in the trace to obtain our latency measures [10]. The packet monitor also recorded the amount of data transmitted between the client and the server. The ratio between the data traffic in the two tests yields a scale factor. This scale factor shows the loss of data between the server and the client due to inability of the client to process the data quickly enough. The product of the scale factor with the latency measurement produces the true latency accounting for client processing time.

To run the web browsing benchmark, we used Mozilla Firefox 1.0.4 running on the thin-client server for the thin clients, and Windows Internet Explorer (IE) Mobile for 2003 and Mobile for 5.0 for the native browsers on the X5 and X51v PDAs, respectively. In all cases, the web browser used was sized to fill the entire display region available.

To measure video playback performance, we used a video benchmark that consisted of playing a 34.75s MPEG-1 video clip containing a mix of news and entertainment programming at full-screen resolution. The video clip is 5.11 MB and consists of 834 352x240 pixel frames with an ideal frame rate of 24 frames/sec. We measured video performance using slow-motion benchmarking by monitoring resulting packet traffic at two playback rates, 1 frames/second (fps) and 24 fps, and comparing the results to determine playback delays and frame drops that occur at 24 fps to measure overall video quality [13]. For example, 100% quality means that all video frames were played at real-time speed. On the other hand, 50% quality could mean that half the video data was dropped, or that the clip took twice as long to play even though all of the video data was displayed.

To run the video benchmark, we used Windows Media Player 9 for Windows-based thin-client servers, MPlayer 1.0 pre 6 for X-based thin-client servers, and Windows Media Player 9 Mobile and 10 Mobile for the native video players running locally on the X5 and X51v PDAs, respectively. In all cases, the video player used was sized to fill the entire display region available.

4.3 Measurements

Figures 3 and 4 show the results of running the web brows-

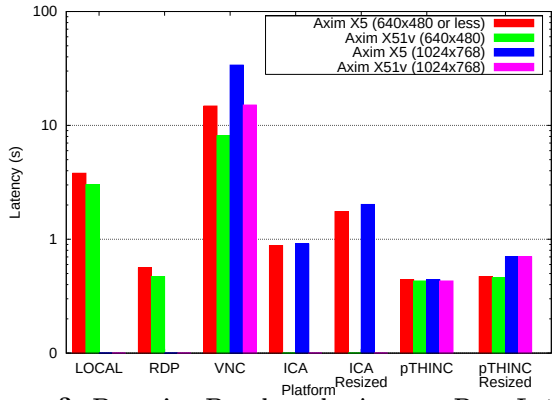

Figure 3: Browsing Benchmark: Average Page Latency

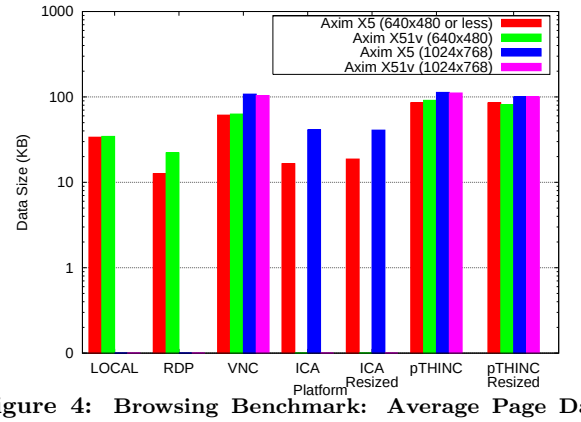

Figure 4: Browsing Benchmark: Average Page Data Transferred

ing benchmark. For each platform, we show results for up to four different configurations, two on the X5 and two on the X51v, depending on whether each configuration was supported. However, not all platforms could support all configurations. The local browser only runs at the display resolution of the PDA, 480×680 or less for the X51v and the X5. RDP only runs at 640×480. Neither platform could support 1024×768 display resolution. ICA only ran on the X5 and could not run on the X51v because it did not work on Windows Mobile 5.

Figure 3 shows the average latency per web page for each platform. pTHINC provides the lowest average web browsing latency on both PDAs. On the X5, pTHINC performs up to 70 times better than other thin-client systems and 8 times better than the local browser. On the X51v, pTHINC performs up to 80 times better than other thin-client systems and 7 times better than the native browser. In fact, all of the thin clients except VNC outperform the local PDA browser, demonstrating the performance benefits of the thin-client approach. Usability studies have shown that web pages should take less than one second to download for the user to experience an uninterrupted web browsing experience [14]. The measurements show that only the thin clients deliver subsecond web page latencies. In contrast, the local browser requires more than 3 seconds on average per web page. The local browser performs worse since it needs to run a more limited web browser to process the HTML, JavaScript, and do all the rendering using the limited capabilities of the PDA. The thin clients can take advantage of faster server hardware and a highly tuned web browser to process the web content much faster.

Figure 3 shows that RDP is the next fastest platform after pTHINC. However, RDP is only able to run at a fixed resolution of 640×480 and 8-bit color depth. Furthermore, RDP also clips the display to the size of the PDA screen so that it does not need to send updates that are not visible on the PDA screen. This provides a performance benefit assuming the remaining web content is not viewed, but degrades performance when a user scrolls around the display to view other web content. RDP achieves its performance with significantly lower display quality compared to the other thin clients and with additional display clipping not used by other systems. As a result, RDP performance alone does not provide a complete comparison with the other platforms. In contrast, pTHINC provides the fastest performance while at the same time providing equal or better display quality than the other systems.

Since VNC and ICA provide similar display quality to pTHINC, these systems provide a more fair comparison of different thin-client approaches. ICA performs worse in part because it uses higher-level display primitives that require additional client processing costs. VNC performs worse in part because it loses display data due to its client-pull delivery mechanism and because of the client processing costs in decompressing raw pixel primitives. In both cases, their performance was limited in part because their PDA clients were unable to keep up with the rate at which web pages were being displayed.

Figure 3 also shows measurements for those thin clients that support resizing the display to fit the PDA screen, namely ICA and pTHINC. Resizing requires additional processing, which results in slower average web page latencies. The measurements show that the additional delay incurred by ICA when resizing versus not resizing is much more substantial than for pTHINC. ICA performs resizing on the slower PDA client. In contrast, pTHINC leverage the more powerful server to do resizing, reducing the performance difference between resizing and not resizing. Unlike ICA, pTHINC is able to provide subsecond web page download latencies in both cases.

Figure 4 shows the data transferred in KB per page when running the slow-motion version of the tests. All of the platforms have modest data transfer requirements of roughly 100 KB per page or less. This is well within the bandwidth capacity of Wi-Fi networks. The measurements show that the local browser does not transfer the least amount of data. This is surprising as HTML is often considered to be a very compact representation of content. Instead, RDP is the most bandwidth efficient platform, largely as a result of using only 8-bit color depth and screen clipping so that it does not transfer the entire web page to the client. pTHINC overall has the largest data requirements, slightly more than VNC. This is largely a result of the current pTHINC prototype's lack of native support for 16-bit color data in the wire protocol. However, this result also highlights pTHINC's performance as it is faster than all other systems even while transferring more data. Furthermore, as newer PDA models support full 24-bit color, these results indicate that pTHINC will continue to provide good web browsing performance.

Since display usability and quality are as important as performance, Figures 5 to 8 compare screenshots of the different thin clients when displaying a web page, in this case from the popular BBC news website. Except for ICA, all of the screenshots were taken on the X51v in landscape mode

Figure 5: Browser Screenshot: RDP 640x480

Figure 6: Browser Screenshot: VNC 1024x768

Figure 7: Browser Screenshot: ICA Resized 1024x768

Figure 8: Browser Screenshot: pTHINC Resized 1024x768

using the maximum display resolution settings for each platform given in Table 2. The ICA screenshot was taken on the X5 since ICA does not run on the X51v. While the screenshots lack the visual fidelity of the actual device display, several observations can be made. Figure 5 shows that RDP does not support fullscreen mode and wastes lots of screen space for controls and UI elements, requiring the user to scroll around in order to access the full contents of the web browsing session. Figure 6 shows that VNC makes better use of the screen space and provides better display quality, but still forces the user to scroll around to view the web page due to its lack of resizing support. Figure 7 shows ICA's ability to display the full web page given its resizing support, but that its lack of landscape capability and poorer resize algorithm significantly compromise display quality. In contrast, Figure 8 shows pTHINC using resizing to provide a high quality fullscreen display of the full width of the web page. pTHINC maximizes the entire viewing region by moving all controls to the PDA buttons. In addition, pTHINC leverages the server computational power to use a high quality resizing algorithm to resize the display to fit the PDA screen without significant overhead.

Figures 9 and 10 show the results of running the video playback benchmark. For each platform except ICA, we show results for an X5 and X51v configuration. ICA could not run on the X51v as noted earlier. The measurements were done using settings that reflected the environment a user would have to access a web session from both a desktop computer and a PDA. As such, a 1024×768 server display resolution was used whenever possible and the video was shown at fullscreen. RDP was limited to 640×480 display resolution as noted earlier. Since viewing the entire video display is the only really usable option, we resized the display to fit the PDA screen for those platforms that supported this feature, namely ICA and pTHINC.

Figure 9 shows the video quality for each platform. pTHINC is the only thin client able to provide perfect video playback quality, similar to the native PDA video player. All of the other thin clients deliver very poor video quality. With the exception of RDP on the X51v which provided unacceptable 35% video quality, none of the other systems were even able to achieve 10% video quality. VNC and ICA have the worst quality at 8% on the X5 device.

pTHINC's native video support enables superior video performance, while other thin clients suffer from their inability to distinguish video from normal display updates. They attempt to apply ineffective and expensive compression algorithms on the video data and are unable to keep up with the stream of updates generated, resulting in dropped frames or long playback times. VNC suffers further from its client-pull update model because video frames are generated faster than the rate at which the client can process and send requests to the server to obtain the next display update. Figure 10 shows the total data transferred during

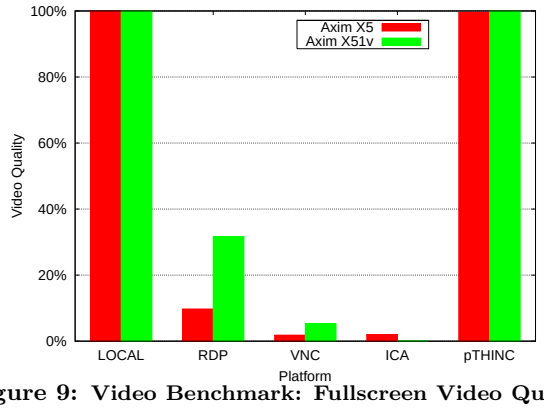

Figure 9: Video Benchmark: Fullscreen Video Quality

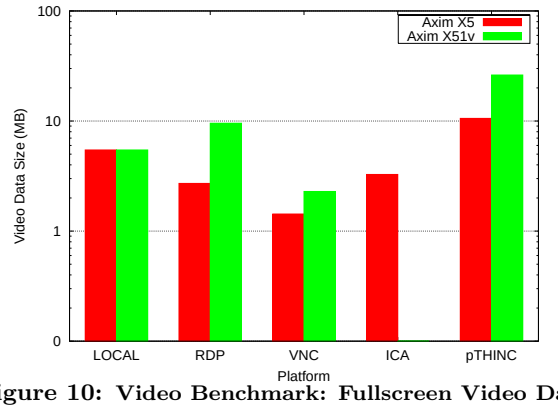

Figure 10: Video Benchmark: Fullscreen Video Data

video playback for each system. The native player is the most bandwidth efficient platform, sending less than 6 MB of data, which corresponds to about 1.2 Mbps of bandwidth. pTHINC's 100% video quality requires about 25 MB of data which corresponds to a bandwidth usage of less than 6 Mbps. While the other thin clients send less data than THINC, they do so because they are dropping video data, resulting in degraded video quality.

Figures 11 to 14 compare screenshots of the different thin clients when displaying the video clip. Except for ICA, all of the screenshots were taken on the X51v in landscape mode using the maximum display resolution settings for each platform given in Table 2. The ICA screenshot was taken on the X5 since ICA does not run on the X51v. Figures 11 and 12 show that RDP and VNC are unable to display the entire video frame on the PDA screen. RDP wastes screen space for UI elements and VNC only shows the top corner of the video frame on the screen. Figure 13 shows that ICA provides resizing to display the entire video frame, but did not proportionally resize the video data, resulting in strange display artifacts. In contrast, Figure 14 shows pTHINC using resizing to provide a high quality fullscreen display of the entire video frame. pTHINC provides visually more appealing video display than RDP, VNC, or ICA.

5. RELATED WORK

Several studies have examined the web browsing performance of thin-client computing [13, 19, 10]. The ability for thin clients to improve web browsing performance on wireless PDAs was first quantitatively demonstrated in a previous study by one of the authors [10]. This study demonstrated that thin clients can provide both faster web browsing performance and greater web browsing functionality. The study considered a wide range of web content including content from medical information systems. Our work builds on this previous study and consider important issues such as how usable existing thin clients are in PDA environments, the trade-offs between thin-client usability and performance, performance across different PDA devices, and the performance of thin clients on common web-related applications such as video.

Many thin clients have been developed and some have PDA clients, including Microsoft's Remote Desktop [3], Citrix MetraFrame XP [2], Virtual Network Computing [16, 12], GoToMyPC [5], and Tarantella [18]. These systems were first designed for desktop computing and retrofitted for PDAs. Unlike pTHINC, they do not address key system architecture and usability issues important for PDAs. This limits their display quality, system performance, available screen space, and overall usability on PDAs. pTHINC builds on previous work by two of the authors on THINC [1], extending the server architecture and introducing a client interface and usage model to efficiently support PDA devices for mobile web applications.

Other approaches to improve the performance of mobile wireless web browsing have focused on using transcoding and caching proxies in conjunction with the fat client model [11, 9, 4, 8]. They work by pushing functionality to external proxies, and using specialized browsing applications on the PDA device that communicate with the proxy. 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 and helper applications to work seamlessly with production systems without any additional proxy configuration or web browser modifications.

With the emergence of web browsing on small display devices, web sites have been redesigned using mechanisms like WAP and specialized native web browsers have been developed to tailor the needs of these devices. Recently, Opera has developed the Opera Mini [15] web browser, which uses an approach similar to the thin-client model to provide access across a number of mobile devices that would normally be incapable of running a web browser. Instead of requiring the device to process web pages, it uses a remote server to pre-process the page before sending it to the phone.

6. CONCLUSIONS

We have introduced pTHINC, a thin-client architecture for wireless PDAs. pTHINC provides key architectural and usability mechanisms such as server-side screen resizing, client-side screen rotation using simple copy techniques, YUV video support, and maximizing screen space for display updates and leveraging existing PDA control buttons for UI elements. pTHINC transparently supports traditional desktop browsers and their helper applications on PDA devices and desktop machines, providing mobile users with ubiquitous access to a consistent, personalized, and full-featured web environment across heterogeneous devices. We have implemented pTHINC and measured its performance on web applications compared to existing thin-client systems and native web applications. Our results on multiple mobile wireless devices demonstrate that pTHINC delivers web browsing performance up to 80 times better than existing thin-client systems, and 8 times better than a native PDA browser. In addition, pTHINC is the only PDA thin client

Figure 11: Video Screenshot: RDP 640x480

Figure 12: Video Screenshot: VNC 1024x768

Figure 13: Video Screenshot: ICA Resized 1024x768

Figure 14: Video Screenshot: pTHINC Resized 1024x768

that transparently provides full-screen, full frame rate video playback, making web sites with multimedia content accessible to mobile web users.

7. ACKNOWLEDGEMENTS

This work was supported in part by NSF ITR grants CCR-0219943 and CNS-0426623, and an IBM SUR Award.

8. REFERENCES

[1] R. Baratto, L. Kim, and J. Nieh. THINC: A Virtual Display Architecture for Thin-Client Computing. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP)*, Oct. 2005.

[2] Citrix Metaframe. `http://www.citrix.com`.

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

[4] 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.

[5] GoToMyPC. `http://www.gotomypc.com/`.

[6] Health Insurance Portability and Accountability Act. `http://www.hhs.gov/ocr/hipaa/`.

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

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

[9] 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.

[10] A. Lai, J. Nieh, B. Bohra, V. Nandikonda, A. P. Surana, and S. Varshneya. Improving Web Browsing on Wireless PDAs Using Thin-Client Computing. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, May 2004.

[11] A. Maheshwari, A. Sharma, K. Ramamritham, and P. Shenoy. TranSquid: Transcoding and caching proxy for heterogenous ecommerce environments. In *Proceedings of the 12th IEEE Workshop on Research Issues in Data Engineering (RIDE '02)*, Feb. 2002.

[12] .NET VNC Viewer for PocketPC. `http://dotnetvnc.sourceforge.net/`.

[13] 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.

[14] J. Nielsen. *Designing Web Usability*. New Riders Publishing, Indianapolis, IN, 2000.

[15] Opera Mini Browser. `http://www.opera.com/products/mobile/operamini/`.

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

[17] R. W. Scheifler and J. Gettys. The X Window System. *ACM Trans. Gr.*, 5(2):79–106, Apr. 1986.

[18] Sun Secure Global Desktop. `http://www.sun.com/software/products/sgd/`.

[19] S. J. Yang, J. Nieh, S. Krishnappa, A. Mohla, and M. Sajjadpour. Web Browsing Performance of Wireless Thin-Client Computing. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, May 2003.