Seeds of Internet Growth

Actively Deployable Mobile Services for Adaptive Web Access

The proliferation of mobile devices in wireless environments has put special requirements on the ability to access the Web seamlessly. To address the impact of varying contextual characteristics on mobile access, the authors developed the WebPADS framework, which can actively deploy new mobile services. Moreover, WebPADS can dynamically reconfigure its services and migrate them to adapt to the vigorous changes in the wireless environment.

Siu Nam Chuang, Alvin T.S. Chan, Jiannong Cao, and Ronnie Cheung
Hong Kong Polytechnic University

The convergence of wireless communication and portable devices has driven mobile computing's rapid advance, which in turn has led to significant improvements in wireless accessibility and the development of smaller but more powerful devices. However, the underlying technologies driving the Web are still based largely on wired communication, which assumes network characteristics such as large bandwidth availability, low error rates, and always-on connectivity. A wireless environment invalidates these assumptions.

Techniques and approaches for overcoming these obstacles are well researched and documented, and some have demonstrated promising results.[1–3] However, the proposed solutions[2,3] noticeably lack flexibility in the aspects we deem most important for supporting a robust Web service architecture in a wireless environment. First, new services should be deployable to the operating environment actively during runtime, without interrupting the execution of existing applications; this would let mobile devices benefit from new services anytime, anywhere, including foreign networks. Second, the Web architecture must be able to detect and adapt to environmental changes dynamically so that services can reconfigure themselves to optimize the mobile device's operations. Finally, the architecture must support service migration to maintain the service's provision as the mobile node moves across different domains. We refer to these three features as *active deployment*, *dynamic reconfiguration*, and *service migration*.

To address these challenges, we designed and implemented the Web Proxy for Actively Deployable Services (WebPADS) framework,[4] which aims to exploit the desired features over a wireless network. Consisting of a proxy client and a proxy server, the WebPADS system acts as a Web proxy system to a Web client. By intercept-

Table 1. Comparison of mobile middleware features.

	WAP	i-mode	WBI	WebPADS
TCP/IP	WAP protocol stack	Original TCP/IP	Original TCP/IP	Original TCP/IP
HTTP	WML	Original HTTP	Modified HTTP	Modified HTTP
Supports generic applications	No	No	Yes	Yes
Client–proxy approach	Yes	No	No	Yes
Wireless awareness	Yes	Yes	No	Yes
Active deployment	No	Yes	Yes	Yes
Dynamic adaptation	No	No	No	Yes
Mobility support	Yes	Yes	No	Yes

ing Web traffic on both ends of a wireless link, WebPADS can transform and optimize the traffic for maximum performance. A WebPADS system can dynamically reconfigure its services to adapt to the wireless environment's vigorous changes. Furthermore, a WebPADS proxy client can migrate WebPADS-provided services from one server to another to track and serve the mobile node continuously.

Middleware's Role

For years, application developers have successfully used middleware technologies as intermediaries for providing transparent network services or as environments for network applications. However, completely hiding implementation details from applications makes little sense in a mobile environment because mobile systems need to adapt quickly. Application designers thus need a new form of awareness, not just transparency, to let them inspect the application's context and adapt the middleware's behavior accordingly.

Table 1 compares WebPADS' characteristics to those of the systems described in the "Related Work in Mobile Web Access" sidebar on page 29. One of the mobile Web middleware's most desirable features is its ability to support existing applications and interoperate over existing network infrastructures. As such, it is desirable that the system uses HTTP and the TCP/IP protocol stack, but with enhancements that optimize traffic on the wireless link.

WebPADS Framework

WebPADS' general service architecture follows a client–proxy model: as Figure 1 (next page) shows, the architecture has a client in the mobile node and a server in the fixed network. This client-server pair intercepts, transforms, and optimizes all HTTP requests and responses for transport over the wireless network. WebPADS-provided services are composed of chains of service objects called *mobilets*, which are service entities that can download, push, or migrate mobile code to a WebPADS platform for execution within the client–proxy WebPADS environment. Mobilets exist in pairs: a master mobilet resides at the WebPADS client, and a slave mobilet resides at the WebPADS server. A pair of mobilets cooperates to provide a specific service — for example, when a slave mobilet shares a major portion of the processing burden, the master mobilet controls the processing of the slave mobilet and presents the processed output to the mobile application.

WebPADS' service-chain model enables great flexibility. We can establish a consistent synchronization and dataflow model by abstracting a *virtual channel* to provide an enhanced communication service over the wireless link. To do so, the WebPADS client chains its mobilets together in a specific order and instructs the WebPADS server to chain the corresponding peer mobilets together in a nested order. The service chain model offers flexibility because the chain can be dynamically reconfigured to adapt to changes in the wireless environment without interrupting the service provision to other mobile nodes. Furthermore, mobilets can migrate to a new proxy server when the mobile node moves to a different network domain.

Active Deployment

A static client–proxy architecture that offers fixed functionality can't cope with the increasing demand for innovative wireless services. Moreover, mobile nodes and all fixed-network agents must be dynamically updateable to ensure compatible, consistent operation. To tackle the varying characteristics of a wireless environment, the WebPADS framework supports deploying new mobilets actively from a WebPADS client to a WebPADS server. The node carries relevant mobilets with it as it travels across foreign domains, and as the need arises, the client sends

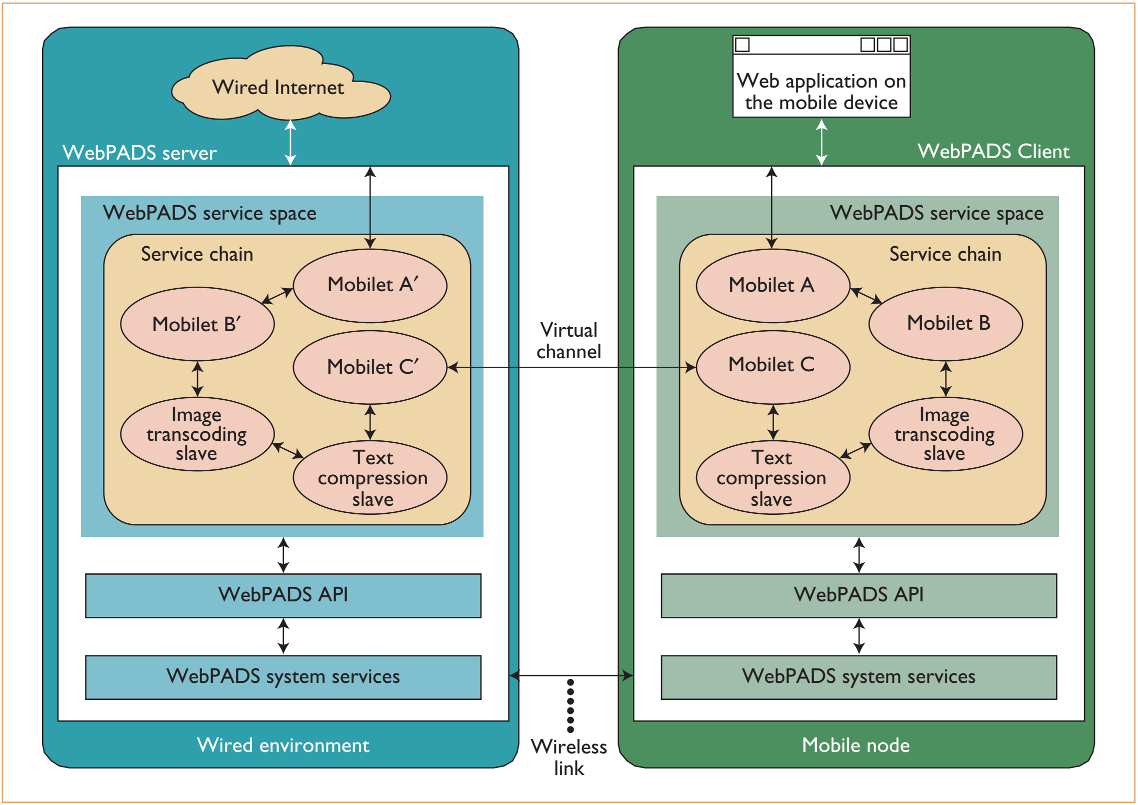

Figure 1. The WebPADS system architecture. The WebPADS proxy client and server coordinate to act as a Web proxy system to the Web application on the mobile device, which intercepts and optimizes all HTTP traffic for transport over the wireless network .

mobilets to a WebPADS proxy server, which configures them to operate in a coordinated manner.

Contextual Events

The capability to detect contextual changes is an essential requirement for adaptive Web access. Accordingly, WebPADS contains an event framework that supports a hierarchical event detection and reporting subsystem. The WebPADS event framework is designed to give developers access to event-monitoring and notification functionalities. The framework provides a direct set of APIs and an application-specific XML-based scripting language for event composition in the system. We have defined a list of primitive events that monitor various attributes of CPU, memory, storage device, network, and power. The WebPADS system also supports event composition by monitoring multiple events across various environmental contexts. A mobilet, for example, can subscribe to a *composite* event that monitors multiple contextual events, such as `network-availability`. This composite event source monitors a list of primitive events under the *network* event type. These events represent different attributes of a network interface — for example, `network-maximum-data-rate` or `network-delay`. Based on the real time statuses of these primitive events, the `network-availability` event reports four coarse status levels: high, medium, low, and unavailable. High availability means that the network link is at its optimal status: minimal packet drop rate, low delay, and available bandwidth close to its practical limit. Medium availability means that a minor portion of the packets will be delayed, packet drop could occur infrequently, and the bandwidth is only a portion of its practical limit. Low availability suggests that packet drop occurs frequently, packets could be severely delayed, and the available bandwidth is very limited. Unavailable indicates no packet can be transported over this network link, which could cause a disconnection.

As with a primitive event source that monitors a single attribute of a context, multiple subscribers can subscribe to the same composite event. A subscriber can also compose new composite events that best suit its interest. In this way, a subscriber can avoid

Related Work in Mobile Web Access

The innovations in hardware and software technologies have greatly accelerated the evolution of mobile computing. In particular, mobile middleware aims to support ubiquitous wireless Web access, regardless of a user's location or access device.

The Wireless Application Protocol (WAP; www.wapforum.org) connects cellular networks to the Internet. It also alleviates the low bandwidth and protocol inefficiency problems inherent in cellular networks. Currently, all WAP clients interpret only the simplified Wireless Markup Language (WML) and WMLscript that form the baseline formatting language for presenting Web content over cellular networks. The divergence from mainstream HTML means that all Web content must be translated to its WML equivalent before a WAP client's limited display screen can render it. Although the latest version of WML is compliant with XHTML Basic (http://www.w3.org/TR/xhtml-basic/), WAP is still unable to support non-XHTML Basic compliant documents directly. Supporting end-to-end wireless Web access requires a more generic architecture that seamlessly integrates itself into the Internet environment.

i-mode (www.nttdocomo.co.jp/english/p_s/imode/) is the first always-connected mobile service serving atop a packet-switching cellular network. i-mode lets its handset users connect to any Web site via compact HTML (cHTML; www.w3.org/TR/1998/NOTE-compactHTML-19980209/), which defines a subset of HTML for small information appliances such as smart phones and PDAs without using dial-up access. i-mode also supports i-appli, which is a customized, downloadable version of Java 2 Micro Edition (J2ME) that runs on i-mode phones. A recent report (see www.eurotechnology.com/imode/faq.html) showed that, in 2000, Japan's 15 million i-mode users made up 60 percent of the world's wireless Internet population. However, i-mode's success is rooted not in its technology, but in its operational model, which requires exceptional ease of use, enriched content, and inexpensive rates for both users and content providers.

Web Intermediaries (WBI; www.almaden.ibm.com/cs/wbi/) is a programmable HTTP proxy designed to support development and deployment of intermediary Web applications, which can be located anywhere along a data stream, to process or enhance the data as it passes through.[1] WBI supports flexible composition of services via MEG elements — monitors, editors, and generators. These elements support HTTP request processing, HTTP response generating and processing, and monitoring of both requests and responses. However, the binding of elements is static, so once configured, it is impossible to reconfigure the composition without stalling the WBI at runtime. Therefore, WBI cannot support dynamic configuration of these elements to respond to the wireless link's changing characteristics.

Mobile middleware shows encouraging results in both performance improvements and value-added services for wireless Web access. However, the lack of adaptation support for current middleware systems limits their effectiveness for wireless applications, which must be aware of and adapt to the changes in the wireless environment.

Reference

1. R. Barrett and P.P. Maglio. "Intermediaries: New Places for Producing and Manipulating Web Content," *Proc. 7th Int'l World Wide Web Conf.*, Elsevier Science, 1998, pp. 509–518.

the repeated effort of subscribing to and checking all primitive events, but it still receives enough contextual information to determine its adaptation reaction. Various adaptation actions could occur at different spots: a mobile application's internal logic branching, a service chain's dynamic reconfiguration, and a mobilet's behavior adjustment.

Dynamic Reconfiguration

To adjust services provided by the WebPADS system, the WebPADS proxy client coordinates the dynamic service reconfiguration process to best adapt to the current context. On initialization of a reconfiguration process, based on descriptions of an XML configuration file, the WebPADS client forwards the concerned event and the associated new service-chain map to the WebPADS server. When specific contextual changes trigger a dynamic service reconfiguration, the mobilets specified in the new service-chain map first go through source-code-loading procedures to locate all the new mobilet classes' sources. (This same process occurs simultaneously at both the WebPADS client and server.) When all the new mobilet classes are ready, the current service composition undergoes a service deletion–addition process, during which the WebPADS proxy client and server reconfigure the existing service composition to match the new service-chain map's description. Once the current service composition is updated, the mobilets' execution threads resume, and service execution recommences.

Migration

Although dynamic reconfiguration aims to adaptively configure mobilet services on the basis of the localized operating environment, it does not address the issue of adaptability as the mobile unit moves across network domains. The WebPADS server and client operate in a geographically collocated domain to minimize transit latency. A WebPADS client might initially be set up to operate within domain A, for example, with WebPADS server A acting as a peer proxy. If the client then moves to domain B, which is geographically dis-

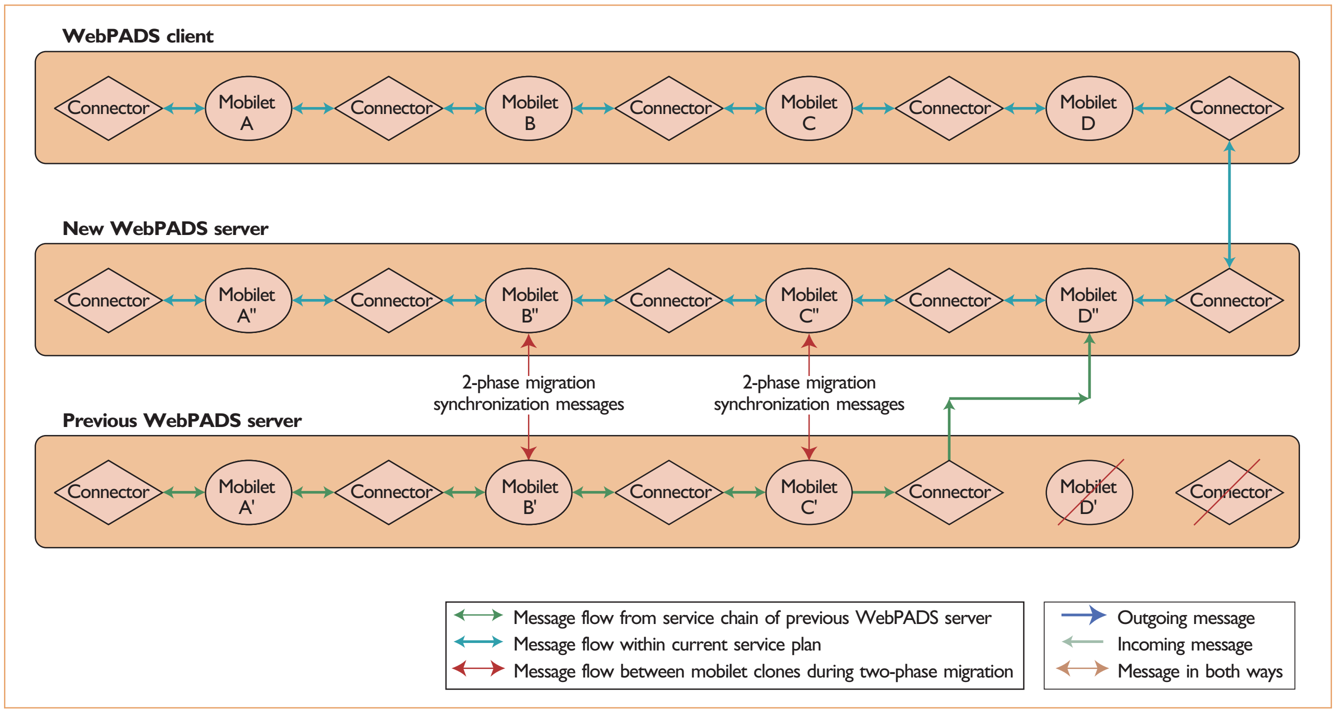
WebPADS client
Connector
Mobilet A
Mobilet B
Mobilet C
Mobilet D
New WebPADS server
Mobilet A''
Mobilet B''
Mobilet C''
Mobilet D''
2-phase migration synchronization messages
2-phase migration synchronization messages
Previous WebPADS server
Mobilet A'
Mobilet B'
Mobilet C'
Mobilet D'
Message flow from service chain of previous WebPADS server
Message flow within current service plan
Message flow between mobilet clones during two-phase migration
Outgoing message
Incoming message
Message in both ways

Figure 2. Message flow during two-phase migration. Two-phase migration minimizes the interruption caused when mobilets B′ and C′ cannot be migrated instantly in a normal migration process.

tant from domain A, information flow via server A becomes suboptimal, possibly incurring unnecessary latency. When the WebPADS client notices the relocation by monitoring IP changes and discovers a new WebPADS server via the service location protocol (SLP), it coordinates a transfer of the service chain from server A to WebPADS server B in domain B.

Two-Phase Migration

There are cases when a mobilet cannot be migrated immediately to the new WebPADS server. A mobilet could be performing operations that would make it difficult to migrate or hard to resume after migration. Examples of these operations include time-consuming I/O operations, location-dependent transactions, or calculations that generate huge amounts of temporary data. In such cases, the migration subsystem preserves the ongoing communication session through a two-phase migration process, with the user experiencing minimal interruption. Not all mobilets in the service chain require a two-phase migration during the migration process, though. When a mobilet requests two-phase migration, the previous WebPADS server in the chain must retain all mobilets in the incoming message direction — for example, look at the mobilets on the left-hand side (A′ and B′) of requesting mobilet (C′) in Figure 2. If more than one mobilet requests two-phase migration, the last one in the outgoing message direction becomes the service chain's new endpoint during migration.

As Figure 2 shows, mobilets B′ and C′ have requested the two-phase migration service. Mobilet A′, which is in the incoming message direction of mobilets B′ and C′, is retained to allow outgoing messages to flow into mobilets B′ and C′. In Figure 2, connectors provide a unified message-forwarding function for mobilets, such that the sender need not know the recipient's details. Moreover, the connector is also responsible for state flooding in which an instruction from the WebPADS system can spread the state transition command throughout the entire service chain.

The retained service chain at the previous WebPADS server maintains the service in two ways. First, it forwards the processed message stream in the outgoing direction to the service chain at the new WebPADS server. This technique hides the changes in the Web proxy from the Web server currently interacting with the client. This lets existing transactions continue processing while the system is in the two-phase migration state, which in turn avoids abandoning transactions or having to wait until all transactions complete before performing service migration. Second, it supports direct messaging between mobilets in the two-phase migration process to synchronize their states — for example, mobilets B′ and C′ at the previous server can synchronize with their cloned counterparts at the new server. This technique allocates extended migration preparation time to the mobilets. The mobilet at the previous WebPADS server thus can continue updating its clone at the new server until that clone can completely replace it.

Implementation

The WebPADS runtime exposes several levels of APIs for mobilets and mobile applications that use WebPADS services. Through its APIs, WebPADS supports naming, messaging, context monitoring, adaptation, and mobility. To facilitate flexible and robust deployment as well as mobilet migration, we chose Java as the system's development language, and used several of its properties:

- *Dynamic class loading* lets the Java virtual machine (JVM) load and define classes at runtime. The class-loading mechanism is extensible and enables mobilet classes to be loaded via the network.
- *Multithreading* lets an object execute in its own thread of execution, which in turn allows parallel execution of different object functions. As such, system components and mobilets can be designed to behave autonomously. Together with the provision of synchronization primitives in Java, WebPADS can use asynchronous messaging, which allows for pipelined message processing.
- *Serialization* represents an object's states in a serialized form, which makes them transferable over the network for later reconstruction. This mechanism maps directly to the requirement for migrating mobilets across clients and servers, thus simplifying the migration process.

Java, being an object-oriented, network-aware, robust, portable, multithreaded, dynamic language, is an ideal platform for developing the WebPADS system.

A Case Example

We fully implemented the WebPADS system using Java to allow us to evaluate the system's performance based on empirical results. To do so, we conducted experiments with four PCs in an emulated wireless environment. One PC acted as the mobile node, we configured another to act as a wireless router controlling the characteristics of the emulated wireless link using the NIST Net network emulator (http://snad.ncsl.nist.gov/itg/nistnet/). We then connected the mobile node to the Internet via the wireless router. The other two PCs acted as WebPADS servers residing on the wired LAN. These servers and the wireless router were located on the same fixed LAN within the Hong Kong Polytechnic University's campus network.

We tested the WebPADS system's performance under various bandwidths, ranging from 20 Kbps to 1 Mbps. We conducted four sets of experiments, measuring the time consumption and efficiency of the following operations:

- active deployment of two to 20 mobilets, with mobilet sizes of 10 Kbytes and 20 Kbytes;
- dynamic reconfiguration of two service chain setups, one with two mobilet removals and two mobilet additions, the other with five mobilet removals and five mobilet additions;
- migration of two to 20 mobilets, with mobilet sizes of 2 Kbytes and 10 Kbytes; and
- retrieval of Web content through WebPADS, with results compared to a direct Internet connection (Web content size is 1 Mbyte, with three variations — image intensive, text intensive, and half image/half text).

The results showed that the active deployment procedure fully uses the limited bandwidth, which means that the service deployment time was close to the optimal value (bounded by bandwidth limitation). We determined active deployment time from service code size and available bandwidth, plus a delay of less than one second caused by WebPADS. For dynamic reconfiguration, our results showed that the overall reconfiguration time was relatively insensitive to bandwidth variations and always below two seconds. The service suspension time during reconfiguration was also less than 0.6 seconds. Bandwidth variation had limited effect on migration time; the majority of migration delay resulted from system overhead, including WebPADS, the JVM, and the operating system. Service migration with active deployment and two-phase migration did not introduce extra overhead to the overall migration process (in which the active deployment portion was bounded by the bandwidth, and the two-phase migration portion was determined by the requirements of the mobilet undergoing two-phase migration). In this

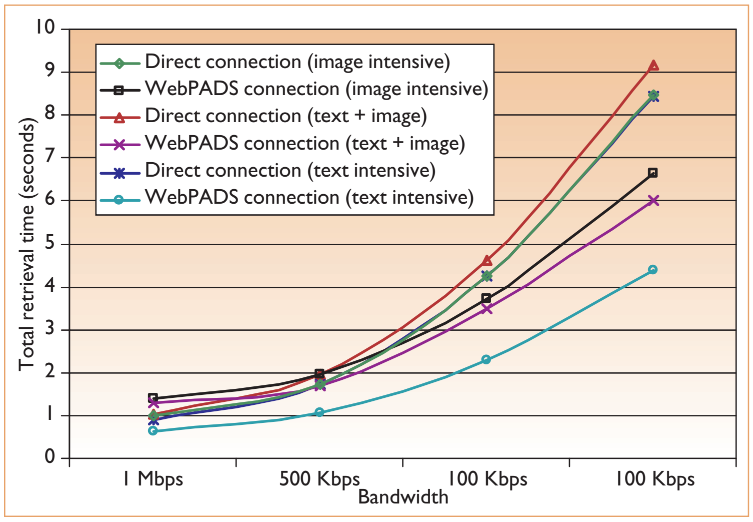

Figure 3. Web content retrieval time, comparing WebPADS and a direct Internet connection. Results showed that WebPADS performed better for bandwidth with less than 500 Kbps when retrieving typical Web content.

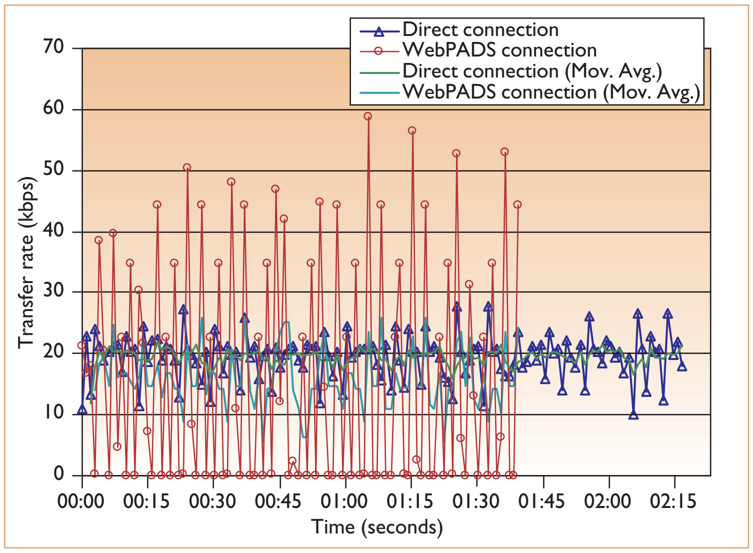

Figure 4. Comparison of Web traffic patterns over a 200-Kbps wireless link. Compared to a direct Internet connection, the results show that WebPADS generated less traffic and took less time to retrieve the same set of Web content.

process, the service suspension time during migration is unaffected by active deployment and two-phase migration. Without the need for active deployment and two-phase migration, a chain of 20 mobilets can be migrated within 2.4 seconds using a wireless link of 20 Kbps. In a normal deployment scenario, a service chain usually contains fewer than 20 mobilets, which means that service migration takes less than 2.4 seconds.

To demonstrate the WebPADS framework in action, we developed a set of HTTP mobilet services that exploits the framework's benefits. We developed two mobilets for HTTP services:

- A text-compression mobilet pair compresses text in the HTTP stream at the WebPADS server and decompresses the HTML files at the WebPADS client.
- An image-transcoding mobilet pair can convert JPEG images to GIF, convert color images to grayscale, and resize images.

We set the image-transcoding mobilets to perform only format conversion, disabling the resizing function. Comparing a direct Internet connection to the Internet connection through WebPADS, we measured the total retrieval times for 1 Mbyte of Web content under various operating bandwidths. Figure 3 shows the measured results for three sets of content: text-intensive, half-text and half-image, and image-intensive Web pages.

For retrieving text-intensive pages, WebPADS was significantly faster under all tested bandwidths, mainly due to the text-compression mobilets' high efficiency: they roughly halved the data volume while incurring only minimal processing delay. The image-transcoding service was comparatively less efficient: its average data-volume reduction was only around 20 percent, which was not sufficient to offset its processing delay under high bandwidths. As the curve for mixed content types shows, WebPADS' performance in processing mixed text and image contents closely models normal Web traffic. Excluding the results for text-intensive content, WebPADS performed poorer than the direct Internet connection when the bandwidth was higher than 500 Kbps and better when bandwidth was lower than that. This result is not surprising: the significant reduction in data required to send text and images across a bandwidth-limited wireless environment offsets the computing delays incurred in compressing that content.

To better understand WebPADS' bandwidth usage, we captured traffic patterns for both WebPADS and direct Internet connections. We performed the experiment over a 200-Kbps link, in which each connection retrieved a total of 3 Mbytes of Web page content (half text and half image). Figure 4 shows the measured and the 4-second average transfer rates for both connections. WebPADS' transfer rate fluctuation was much greater than the direct Internet connection's. Such bursty traffic is due to the constant processing delay the mobilets caused — the delay created intervals of network idle times. Due to the reduced data volume created by text compression and image-transcoding mobilets, the total transfer time for WebPADS was still significantly shorter than the direct connection's.

Future Work

In addition to developing the set of HTTP services we described earlier, we've also successfully used the WebPADS platform to build a Web metasearching service within a mobile environment.[5] To truly realize the system's benefits, however, we must have other forms of HTTP mobilets, so that most Web applications could benefit from the services that WebPADS provides in various operational environments. Mobilets that act as personalization and intelligent agents could further enhance the user experience, for example. Services such as Web page layout reformatting and page caching and prefetching also could benefit from the mobilet chain model, further improving performance and user experience.

A major concern with small portable devices is power consumption. In our experiments, we noticed that executing the WebPADS client did not increase the mobile device's CPU usage significantly because most of the computation-intensive tasks occurred on the server side. However, we note that a poorly implemented mobilet pair could potentially exhaust the mobile device's computational resources. As part of our future work, we will investigate mobilet resource management and extend the mobilet model to support QoS-aware mobilets. We also intend to port WebPADS to operate over a J2ME platform. This will let WebPADS clients run on PDAs and Java-enabled smart phones. We envision power-aware mobilet services optimizing the device battery life.

Our WebPADS design was never intended to limit its application to HTTP; mobilets that serve other protocols such as FTP and SMTP are possible development directions. WebPADS can support generic adaptive mobile applications to operate across different application transfer protocols, and to be used as the underlying platform to support adaptive service deployment over wireless environment.

Acknowledgments

This project is jointly supported by Hong Kong Polytechnic University research grant G-T 201 and the Department of Computing's research grant 4-Z040.

References

1. B. Krishnamurthy and J. Rexford, *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*, Addison-Wesley, 2001.
2. M. Liljeberg et al., "Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Approach," *Proc. 2nd Int'l Workshop Services in Distributed and Networked Environments*, IEEE CS Press, 1995, pp. 132–139.
3. B. Hausel and D. Lindquist, "WebExpress: A System for Optimizing Web Browsing in a Wireless Environment," *Proc. ACM Int'l Conf. Mobile Computing and Networking* (MobiCom 96), ACM Press, 1996, pp. 108–116.
4. S.N. Chuang et al., "Dynamic Service Reconfiguration for Wireless Web Access," *Proc. 12th Int'l World Wide Web Conf.*, ACM Press, 2003, pp. 58–67.
5. K.M. Chan and K.H. Shiu, "Web Metasearching for Mobile Environments Using METARIA," to be published in *Proc. 18th Int'l Conf. Advanced Information Networking and Applications*, IEEE CS Press, 2004.

Siu-Nam Chuang is a PhD student in the Department of Computing at Hong Kong Polytechnic University. His research interests include context-aware mobile computing, adaptive computing, and mobile middleware. Chuang received his MPhil in computer science from the Hong Kong Polytechnic University. Contact him at cssiunam@comp.polyu.edu.hk.

Alvin Chan is an assistant professor at Hong Kong Polytechnic University. His research interests include mobile computing, context-aware computing, and smart card applications. Chan received his PhD in computer science and engineering from the University of New South Wales. Contact him at cstschan@comp.polyu.edu.hk.

Jiannong Cao is an associate professor in the Department of Computing at Hong Kong Polytechnic University. His research interests include parallel and distributed computing, networking, mobile computing, fault tolerance, and distributed software architecture and tools. Cao received his BSc from Nanjing University, China, and his MSc and PhD from Washington State University, all in computer science. Contact him at csjcao@comp.polyu.edu.hk.

Ronnie Cheung is an assistant professor in the Department of Computing at Hong Kong Polytechnic University. He also serves as chair of the ACM's Hong Kong chapter and vice-chair of the Hong Kong Web Society. Cheung received his BS in mathematics and computing from the University of Hong Kong, and his MS in artificial intelligence from the University of Essex. Contact him at csronnie@comp.polyu.edu.hk.