AN ARCHITECTURE FOR ADAPTIVE MOBILE APPLICATIONS

Thomas Kunz[1]
Systems and Computer Engineering
Carleton University
tkunz@sce.carleton.ca

James P. Black
Computer Science
University of Waterloo
jpblack@uwaterloo.ca

ABSTRACT

Mobile applications execute in an environment characterized by scarce and dynamically varying resources. We believe that applications have to adapt dynamically and transparently to the amount of resources available at runtime. To achieve this goal, we use the conventional extension of the client-server model to a client-proxy-server model. The mobile devices execute the client, which provides the user interface and some part of the application logic. The proxy is a component of the application that executes in the wired network to support the client. As the user moves, the proxy may also move to remain on the communication path from the mobile device to a fixed correspondent host. Logically, the proxy hides the "mobile" client from the server, who thinks it communicates with a standard client (i.e., a client that executes on a powerful desktop directly connected to the wired network). The new contribution of our research lies in the division of labor between client and proxy. The application logic of the "standard client" is split dynamically between the mobile client and the proxy, using mobile code, to adapt to the dynamic wireless environment and to address the limitations of the portable device. Using mobile code allows us to experiment with different adaptation strategies and to explore resource tradeoffs in a unified framework. We are developing the components of a flexible and general-purpose runtime infrastructure to facilitate the rapid development and deployment of such adaptive mobile applications. We will evaluate our infrastructure by implementing a number of wireless applications and by building simulation tools to validate the scalability of our architecture when considering metropolitan and provincial cellular systems. The simulations will be driven by trace data that we are collecting in cooperation with a Canadian cellular service provider.

INTRODUCTION

The convergence of two technological developments has made mobile computing a reality [17]. In the last few years, developed countries spent large amounts of money to install and deploy wireless communication facilities. Originally aimed at telephone services (which still account for the majority of usage), the same infrastructure is increasingly used to transfer data. The second development is the continuing reduction in size of computer hardware, leading to portable computation devices such as laptops, palmtops, or functionally enhanced cell phones. Given current technology, a user can run a set of applications on a portable device and communicate over a variety of communication links, depending on his/her current location. For example, the user can access the wired corporate LAN at 10 Mbps or higher in the office. Roaming in the building, connectivity is provided by an indoor wireless LAN at 1-2 Mbps. Outdoors, connectivity is provided by cellular wireless-IP networks, providing bandwidths of a few tens of kbps. Furthermore, the sets of services available in each location will generally differ.

[1] The contact for this paper is Thomas Kunz, Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6, email: tkunz@sce.carleton.ca, phone: (613) 520-3573, fax: (613) 520-5727

Similar discrepancies will also persist in future wireless networks, such as the ones under study by the International Telecommunication Union [11] and the European Union's ACTS program [1]. Unlike second-generation cellular networks, future cellular systems will cover an area with a variety of non-homogeneous cells that may overlap. This allows the network operators to tune the system layout to subscriber density and subscribed services. Cells of different sizes will offer widely varying bandwidths: very high bandwidths with low error rates in pico-cells, very low bandwidths with higher error rates in macro-cells. Again, depending on the current location, the sets of available services might also differ. It is generally argued that applications should "adapt" to the current environment, for example by filtering and compressing data or by changing the functionality offered to the user, see [2,6,22,28,30]. Some researchers even argue that all future applications, not just the ones intended for execution on mobile devices, will have to be able to adapt to changing requirements and changing implementation environments on time scales from microseconds to years [13]. This paper describes the architecture we propose to facilitate the development of such adaptive mobile applications, based on mobile code.

The alternative to adaptive applications is multiple functionally identical or similar binaries, tuned for specific environments. This is an inferior solution, for a number of reasons. The user of a portable device has to install and maintain multiple applications, which is a drain on the limited storage capabilities typically found on those devices. It also potentially results in different user interfaces and causes high software development overheads when developing the "same" mobile application multiple times. Finally, it forces the user to identify the current execution conditions and select the "right" application.

MOBILE APPLICATIONS

To define a suitable architecture, we first identify categories of applications a mobile user is most likely to execute on his mobile device. Due to the existing limitations of portable devices (limited computational power, disk space, screen size, etc.), we claim that portable devices should not be considered general-purpose computers. Even though portable devices will become increasingly powerful, they will never match the computational power and facilities available on typical desktop machines. Similarly, while the wireless technology will improve, providing more and more bandwidth to the end user, wired network technology will advance as well, with the result that wireless networks will remain, in the near to medium future, orders of magnitudes slower. Therefore, mobile computing will always be characterized by a scarcity of resources, relatively speaking. In our opinion, an end-user will execute applications in one of the following six categories in such an environment:

- Standalone applications such as games or utilities;
- Personal productivity software (word processors, presentation software, calendars);
- Internet applications such as e-mail, WWW browsers, multi-user calendars, or telnet;
- Vertically integrated business applications (field installation and services, security);
- New "location-aware" applications: tour planners, interactive guides;
- Ad-hoc network and groupware applications.

The first category was originally of little interest to us, since these applications do not involve communication. However, the main idea underlying our architecture is to transparently support resource-constrained mobile devices by powerful proxy servers. We are therefore currently exploring how to generalize this idea to support standalone applications as well.

Applications in the second category will be used on multiple platforms: a user will have a version of his/her favorite word processor executing on a laptop as well as on the more powerful desktop in the office. This requires the exchange and synchronization of documents between the machines. Depending on the prevailing view of available network connectivity, two possible approaches are imaginable. Windows CE and MS Office exemplify a first solution. A user works on a document at either the laptop or the desktop, synchronizing multiple versions only infrequently and in a controlled environment. A second solution assumes that connectivity is more pervasive, allowing access to "authoritative" copies of a document on demand. This solution will require client-server applications to allow access to remote documents in the presence of highly variable communication links.

The Internet applications constitute a very interesting and challenging category. Mobile devices are often considered as the "on-ramp" to the Internet, see for example [10]. Consequently, a user will want to execute the client side of typical Internet applications on his portable device, communicating with servers in the existing Internet infrastructure. This is not as straightforward as it may seem at first glance. The Internet developed as a wired network, connecting powerful computers over relatively high-speed communication links. The assumptions underlying the design of many Internet clients reflect this view of the world. They are therefore not particularly well suited to a mobile environment. For example, the communication protocol of choice is TCP, which is known to behave poorly in the presence of wireless links with their corresponding high bit-error rates. Client applications typically assume that they have sufficient bandwidth, memory, and computational power at their disposal, which is equally questionable. Given the huge amount of money invested in the current infrastructure, it is unrealistic to expect that the whole Internet will change to accommodate mobile users overnight. In particular, servers deployed worldwide will not change in the near future. To facilitate access to the Internet, only the client side of the application can be adapted to function well in the dynamic and resource-constrained mobile environment. The architecture proposed below is intended for applications in this category.

Vertically integrated business applications are often structured as client-server applications. Furthermore, the backends (servers) have to support both existing wired desktops and wireless mobile devices. One example is a bank, where the back office has to support account managers in branch offices as well as mobile customer service representatives. Therefore, the clients executing on the portable devices face challenges similar to those faced by traditional Internet clients. They have to adapt to the limitations of the portable device in a dynamically changing execution environment. To facilitate the deployment of mobile applications, solutions should be transparent to the servers. Due to these similarities, we believe that the architecture proposed below applies equally well to this group of applications.

The location-aware applications exploit the fact that a user is mobile. Possible examples include travel guides, which might display the shortest path from a user's current location to the closest/cheapest/best Italian restaurant, or applications that allow a user to print a document on the closest color postscript laser printer. To the extent that these applications utilize the existing Internet (discovering and accessing nearby resources, for example), the architecture described below can be of value here as well.

Applications in the final category arise out of the mobility of a number of users, for example the meeting of a number of researchers or managers, each equipped with a portable device. Users might want to establish ad-hoc networks to exchange documents (the newest version of the transparencies for the invited talk) or to execute groupware applications to update a shared business plan. Since these applications will not, to a large extent, be limited by the need to interact with an existing infrastructure, the proposed architecture is probably not directly relevant to them. Similar to standalone applications, we are however exploring how to generalize our ideas to support ad-hoc network applications.

THE CLIENT-PROXY-SERVER MODEL

Most application categories discussed above comprise client-server applications. An application on a mobile device or desktop workstation provides some functionality to the

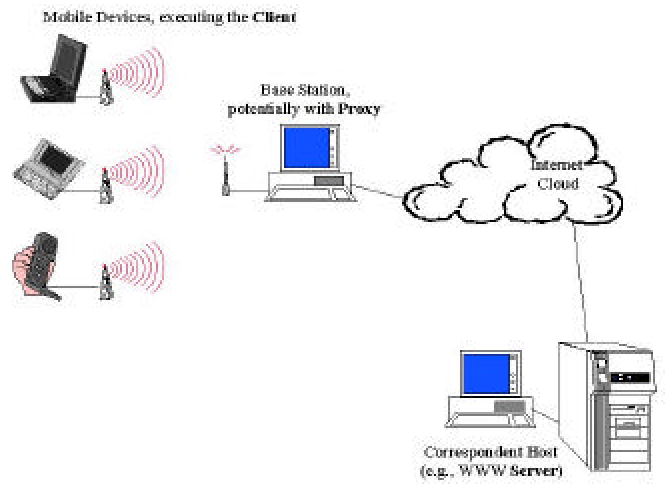
Mobile Devices, executing the Client
Base Station, potentially with Proxy
Internet Cloud
Correspondent Host (e.g., WWW Server)

Figure 1: The Client-Proxy-Server Model

end-user in conjunction with server(s) in the Internet. Examples are the WWW browsers that retrieve documents from servers around the world, or clients that connect to FTP servers to upload or download files. Our architecture is based on an extension of this traditional client-server model to a client-proxy-server model. Figure 1 shows the relevant components.

The mobile devices execute the client, which provides the user interface and some part of the application logic. In all wireless technologies, a mobile device communicates with the fixed network through a base station, which is the access point for all devices in a cell. Through some switching fabric, the base station provides connectivity with other hosts in the network, such as a WWW server or an SMTP server. The third component of our model is the proxy, a component of the application that executes in the wired network and supports the client. One possible location for the proxy is the base station. In general, any computer on the communication path between client and server can host the proxy. Logically, the proxy hides the "mobile" client from the server, who thinks it communicates with a standard client (i.e., a client that executes on a powerful desktop directly connected to the wired network). The application logic of the "standard client" is split between the mobile client and the proxy to adapt to the dynamic wireless environment and to address the limitations of the portable device. For example, the proxy could execute on a powerful workstation with large amounts of memory and disk space. This would make the proxy an ideal candidate to manage large caches or to perform computationally intensive tasks such as interpreting an MPEG video stream and turning it into a pixmap. Where bandwidth limitations over the wireless link are of major concern, the proxy could provide filtering and compression functions.

To avoid suboptimal communication paths, the proxy must migrate within the fixed network, following the user. "Migrating" the proxy can mean either that the proxy moves physically from machine to machine, or that a system of proxies exists, and only relevant state information is passed during a hand-over. The two approaches are complementary, and the best choice depends upon the details of a given situation. Other issues raised by user mobility are:

- How should the proxy migration decision be made (based on what information, using what algorithm)?
- Given that we expect a large number of mobile users, how does the approach scale? For example, what happens if a sizeable subset of users decides to head to the same shopping mall at noon,

checking stock options over the WWW while having lunch?

COMPARISON TO RELATED WORK

The idea of using a proxy in the wired network to support a mobile device is well known. In fact, most wireless WWW browsers, among others, use one or more proxies to support their operation in a low-bandwidth environment [5,7,8,16]. However, our approach is more concerned with adaptability, flexibility, and mobility than work published in the literature, as explained below.

ADAPTABILITY

Existing proposals often install a proxy that filters and/or compresses data for a specific application. This filter is either enabled or disabled. Examples are filters to turn color inline images into lower-resolution grayscale images or to convert postscript into ASCII text, or e-mail readers that provide the "subject" line of a mail message only, avoiding the transmission of the main message body over a slow link. In the scenario of the introduction, the environment changes dynamically, depending on the user's behavior and that of others. We therefore foresee the need to make the proxies more dynamic, to adapt more closely to changes in the environment. Examples are changing the resolution and/or color of an inline image only when necessary, and changing the resolution incrementally. Another example could be to allow an e-mail reader to read the body of small e-mail messages but to avoid downloading large messages. To enable this more fine-grained adaptation, a mechanism is needed that provides the mobile application with information about the environment, including device information, information about the wireless link, infrastructure information like reachable servers or services, and location-related information.

Adaptation decisions need to take information about multiple resources into consideration. For example, high available bandwidth might favor a shift from the proxy to the mobile. However, the mobile might be running low on power, arguing for a shift from the mobile to the proxy. Similarly, certain adaptation options might not be feasible in environments that lack the appropriate infrastructure. Previous research tended to focus on one resource only: [24,26] focus on power consumption, [21] on location information, [9] on local services, and [4,30] on the network characteristics.

A completely different approach would be to design the division of labor between mobile client and proxy for the worst case only, which would allow a user to get consistent functionality across all environments. We consider such a solution less than desirable, for at least two reasons. First, whenever possible, a user should experience the best service achievable. Normalizing to the lowest common denominator often unnecessarily deprives the user of full service and functionality. Second, a proxy server will, in general, support many mobile users. To the extent that the individual portable devices can contribute to better client functionality, they should be encouraged to do so. Otherwise, even a powerful proxy server might be overloaded, resulting in poor performance for all applications.

FLEXIBILITY

Most existing proxy solutions operate at the protocol level. They intercept messages and reduce the data sent over the wireless link by filtering less important data and utilizing compression algorithms (see for example [30]). As discussed in [5], while data compression and filtering improve the perceived performance of web browsers, they are not sufficient. Support for new types of operations, such as browsing while disconnected and/or asynchronous browsing, is needed in a mobile environment. The research in [5,16] is based on a static client-proxy-server architecture, where the division of labor between client and proxy is determined and fixed at design time. Both papers, however, emphasize that a more flexible, dynamic, adaptation based on mobile code should be explored in future.

Our approach is aimed at more general proxy solutions. As mentioned above, we envision mobile applications where the proxy can overcome some of the limitations of the

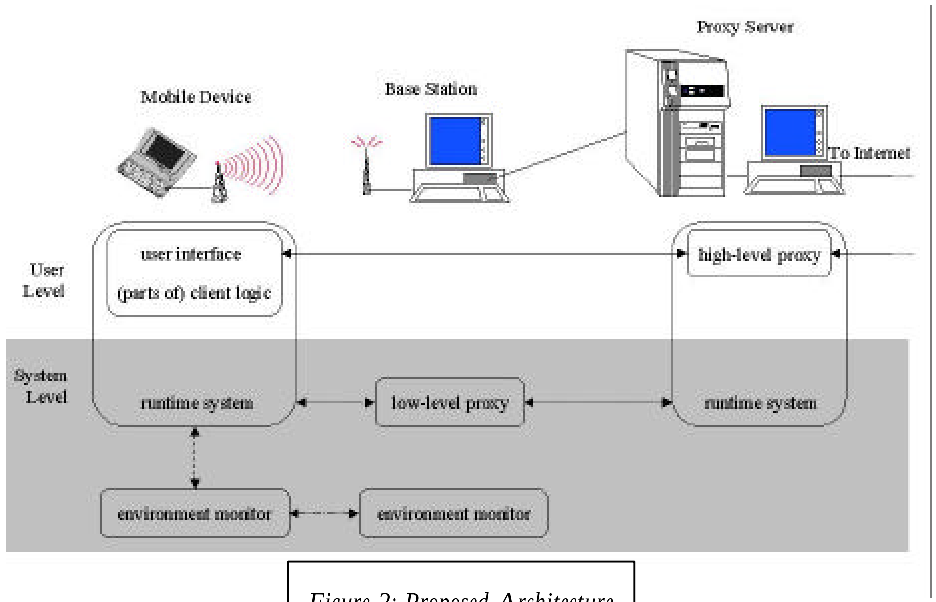

Figure 2: Proposed Architecture

portable device by maintaining a cache or performing CPU-intensive interpretation of the data stream. Offloading some of the application logic to a proxy comes at a cost. We therefore want to shift parts of the client logic only when necessary (to conserve battery power or reduce network bandwidth, for example). This again argues for a dynamic adaptive proxy design, with the added complexity of moving not only data, but also code.

MOBILITY

A third major difference from previous work is that we plan to address issues of mobility. Nearly without exception, existing solutions address the situation where a mobile host interacts with a fixed host over the same wireless link forever. In fact, [30] claims that the issue of proxy migration is beyond the current state-of-the-art. Given that “migrating” the proxy does not necessarily mean that we physically migrate the proxy, we disagree with this statement. We will investigate how to create a proxy infrastructure, perhaps connected with or related to the concept of Intelligent Networks [27] in telephony, which was devised for fast deployment of new services in a network. “Migrating” the proxy would then take the form of a handoff between two proxies. There are three salient issues to be addressed here.

- What information should be collected (and how to collect it) to make “migration” decisions?
- What are good decision algorithms, based on this information?
- How does the solution scale to a metropolitan or provincial coverage area?

ARCHITECTURE

To achieve truly adaptive applications, we need to design and implement a number of components. Figure 2 gives an overview of the relevant pieces and how they interact. In this architecture, we distinguish two proxies, a *high-level* proxy and a *low-level* proxy, similar to [30]. They play distinctive roles and require different mechanisms for their implementation.

A central piece of infrastructure is the environment monitor. The client application needs to be informed of changes in the environment. This can be achieved by a daemon process on the mobile device, monitoring relevant environment parameters and notifying the application through registered call-back functions. To avoid interruptions of the client logic, these call-back functions could execute in a separate thread, for example, communicating with the rest of the mobile client through shared state. An application must register with this daemon, inform it of environment parameters it is interested in and the conditions under which it wants to be notified, and provide appropriate call-back functions. To obtain reliable information about the wireless link, the client-side daemon might interact with a similar daemon on the other end of the wireless link.

The mobile device will execute the user interface and parts of the client application logic. Other parts of the client logic execute on a dedicated proxy server, which is a powerful machine in the fixed network. Ideally, the proxy server should be close to the base station. The high-level proxy communicates with the server, transparently hiding the fact that the server communicates with a mobile client. Since the high-level proxy will execute parts of the application code, it will exist at the user level.

The application logic is divided between the mobile client and the high-level proxy. This division of labor changes over time, depending on the current environment. In previous work, we designed and implemented a number of applications that registered with the monitor and made decisions about the most appropriate adaptation, based on the feedback received from the monitor, see for example [20,29]. To facilitate the development of adaptive mobile applications, we plan to factor the partition algorithms out into the runtime system. Each client application will be designed concentrating on the functional aspect only. To enable dynamic partitioning, the application may register certain information with the runtime system. It is up to the runtime system to connect with the monitor and to combine the information registered by the application with the feedback received from the monitor to make partitioning decisions.

We have started implementing this architecture as follows. The mobile device and the proxy server both execute a Java virtual machine. The client application consists of a number of objects, communicating via method invocation. Upon creation, each object may register with the runtime system. The registrations are of the form “execute on mobile if bandwidth greater than X” or “execute in same location as object O.” A default policy handles objects that did not explicitly register and provide specific information. For example, a default policy links object size to bandwidth and/or available memory. Objects exceeding a threshold size execute on the proxy server, otherwise they execute on the mobile host. The runtime system monitors the relevant parameters and initiates object migration if necessary. Object migration is achieved by a mobile code technology based on Java (Objectspace Voyager [24]).

A low-level proxy supports the communication between client and high-level proxy. One example of such a low-level proxy is SNOOP [3]. The low-level proxy operates at the data, network, and/or transport layers. Protocols at these layers are typically provided as part of the operating system protocol stack, so for maximal efficiency we expect the low-level proxy to become part of the operating system. The low-level proxy supports communication over the wireless link, so we envision that the low-level proxy will execute on a host that directly connects to the wireless link. There are only two choices: the mobile device and the base station. Given the restrictions of the mobile device, the logical place for the low-level proxy is the base station. However, a low-level proxy might also be split between the mobile device and base station, to provide symmetric support for communication to and from the mobile device.

Most published low-level proxies work under the assumption that the low-level proxy is transparent to higher-level communication. Another useful extension of the proxy approach is to extend the capabilities of the higher-level protocols. In previous work [15,19], we identified services such as prioritizing competing TCP connections or keeping TCP connections alive in the presence of spurious disconnections. In this case, the existence of the low-level proxy cannot be kept transparent to the client. In fact, the client needs an interface to request such enhanced services, and to provide the proxy with necessary information. As far as possible, this functionality should be hidden in the runtime system, invisible to the application logic implemented by the programmer.

EXAMPLES

To give an indication of how the architecture will support adaptive mobile applications, this section briefly describes designs for a number of potential applications.

MPEG FILTER

We are completing the implementation of a transparent MPEG filter that operates as part of the low-level proxy [18]. It selectively drops frames of various types depending on the current network conditions. This is achieved transparently to the application through the use of a second, lower-level filter [14] that adjusts the TCP protocol headers to mask the removal of data. This preserves the end-to-end semantics of the TCP stream, even though the two endpoints have divergent views about the amount of data exchanged.

WWW BROWSER

A possible design for a WWW browser uses the high-level proxy as a data filter. The client registers with the monitor, asking to be informed of changes in the bandwidth. In a high-bandwidth environment (for example, when the user is working in the office), the high-level proxy forwards HTTP requests to the servers and forwards the replies to the client. In addition, the high-level proxy can maintain a cache of documents, potentially sharing this cache with other clients executing in the same cell. For medium-bandwidth environments, the high-level proxy filters the incoming inline images to reduce the resolution. In a low-bandwidth environment, the high-level proxy filters inline images to reduce both the resolution and the color. In this example, client and proxy communicate with the standard HTTP protocol.

This first design follows the customary use of proxies to filter and compress the data stream, potentially making the proxy somewhat more adaptive based on feedback about the available bandwidth received from the monitor. A somewhat different design, emphasizing code mobility, is as follows. Based on information about the mobile device, routines to process and format the HTTP replies are assigned to either the portable device or the high-level proxy. The high-level proxy would send pixmaps representing the web page to the client, who displays them and returns information about user actions. Full-color high-resolution pixmaps will increase the bandwidth requirement, but reduce the computational load on the client, saving energy. Grayscale and/or low-resolution pixmaps can help to reduce both energy and bandwidth consumption. In either design, the proxy communicates with the server following the HTTP protocol, hiding the mobile client and the current division of labor between client and proxy from the other servers.

MUSEUM GUIDE

A different application is an interactive museum guide. A visitor wanders through a museum with a palmtop in his/her hands. Information about each exhibit is displayed as the visitor approaches an object. The application also provides the visitor with the option to inquire about a specific exhibit and to display the best path from his/her current location to that object.

To enable this application, the monitor must provide location information. This could be achieved either directly, with a system such as GPS, or indirectly, by deducing the current location from the base station with which the device communicates. Computing the shortest path is potentially rather complex and is therefore done at the high-level proxy. This proxy also provides a cache for previous information and the query routines to fetch information from the central database, based on the current location. The portable device executes the interface component. The available bandwidth will vary, depending on the number of people crowding in front of the more popular exhibits, and whether the object is indoors or outdoors. The application can adapt to these changes by offloading the graphical processing to the proxy, similar to the WWW browser example. In this case, the backend (the database that provides the information about exhibits) can serve other applications as well, for example to generate dynamic web pages.

Bandwidth	All folders on Mobile	All folders on Proxy	Adaptive scheme
10 Mbps	369	529	486
1 Mbps	1,346	577	532
100 kbps	2,018	637	669
64 kbps	-	2,583	2,190

Table 1: Elapsed Times (seconds) for variants of the mail application

MAIL READER

A mobile mail reader can use the monitor to inquire about locally available SMTP servers. It establishes connections to the closest SMTP server (potentially based on security and/or cost considerations) to send e-mail messages composed by the user. To read e-mail, the client must connect to the "home" message store. Typically, a user will have many different folders with many messages in each. Copies of these folders are downloaded to the high-level proxy server. Depending on the network conditions and the available space (memory, disk) at the mobile device, copies of the smaller folders can be downloaded to the mobile. The application will dynamically determine whether a mail folder exists locally or at the proxy server, processing requests for listing all messages or displaying certain messages at the appropriate location.

A partial version of such a mail reader is described in [20]. We experimented with different policies for placing the folders on either the mobile host or the proxy server. To evaluate the performance, we measured the elapsed time to execute scripts representing user sessions for the various configurations over different bandwidths. Table 1 lists the results.

We found that the dynamic adaptation scheme, while not always resulting in the lowest possible response times, performs relatively well across all bandwidths. Furthermore, the two "static" designs, creating all folders at either the mobile device or the proxy independent of the current network condition, result in poor performance in some environments. For applications that encounter the whole range of bandwidths (and other characteristics of a dynamically changing environment), we need adaptive application designs. Otherwise we end up with applications that either result in extremely poor performance some of the time, or a user will need to install multiple binaries for the same task. As this example also demonstrates, the adaptation can be transparent to the user and application developer, assuming that a mobile code system with location-transparent method invocations is being used.

SUMMARY AND CONCLUSIONS

Mobile computing is a relatively new field. While the challenges arising from mobility and the limitations of the portable devices are relatively well understood, there is no consensus yet as to what should be done to address these challenges. A comprehensive solution has to address many different aspects, such as the issue of dynamically changing bandwidth, the power, computational, and other limitations of the portable devices, or the varying availability of services in different environments. This paper presents our architecture for such adaptive mobile applications. We motivated the architecture by classifying likely mobile applications and identified common properties. The architecture intends to be more general than previous work with respect to adaptability, flexibility, and user mobility. We developed various pieces of the overall architecture and collected some preliminary experience with adaptive mobile applications. The results, reported in [15,18,19,20,23,29], are encouraging. Work is currently under way to implement the missing components of our architecture. Once completed, we will explore the following questions: What are good indicators of relative resource scarcity and how do we collect them at runtime? What adaptation policies are appropriate for different resources (bandwidth, power, or memory)? Can we tradeoff one resource for another? How transparent can application adaptation be made to end-users? How transparent can it be made to application developers? We are also cooperating with a

Canadian cellular service provider to evaluate our ideas and explore the scalability of our architecture for province-wide cellular systems.

BIOGRAPHIES

Thomas Kunz received the Dr. Ing. degree from the Technical University of Darmstadt, Federal Republic of Germany, in May 1994. From 1994-1997 he worked as assistant professor in the Department of Computer Science at the University of Waterloo, Ontario, Canada. During this time, he was a member of the Shoshin research group, participating in the development of a distributed debugger. He also started work on client-server applications over wireless links. In 1997, he joined the Department of Systems and Computer Engineering at Carleton University as assistant professor. Starting in the summer of 1998, he heads a research project that explores how to adapt mobile applications transparently to scarce and dynamically varying execution environment resources. His other research interests include load balancing in distributed systems, distributed debugging, management of distributed applications and systems, and wireless communication systems and protocols. He can be reached at: Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario K1S 5B6. His e-mail address is tkunz@sce.carleton.ca.

J.P. Black received his PH.D. in Computer Science from the University of Waterloo in 1982, having previously studied at the University of Calgary and the Institute National Polytechnique de Grenoble, France. He has been on the faculty of the University since 1984. He is currently Associate Provost, Information Systems and Technology, and is an Associate Professor in the Department of Computer Science. As Associate Provost, he is responsible for the administration and long term planning of all information systems and information technology at UW. His current research interests include the management of distributed applications and systems, distributed debugging and the visualization of complex executions, and mobile and wireless computing. He was involved in the design and implementation of the Newcastle Connection, and early distributed extension to Unix at the Computing Laboratory, University of Newcastle on Tyne. More recently, he has been involved in the design and development of Poet, a suite of software for visualizing and debugging distributed and concurrent applications. His research forms part of the activities of the Shoshin research group at Waterloo.

REFERENCES

1. ACTS Mobility, Personal & Wireless Communications Domain, *Evolving the UMTS Vision*, December 1997, http://www.infowin.org/ACTS/IENM/CONCERTATION/MOBILITY/.
2. B. R. Badrinath, A. Acharya, and T. Imielinski, *Structuring Distributed Algorithms for Mobile Hosts*, Proceedings of the 14th International Conference on Distributed Computing Systems, Poznan, Poland, June 1994, pages 21-28.
3. H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, *Improving TCP/IP Performance over Wireless Networks*, Proceedings of the First Annual International Conference on Mobile Computing and Communications, Berkeley, USA, November 1995, pages 2-11.
4. J. Bolliger and T. Gross. *A framework-based approach to the development of network-aware applications*. IEEE Transactions on Software Engineering, 24(5):376-390, May 1998.
5. H. Chang, C. Tait, N. Cohen, M. Shapiro, S. Mastrianni, R. Floyd, B. Housel, and D. Lindquist, *Web Browsing in a Wireless Environment: Disconnected and Asynchronous Operation in ARTour Web Express*, Proc. 3rd Ann. ACM/IEEE Conf. on Mobile Computing and Networking, Budapest, Hungary, September 1997, pages 260-269.
6. G. H. Forman and J. Zahorjan, *The Challenges of Mobile Computing*, University of Washington, Department of Computer Science, Tech. Rep. #93-11-03, March 1994.

7. S. Gessler and A. Kotulla, *PDAs as mobile WWW browsers*, Proc. of the Second World Wide Web Conference: Mosaic and the Web, Chicago, Illinois, USA, October 1994, http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/gessler/www_pda.html.

8. R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas, *Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing*, IEEE Personal Communications, December 1998, pages 8-17.

9. T. D. Hodes, R. H. Katz, E. Servan-Schreiber, and L. Rowe, *Composable Ad-hoc Mobile Services for Universal Interaction*, Proc. 3rd Ann. ACM/IEEE Conf. on Mobile Computing and Networking, Budapest, Hungary, September 1997, pages 1-12.

10. Industry Canada website http://spectrum.ic.gc.ca/.

11. International Telecommunication Union, *IMT-2000 Website*, http://www.itu.int/imt/.

12. Internet Engineering Task Force website: http://www.ietf.org/.

13. K. Kavi, J. C. Browne, and A. Tripathi, *Computer Systems Research: The Pressure is on*, IEEE Computer, January 1999, pages 30-39.

14. D. Kidston, *Transparent Communication Management in Wireless Networks*, Master's Thesis, Dept. of Computer Science, University of Waterloo, October 1998.

15. D. Kidston, J.P. Black, T. Kunz, M.E. Nidd, M. Lioy, B. Elphick, and M. Ostrowski. *Comma: A communication manager for mobile applications*. Proceedings of the 10th Annual Int. Conf. On Wireless Communications, Calgary, Alberta, Canada, pages 103-116, July 1998.

16. M. T. Le, S. Seshan, F. Burghardt, et al., *Software Architecture of the InfoPad System*, Proceedings of the Mobidata Workshop on Mobile and Wireless Information Systems, Rutgers, NY, USA, 1994.

17. Y. Li and V. C. M. Leung, *Supporting Personal Mobility for Nomadic Computing over the Internet*, Mobile Computing and Communications Review, 1(1), Apr. 1997, pages 22-31.

18. F. G. Lin, *An Adaptive MPEG Filter*, Master's Thesis, Department of Computer Science, University of Waterloo, in progress.

19. M. Lioy, *Providing TCP-level Services to Mobile Computers in a Wireless Environment*, Master's Thesis, Department of Computer Science, University of Waterloo, September 1997.

20. H.-Y. Lo, *M-Mail: A Case Study of Dynamic Application Partitioning in Mobile Computing*, Master's Thesis, Department of Computer Science, University of Waterloo, May 1997.

21. H. Maass, *Location-Aware Mobile Applications based on Directory Services*, Proc. 3rd Ann. ACM/IEEE Conf. on Mobile Computing and Networking, Budapest, Hungary, September 1997, pages 23-33.

22. E. G. Manning, S.Khan, R. Lea, G. C. Shoja, and M. M. J. Zastre, *Metaspaces and Mobile Computing: Promises and Challenges*, Lecture Notes in Computer Science, Volume 1274, Springer Verlag, 1997, pages 350-362.

23. M.E. Nidd, T. Kunz, and J.P. Black. *Wireless applications and API design*. Proceedings of the 4th Int. IFIP Workshop on QoS, Paris, France, pages 83-85, March 1996.

24. Objectspace Voyager, http://www.objectspace.com/developers/voyager/white/index.html

25. M. Othman and S. Hailes, *Power Conservation Strategy for Mobile Computers Using Load Sharing*, Mobile Computing and Communications Review, 2(1), January 1998, pages 44-51.

26. A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, *Saving Portable Computer Battery Power through Remote Process Execution*, Mobile Computing and Communications Review, 2(1), January 1998, pages 19-26.

27. Telecom Finland, *Intelligent Networks and Services*, http://www.tfi.net/rd/network.html.

28. G. Welling and B. R. Badrinath, *A Framework for Environment Aware Mobile Applications*, 17th International Conference on Distributed Computing Systems,

Baltimore, Maryland, USA, May 1997, pages 384-391.

29. T. J. Whalen, *Design Issues for an Adaptive Mobile Group Editor*, Master's Thesis, Department of Computer Science, University of Waterloo, September 1997.

30. B. Zenel and D. Duchamp, *A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment*, Proc. 3rd Ann. ACM/IEEE Conf. on Mobile Computing and Networking, Budapest, Hungary, September 1997, pages 238-259.