An Architecture to Support Adaptive Mobile Applications

Xiaozhen Cao   Rick Bunt
Dept. of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan
Canada S7N 5A9
E-mail: {xic509, bunt}@cs.usask.ca
TR*Labs*, 111-116 Research Drive
Saskatoon, Saskatchewan
Canada S7N 3R3

Abstract

To serve heterogeneous mobile devices operating on a network, a new architecture is proposed to support adaptive mobile applications. In this architecture, middleware is being developed to act as an intermediary between adaptive mobile applications and the network, with the primary goal being to provide adaptation services. A prototype is being built using Java to support a range of mobile applications, including a Web server, a video server, and an FTP-like application, with three different client devices: a desktop computer with a 1024x768 pixels monitor, a laptop computer, and a pocket PC with a 240x320 pixels screen. The application retrieves resource information from the middleware, applies adaptation strategies and sends adaptation data to heterogeneous client devices over different network conditions ranging from 100 Mbps wired Ethernet LAN (Local Area Network) to 11 Mbps wireless Ethernet LAN (IEEE 802.11b) to 20 kbps CDPD (Cellular Digital Packet Data) WAN service. Some lessons have been learned from this prototype.

1 Introduction

The demand for *mobile computing* is increasing dramatically, fuelled by the rapid development of technology in mobile devices and information access. This new paradigm enables users to access the information and services they require from anywhere at any time, and inexpensive devices make mobile computing both feasible and practical. To provide good end-to-end service without worrying about the hardware configurations or the communication variations, the network should have adaptation ability. Since the applications are the end points of the network, adaptation solutions are most appropriately applied in the application layer [2].

The increasing demands of mobile computing have stimulated the development of the technology. The main challenges include improving network connectivity (e.g., high speed wireless technologies), modifications to network protocols (e.g., mobile IP [11]), and improving capabilities and portability of mobile devices. The ability of wireless network connections to access the Internet makes it possible for small mobile devices to be a part of the Internet. Increased wireless connection bandwidth can support more applications that can take advantage, not only of a network connection, but also of multimedia data.

These improved technologies lead to a significant complication: heterogeneity. Various mobile devices have different capabilities and connect to the Internet through a number of communication methods. Table 1 shows the current capabilities of some of these mobile devices. These vary in quite a few aspects, from memory size to processing speed to battery life. Popular mobile devices include BlackBerry produced

Table 1: Examples of Mobile Devices

Attributes	**Notebooks**	**Pocket PC**	**Handheld PC**	**Cellular Phones**
An Example	Sony PCG-GR170K	iPaq 3670	Stylistic 3500R	Motorola T193
CPU	PIII 1GHz	StronARM 206MHz	Celeron 500MHz	None
RAM	256/512MB+	64MB+	128MB+	-
Screen Type (max)	SXGA+ TFT	Touch Screen	Reflective SVGA TFT	-
Screen Size (max)	15"	4"	10.4"	1.6"
Disk Capacity	20GB+	2GB	15GB	None
Battery Life	2-3 hrs	10 hrs	-	14 hrs

by RIM and Palm produced by 3Com. This research focus on one type of pocket PC – iPaq, produced by Compaq. Table 2 shows the representative bandwidth for some common communication technologies, and shows that there are huge variations in data rates, ranging from less than 9.6 kbps to more than 155 Mbps.

Table 2: Representative Communication Bandwidths

Communication Technology	Bandwidth
Cellular telephone	9.6 kbps
Modem	9.6 - 57.6 kbps
Infrared	19.2 kbps - 4 Mbps
CDPD	19.2 kbps
ISDN	128 kbps
Radio	300 kbps - 10 Mbps
GRPS	40 - 115 kbps
CDMA and WCDMA	144 kbps - 2 Mbps
Ethernet	10-100 Mbps
ATM	155 Mbps

Under this scenario, an adaptation service is very important to mobile users. To build an effective and efficient adaptive application requires awareness of both client capabilities and changes to network conditions, and the ability to adapt the behaviour of the application as circumstances change. But the present support for adaptation is limited. Research has been done in applying the adaptive capability onto a specific class of applications [1, 3, 4], and research has been done on providing system support without modification of existing individual applications [7, 6]. The former approach, referred to as *ad hoc* adaptation, requires no support from the operating system as all of the adaptation is the responsibility of the individual application. It lacks central control over potentially incompatible resource demands of different applications to enforce limits on resource usage. It also makes applications more difficult to implement and to upgrade. The latter approach, referred to as *application-transparent* adaptation [10], puts all the adaptation responsibility on the system. The drawback here is that there may be situations in which the adaptation performed by the system is inadequate or even counter-productive.

To overcome the drawbacks of those two approaches, researchers have tried to find a balance point in between. An example is Odyssey [9], which is designed to gain control from both the system and the applications. But these studies considered dynamic resources only, such as network conditions. The previous work has make a good start in improving the adaptivity to the network conditions, but lacks consideration of variation of the capability of the mobile devices such as screen size and screen type. Because of the increasing heterogeneity of the mobile devices, taking these variables into account is important.

The previous research is reviewed in the next section. A new system model is proposed in Section 3 to support heterogeneous end devices under different network connection conditions. Middleware is designed to serve as an intermediary between the adaptive mobile applications and the network, with the primary goal being to provide adaptation services. The resource information that is required by diverse adaptive mobile applications is divided into static information and dynamic information. The objective is that the resource information would constitute a service provided by the system infrastructure instead of manipulated by each application itself. A prototype of this model, built in Java, is used to study how to monitor resources, how to organize/use the resource information, and how to take advantage of various application specific adaptation strategies. The main design decisions are discussed in Section 4. Section 5 shows the test environment for this example system and some lessons learned. Finally, Section 6 draws some conclusions of this research and proposes some areas for further study.

2 Related Research

Building an adaptive mobile computing system is not a new idea. In 1995, research was conducted on sketching such a system blueprint. Two prominent groups were Satyanarayanan's mobile computing research group at CMU (Carnegie Mellon University) and the PARCTAB group at Xerox PARC.

Odyssey [9], one of the most important adaptive systems built by Satyanarayanan's group, focused on resource management for multiple applications running concurrently on the same client machine. Odyssey uses an application-aware approach. By supporting a collaborative partnership between applications and the system, this approach permits individual applications to determine how best to adapt, but preserves the ability of the system to monitor resources and to enforce allocation decisions. It was designed to run in wireless environments characterized by changing and frequently limited bandwidth. A new property, *fidelity*, was defined to quantify the notion of quality. Three applications, a video player, a Web browser, and a speech recognizer, were implemented as testbeds. Odyssey is a well-designed scalable adaptive system that allows other applications to attach to the system easily. A set of qualitative lessons has been learned and the experiment has given us a basic view of this research.

Choi's [3] work examined how the client computing resources can be used by a Web server to adapt the requested content that best suits the client's machine and thereby to improve the response time. Four types of resource information are considered in this research: type of processor, type of Internet connection, percentage of processor load and percentage of memory load. The significance of each type was studied. The concepts of *pull* and *push* in system design are also defined. In the push approach, the information accompanies each request from the client to a server. In the pull approach, information is requested by the server as needed.

Badrinath [2] summarized the results of several projects in this area, proposed a conceptual framework and extracted some important general lessons learned about adapting data flows over difficult network conditions.

Other interesting work around adaptive mobile applications includes McIlhagga's [8] design methodology for adaptive applications, Lara's [5] Puppeteer system, which proposed the component-based adaptation on remotely using Microsoft PowerPoint and Internet Explorer 5, and Acharya's [1] language extension of Java that supports resource-aware mobile programs.

Since adaptation is based on information related to change, awareness of change is crucial to building successful adaptive applications. Generally speaking, awareness can be categorized as resource awareness, context awareness, or user awareness. Our project focuses on resource awareness, although the other two types of awareness are equally important to a adaptive mobile application.

Resource awareness supports application adaptations triggered by factors such as the presence of accessible devices and network capability. There have been a number of adaptive application systems, such as Odyssey [9], which incorporate resource awareness into their strategies. The emphasis has not been on what resources need to be monitored and how to monitor them, but rather on who is in charge of monitoring the resources. Choi's research [3] addressed resource awareness, but only a limited set of resources. Acharya's work [1] focuses on designing a language, Sumatra, an extension of Java, to support resource-aware mobile programs. It provides interfaces to adaptive applications rather than studies of resource awareness itself that will be emphasized in this research.

Since different applications have different functional and/or non-functional requirements such as performance or availability, the resources taken into account in this project are quite different from application to application. How we take advantage of the adaptation strategies will depend on how well we can manage the information pertaining to those resources.

3 The Architecture

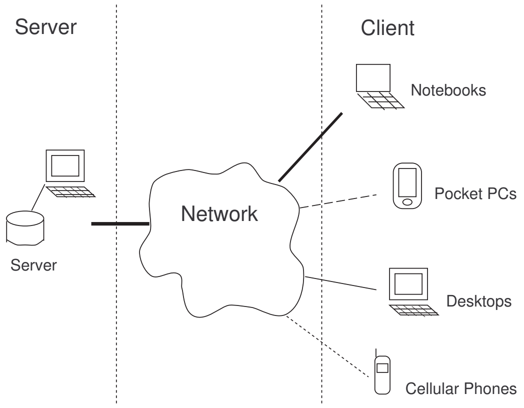

Figure 1: The System Model

Figure 1 shows the mobile computing environment under consideration. There are diverse client devices connected via the network to servers containing information and services required by the mobile users. It is possible that one or more hops exist in the network and one or more subnetworks exist in between. The thickness of the lines denotes the bandwidths in use. A dashed line represents a wireless connection and a solid line represents a wired connection.

3.1 The System Architecture

The proposed architecture to support adaptive mobile applications is an extension of a standard network system architecture. Figure 2 shows the addition of middleware to the application layer.

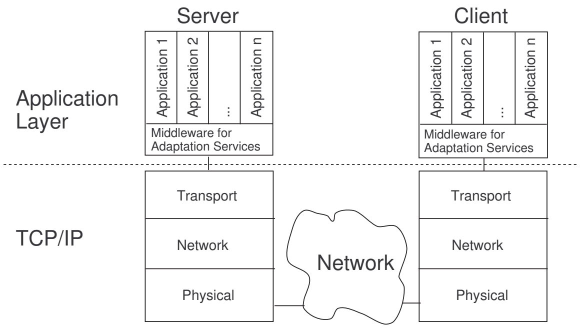

Figure 2: The System Architecture

Some previous designs achieved adaptability by modifying client applications such as Web browsers or media players. This approach doesn't scale well when there can be many different implementations of a particular application. For a single video server, for instance, a user may prefer to use Microsoft Media Player, Real Player or a user-homemade player. Even for players from the same vendor, different versions will be used on different mobile devices. For example, PocketDivx (Windows CE version) will be used only on pocket PCs, whereas Playa (Windows 9x/2000 version) will be used only on other Windows systems. An architecture that requires modifications of client applications will increase the complexity of the task. On the other hand, modification of the server is required only once, and the server is usually configured and maintained by professional system administrators rather than end users. The middleware approach makes it possible to minimize changes to application code on clients.

This architecture is designed to support heterogeneous end devices under different network connection conditions. Middleware serves as an intermediary between the adaptive mobile applications and the network, with the primary goal being to provide adaptation services. Therefore, the resource information becomes a service provided by the system infrastructure instead of manipulated by each application itself. Each application only need to specify to the middleware its own *interest list*, a list of the resources that it is interested in.

This design overcomes the drawbacks of the *application-transparent* approach. For example, the adaptation performed by the system might be insufficient or even counter-productive to a new adaptive mobile application. While the system supports multiple applications, under a particular network condition, a Web server doesn't have to adapt, but a Video server might want to reduce the frame rate to provide acceptable quality. While facing different mobile devices, the same server application also might adapt in different ways, for example, adapting colour resolution to a laptop but adapting screen size to a Pocket PC. Therefore, migrating the adaptation to applications can provide better service to clients.

This design eliminates the increased complexity of upgrading individual applications using an *ad hoc* approach, since the common services are provided by the middleware. To cooperate with the middleware, the server applications can be upgraded by implementing the interfaces described in Section 3.3, and applying the corresponding adaptation strategies based on the resource information from the middleware. The design of the middleware also makes it possible to leave the client applications unmodified.

The functions of the middleware at the server include:

- Communicating with the client and retrieving resource information about the client device.
- Monitoring the dynamically changing resources.
- Processing the resource information.
- Providing resource information to adaptive mobile applications.

The functions of the middleware at the client include:

- Collecting resource information.
- Providing resource information.
- Assisting in generating dynamic resource information reports.

3.2 Resource Information

The main goal of this architecture is to take advantage of the resource information on the client capabilities and the network condition.

The information that the middleware component retrieves can be categorized as static resource information or dynamic resource information. Table 3 shows a list of the resource types that we are considering in this prototype. The static information represents the static characteristics of the client device, including screen type, screen size, CPU capacity, disk capacity and bandwidth, etc. This type of information is fixed during one connection. The bandwidth may change when the user changes the connection method, for instance, from CDPD wireless service to wired Ethernet LAN, but generally, this switch needs to be disconnected and re-connected. As a result, the re-connection will be considered as another connection in a traditional Internet environment. All the static resource information is maintained by the middleware component on the client device. Each application has its own "interest list" of those static resources that it is interested in. Only information on the interest list will be retrieved and provided to the corresponding application. The dynamic information represents the dynamic characteristics while transferring data, including the transfer rate, CPU load, RAM load, etc. The dynamic information will be monitored based on the interest list provided and on certain sampling rates by each application as well.

The static information needs to be transferred only once for each connection, while updated dynamic information needs to be sent to the adaptive application by the middleware dynamically.

3.3 Interfaces

This architecture is designed to support general adaptive mobile applications. A set of interfaces is required by the applications to retrieve client resource information or network status informa-

Table 3: List of Resource Information in Consideration

Resource	Type	Description
Screen Size	Static	Display size in pixels
Screen Type	Static	Colour or BlackWhite
Bandwidth	Static	Connection capacity
CPU Capacity	Static	CPU capacity
Disk Capacity	Static	Hard Drive capacity
RAM	Static	RAM capacity
Transfer Rate	Dynamic	Instantaneous or effective bandwidth in bps
CPU Load	Dynamic	Instantaneous CPU load
Memory Load	Dynamic	Instantaneous Memory load

Table 4: Interfaces

Interface	Description
public AAHandler(String, Vector, AdaptiveServer)	Constructor with host name, resource list and pointer to the application as parameters
protected void setInterestedResourceList(Vector)	Reset the interested resource list
protected void setTimeInterval(int)	Set the time interval for monitoring dynamic info.
protected Hashtable getStaticInfo()	Retrieve static resource info.
protected void retrieveDynamicInfo()	Retrieve dynamic resource info.

tion of interest from the middleware. Because the implementation of the middleware is language dependent, the interfaces will be language dependent as well. Table 4 shows an example of the interfaces of the middleware in Java. These interfaces are used in the prototype of the middleware which will be described in Section 4.

To use the interfaces defined by the middleware, the application should implement the Interface of AdaptiveServer. Therefore a control pointer of a server application will be passed to the middleware without the knowledge of each individual application. The other four interfaces in this architecture include one to set the resource list, one to set the sampling rate of the dynamic information, one to get static information and one to turn on the monitoring of dynamic information.

3.4 Interaction Flow

Figure 3 shows the interactions between the server and the client in this resource-awareness mobile system model. Interaction progresses in the following way:

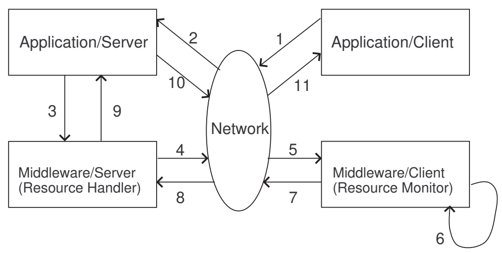

Figure 3: Interactions Between Server and Client

- (1, 2) The client application sends a request to the server application.
- (3) The server application sends the resource information request to the middleware at the server side, which is named the Resource Handler.
- (4, 5) The Resource Handler communicates with the Resource Monitor, the middleware at the client side.
- (6) The Resource Monitor extracts resource information of the client.
- (7, 8) The Resource Monitor sends this re-

source information to the Resource Handler at the server side.

- (9) The Resource Handler provides the server application with the resource information.
- (10, 11) The server application responds to the client application with adaptive content based on the monitored resource information and the appropriate adaptation.

4 The Prototype

We have built a prototype system based on this architecture. The system consists of two components: the middleware and the adaptive mobile applications. The middleware is used to support resource awareness. The adaptive applications are the mobile applications modified to work with the middleware. Three example mobile applications, a Web Server, a Video Server and a FTP Server, each representing one popular category of popular network application, are part of this prototype.

4.1 Assumptions

The assumptions that are made in the implementation of the example system are:

- Unicast. The communication is based on one client and one server.
- No proxies between the server and the client. The client and the server are communicating through sockets directly.
- The server has enough capacity to support clients. Scheduling at the server is not considered at this stage of the research.

4.2 Middleware Design Decisions

The middleware consists of a server component and a client component, both of which are written in Java. The client component itself is acting as a server at the client device, waiting for the request for the static resource information of the client, and sending the information.

Some important design decisions include:

- A pull approach [3] will be used to simplify the implementation complexity.
- The responsibility for resource awareness is divided between the application and the middleware. The middleware takes on the responsibility for monitoring while the application takes on the responsibility of adaptation.
- Both on-demand and continuous monitoring [1] of resources will be applied. On-demand monitoring happens at the very beginning of a connection and static resource information will be retrieved. Continuous monitoring is used to obtain dynamic resource information when needed.
- Generalization. The static information will be analyzed and reported in device types while the dynamic information will be reported in raw mode, leaving the application to make decisions based on its individual requirements.

4.3 Application Design Decisions

Three representative mobile applications, a Web Server, a Video Server and a FTP Server, are developed in this system to represent different categories of resource requirements, and to illustrate the types of choices that can be made. The server side of each application is modified from its original version to fit into the system model while the client side remains the same. That means the server and the client of one application might be written in different programming languages.

The World Wide Web (WWW) is the most popular Internet application. The version implemented, the Web Server, is based on HTTP 1.0. The Web Server detects that a pocket PC is the client and sends a suitable version of the requested page while detecting the existence of a smaller screen size of 240*320 pixels. Table 5 shows the resource requirements of a Web server.

Adaptation is a key feature of applications which support video data because of the pressure from transferring huge data volumes. Some adaptation strategies can be applied based on

Table 5: Resource Requirements of the Web Server

Static Resources	Dynamic Resources
Screen Size Screen Type Bandwidth, etc.	Transfer Rate, etc.

Table 6: Resource Requirements of the Video Server

Static Resources	Dynamic Resources
Screen Type Screen Size Bandwidth Disk Capacity RAM Capacity, etc.	Transfer Rate CPU Load RAM Load, etc.

the resource information, including transforming compression type, dropping frames, using low pass, reducing colour resolution, reducing size and reducing fidelity when bandwidth drops, increasing frame rate, using high pass, increasing colour resolution, increasing size and increasing fidelity when bandwidth increases. Table 6 shows the resource requirements by a video server.

The current Video Server will distinguish a client based only on the Screen Size, CPU Capacity and Bandwidth, although more resource information could be of interest to a video server. Screen Size will affect the image size of a video requested. A video with image size no more than 240*320 pixels will be transferred to a pocket PC. CPU Capacity will affect whether or not the video is compressed. Uncompressed video will be sent to a mobile device with low CPU capacity, for instance, lower than 266 MHz. A divx [12] compressed video will be transferred on the other hand. Bandwidth is the main factor that affects the file size to be transferred. In the current design, a decreased frame rate will be applied to uncompressed video while the bandwidth is decreased, whereas reduced quality will be applied to compressed video while the bandwidth is decreased. The choices are based on the current video processing tool we have and an estimate of file sizes.

FTP Service is selected to represent those applications that do not care about the length of the transmission time, only that the contents must be transferred eventually. Also, there is no particular display requirement on the client device. A FTP Server has been modified to have the ability to communicate with the middleware. This application simply bypasses the middleware system while running. Therefore, no adaptation strategy is applied and no particular resource information is needed.

5 Testing the Prototype

5.1 Testing Configurations

The test devices include a desktop computer (Dell Dimension 4100), a notebook computer (Toshiba Satellite 2000) and a pocket PC (Compaq iPaq H3670). These are selected to represent three of the most popular types of client devices. Current desktops have over 1.3 GHz CPUs, tens of GB of disk space, over 512 MB memory, and displays up to 19". They can be wired with the fastest network connections, for example, 100 Mbps. Because of the desktops' low price, the users can always keep their computers in the leading edge of the technology. Notebooks today can be as good as the best desktops in CPU and memory, but a bit poor on disk space (tens of GB) and limited display sizes (up to 15"). When the notebook is not moving, it can be connected to the fastest network the same as the desktop. When the notebook is mobile, the connection to a network is limited by the communication service that the notebook can use. From the pocket PCs, iPaqs are picked because they are the most popular among the pocket PCs. The iPaq used here is a new handy mobile facility that has a 206 MHz CPU, which is similar to a Pentium II desktop, 64 MB RAM, a 4" colour display, and no hard disk. Due to the iPaq's convenience in mobility and the limitations in resources, the users have different expectations while using mobile applications. Hence, the desktop and the notebook are selected as resource-rich devices, while the pocket PC is selected as a resource-poor device, in CPU, memory, disk space, and display. The desktop is selected as resource-rich device in connection, and the notebook and the pocket PC are selected as resource-poor devices in connection. Example configurations tested are described in Table 7.

5.2 Lessons Learned

Some important lessons have been learned while building the prototype.

- The service of a mobile application can be enhanced by sending adapted data based on resource information. The architecture in this research divides the responsibility of adaptation between the system and the application. This approach can reduce the complexity of modifying individual applications and increase the central control of the system by seeking a balance point between *ad hoc* approaches and *application-transparent* approaches. Furthermore, not only the variety of communication ability but also the heterogeneity of the mobile devices should be taken into account in the modern network environment.

- There are quite a few design issues arising on how to design a proper system in different network scenarios. How to build such adaptive mobile applications? Is a system needed to manage these applications? Where should this system be located (at the server side or the client side)? How does the system monitor resources? How do the system and the applications divide their responsibilities? How can the adaptation strategies be applied?

  Because of the wide variety of mobile applications, the answer to these questions is always "it depends." There are always advantages and disadvantages to any solution. Certain application-specific conditions may influence the developer to choose an approach closer to one end of the spectrum or the other. These considerations and the answers to above questions have been addressed in Section 3 and Section 4.2.

- The complexity of adaptation is application-specific. On the other hand, it is very easy to modify a server application by implementing the Interface of AdaptiveServer to send resource information request and retrieve resource information from middleware through the interfaces defined.

- This prototype could support pocket PCs in sending Web and video data adaptively. In this stage, the adaptation is based on some static information mainly to satisfy pocket PCs having limited resources: screen size, CPU capacity and the bandwidth it encounters

- The difficulty of dynamically transferring video data comes from three constraints: 1) Compression/Decompression speed of current computing ability. 2) The limited playing support on current pocket PCs. 3) The difficulty of retrieving resource data through Window CE.

6 Conclusions and Future Work

A new architecture that supports the adaptive mobile applications is proposed in this research. The architecture introduces an effective way to provide the adaptive mobile applications with the resource information that they are interested in by applying the middleware for adaptation services. Heterogeneous mobile devices such as pocket PCs can take advantage of adaptation tailored for their limited resources. Also, bandwidth is saved while the mobile user satisfaction is still maintained. In this design, the responsibility of monitoring and adapting is carefully divided by the middleware and the mobile applications. Modifications of existing individual applications are limited on server side. The design overcomes the drawbacks of the *application-transparent* approach and eliminates the increased complexity using *ad hoc* approach.

In the next step, the dynamic information should be taken into account by the prototype. More heuristical adaptation strategies should be

Table 7: Configurations for Devices Used in this Research

Desktop Computer	Notebook Computer	Pocket PC
Dell Dimension 4100	Toshiba Satellite 2000	Compaq iPaq H3670
Screen Type: colour	Screen Type: colour	Screen Type: colour
Screen Size: 1024x768	Screen Size: 1024x768	Screen Size: 240x320
Bandwidth: 10M	Bandwidth: 56.8k	Bandwidth: 10M
CPU Capacity: 866M	CPU Capacity: 850M	CPU Capacity: 266M
Disk Capacity: 15G	Disk Capacity: 20G	Disk Capacity: 0
RAM: 128M	RAM: 128M	RAM: 64M

studied in order to exploit the resource information retrieved. Furthermore, a C version of the architecture could be built to serve those adaptive mobile applications in a language other than Java.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by TR*Labs*, Saskatoon.

References

[1] Acharya, A., Ranganathan, M., and Saltz, J. Sumatra: a language for resource-aware mobile programs. Mobile Object Systems, Lecture Notes in Computer Science, February 1997. No. 1222, Springer Verlag (D), pp. 111-130.

[2] Badrinath, B., Fox, A., L.Kleinrock, Popek, G., Reiher, P., and Satyanarayanan, M. A conceptual framework for network and client adaptation. *ACM Mobile Networks and Applications 5* (December 2000), pp. 221–231.

[3] Choi, A., and Lutfiyya, H. Delivering adaptive web content based on client computing resources. In *Proceedings of the 8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI2001)* (Toronto, ON, May 2001), pp. 112–132.

[4] Curran, K., and Parr, G. An adaptable framework for streaming media to mobile applications. In *Proceedings of Wireless 2001* (Calgary, AB, July 2001), pp. K–3.

[5] de Lara, E., Wallach, D. S., and Zwaenepoel, W. Puppeteer: Component-based adaptation for mobile computing. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems* (San Francisco, CA, March 2001), pp. 26–28.

[6] Demers, A. J., Petersen, K., Terry, M. J. S. D. B., Theimer, M. M., and Welch, B. B. The bayou architecture: Support for data sharing among mobile users. In *Proceedings of Workshop on Mobile Computing Systems and Applications* (Santa Cruz, California, December 1994), pp. 2–7.

[7] Howard, J. H., Kazar, M. L., Menees, S. G., Nicholas, D. A., Satyanarayanan, M., Sidebotham, R. N., and West, M. J. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems 6*, 1 (February 1988), pp. 51–81.

[8] McIlhagga, M., Light, A., and Wakeman, I. Towards a design methodology for adaptive applications. In *Proceedings of MOBICOM'98* (Dallas, TX, October 1998), pp. 133–144.

[9] Noble, B., and Satyanarayanan, M. Experience with adaptive mobile applications in Odyssey. *Mobile Networks and Applications 4*, 4 (1999), pp. 245–254.

[10] Noble, B. D., Price, M., and Satyanarayanan, M. A programming interface for application-aware adaptation in mobile computing. In *Proceedings of Second USENIX Symposium on Mobile and Location Independent Computing* (Ann Arbor, April 1995), pp. 57–66.

[11] Perkins, C. E. Mobile IP. In *IEEE Communications Magazine* (May 1997), vol. 35, pp. 84–99.

[12] Project Mayo – Home of OpenDivx. http://www.projectmayo.com/. Web site.