Metaspaces and Mobile Computing: Promises and Challenges

Eric G. Manning[1], Shahadat Khan[2], Rodger Lea[3], Gholamali C. Shoja[1] and Michael M. J. Zastre[1]

[1] Department of Computer Science, University of Victoria
[2] Department of Electrical and Computer Engineering, University of Victoria
[3] Sony Computer Science Laboratory Inc., Tokyo

Abstract. We briefly summarize the mobile computing environment from a computer science point of view, and discuss some of the interesting possibilities raised by the prospect of extremely large numbers of computational nodes networked over wireless channels. Among these are: the use of reflection and the metaspace construct to deal with some of the Quality of Service and Connectivity problems of mobile computing, the problem of dealing with the degraded user interfaces often implied by mobility, and the possible applicability of mobile objects to mobile computing.

1 Introduction

Mobile computing is an exciting field of research because of the deep questions it poses for researchers, which in turn often lead to new service opportunities. In the last few years standards committees in Europe in North America have fulfilled their mandate: lower communication layers now are provided with some solid standards [15], and the IPv6 proposals for a mobile protocol have wide circulation. A greatly increased rate of mobile computer deployment appears imminent.

What will not disappear is the hard work of designing, developing and operating mobile computing systems that are *adaptable*; end-users, service providers, and the communication infrastructure introduce a multi-way tug-of-war that makes this difficult. *Users* specify the policies, and their mobile hosts contain the mechanisms, for selecting one of several telecommunications media. Media will vary in quality, cost, latency, speed, and availability. *Service providers* maintain systems facing a changing (and if lucky, an evolving) set of user applications. Despite the speed of these changes, however, users demand high availability, flexibility, capacity, and low cost for the wireless-based services they purchase. Micro- and pico-cellular *infrastructure* connecting together users and providers must work with usage patterns that make provisioning difficult. *Application software developers* need to specify a QoS (Quality of Service) which in turn implies a *resource profile* required by the application to achieve that QoS [12]; this includes telecommunications channel bandwidth BW, primary and secondary memory M_p and M_s), and CPU cycles C:

$$\overline{R}^p = (BW^p, M_p^p, M_s^p, C^p) \tag{1}$$

but the actual resources available at runtime

$$\overline{R}^a = (BW^a, M_p^a, M_s^a, C^a) \tag{2}$$

may not match the assumed profile. *Adaptation* of the application to the resources available is required. Therefore *adaptation should be a core concept upon which these systems are built.* That is, it should support systems where either equation 3 holds (best case), or equation 4 holds (adaptive case). The variable j is an index into the profile tuple.

$$\forall j \;\; R_j^p \leq R_j^a \tag{3}$$

$$\exists j \; s.t. \; R_j^p > R_j^a \tag{4}$$

Yokote [23] has argued that *metaspaces* provide a mechanism for implementing adaptation. The use of metaspaces in achieving application-transparent adaptation is a topic of research at the University of Victoria [4, 5, 10, 12, 17], and our comments stem from these studies. The purpose of this paper is to show the following:

> *The adaptation needs of a mobile computing environment may be supported by meta-spaced based systems.*

Our discussion begins with a description of the metaspace concept and the role it can play in facilitating adaptation (Section 2). Exploring the empowering potential of this technology is great fun, so we will indulge ourselves a little by exploring the *promises* (Section 3) in order to focus on the *challenges*, with the latter well-served by adaptation (Section 4). In light of these challenges, we suggest how metaspaces can be used through two extended examples. (Section 5). We conclude with some directions for research (Section 6).

2 Adaptation and Metaspaces

Programs require an environment on which to act: memory, file systems, thread control, CPU cycles, telecommunications media bandwidth, etc. The designer's concerns over programs should be separate from concerns about the environment. However, current systems frequently meld these concerns; a different system architecture is needed to keep these dimensions separate. The application designer should be able to assume a stated *resource profile*, and design under the simplifying assumption that these resources will be available, as assumed, at runtime. Discrepancies between the assumed resource profile $\overline{R}^p$ and the resources actually available $\overline{R}^a$ should be managed by the operating system, which *adapts* the application to the resources available. We call this *application transparency* to *resource availability.*

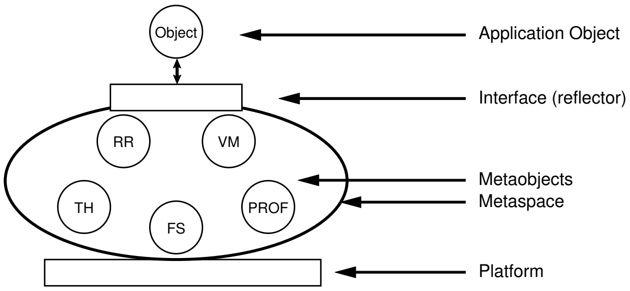

Fig. 1. Metaspace and one user object

The *Apertos* metaspace architecture encapsulates the operating system environment into objects, each object corresponding to some implementation decision, e.g. for memory management, CPU scheduling, IPC, etc. These objects stand in special relationship to application programs (or application objects), and are called *metaobjects*. A collection of metaobjects, plus a distinguished object through which all communication with application objects are passed, is called a *metaspace* (see Figure 1). For instance, metaobjects in the example support Persistency, Virtual Memory, Thread Control, a File System, and messaging. The distinguished interface object is called a *reflector*. The metaspace architecture was originally proposed by Yokote et.al. [23] to implement computational reflection [13, 16, 19]; we have found it valuable in implementing adaptive features in operating systems, whether or not they constitute computational reflection.

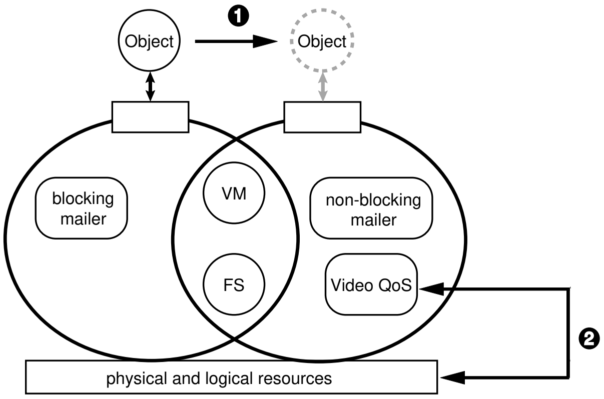

Fig. 2. Two metaspaces with object migration (logical view)

The flexibility of a metaspace architecture has two potential benefits:

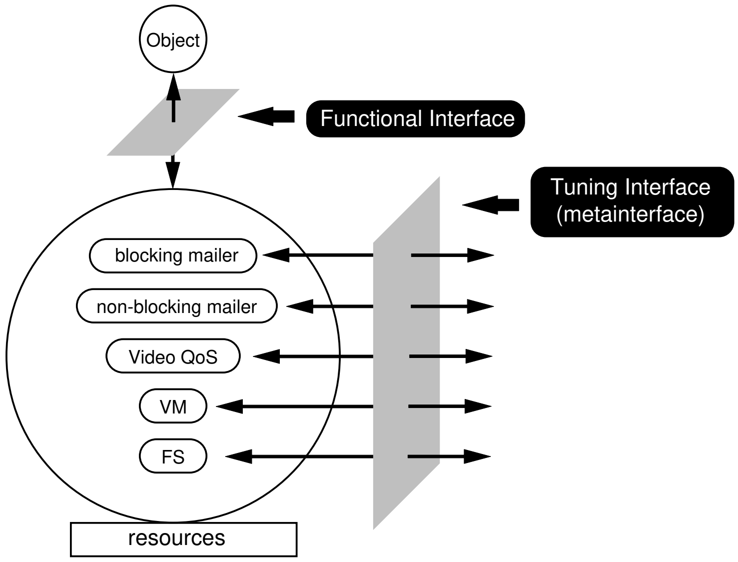

Fig. 3. Another view of system in Figure 2 with interfaces shown

- the simplicity with which an object may potentially takes on new properties, and
- the ability to isolate objects from run-time decisions in the metaspace.

Both are illustrated in Figure 2. Observe that metaobjects may be within more than one metaspace, although there is still one platform. By migrating to the metaspace on the right (arrow labeled "1"), the application object is provided with non-blocking message passing; it is also in an environment with QoS support for video. In both of these cases, our application object needs no modification; instead, the system generates any new bindings. This metaspace architecture can neatly implement adaptation. Suppose

$$BW^p > BW^a \tag{5}$$

as signalled by the platform ("2" in Figure 2). This signal is mapped into a meta-interface call to the Video QoS object which adjusts the actual video data rate V so that

$$V = BW^a \tag{6}$$

(for example, by discarding appropriate components of the video stream as is possible with *scalable codes* such as MPEG).

Figure 3 presents another representation of these two metaspaces; the usual functional interface or programmer API is at the top, and all metaobjects are together. The metainterface is on the side; through this "tuning interface," the metaspace may be queried and controlled by the platform or some other application. For instance, the application object's need for non-blocking messaging would involve interaction through the metainterface to the "mailer" metaobjects. The Video QoS metaobject can be queried by the platform, the object's settings also being modifiable. The metainterface can be analogized to "knobs" on the side of some some black box; they are rotated to various settings as needed.

Our example is of course simplified. A real metaspace architecture has two or more levels of metaspace [5, 10, 23], and the design and specification of the metaspace interface (reflector) requires careful thought because application objects only migrate between compatible interfaces. However, all good design separates design concerns into orthogonal dimensions, and here we have potentially gained a method for adding, modifying or removing behavior from an application without re-writing it.

3 Promises

Communication
Extrapolating from our patterns of daily work in front of workstations, we can arrive at some straightforward uses of mobile computers. These may be considered as simple terminals for a remote system. A more aggressive view regards them as the clients for a multimedia server, such as those for video-conferencing; adaptation is clearly a concern when dealing with streams of video and voice. A laptop client powered by vehicle power could be expected to do more processing (and hence the server might do less) than an internally-powered palmtop device.

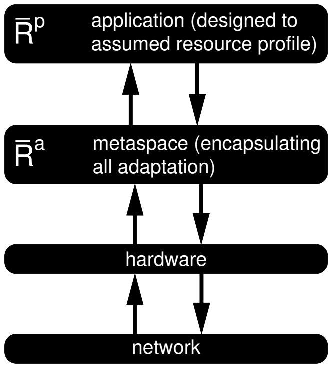

Fig. 4. Separation of concerns

This suggests an important style of *adaptation*. Clients respond to data loss over a relatively lossy medium; servers respond to varying configurations of clients. Building adaptation into the application layer leads to complex code that is difficult to extend as it violates *separation of concerns* (see Figure 4).

Metaspaces, we contend, provide a promising way to separate these concerns.

Collaboration
With an ad-hoc network and CSCW (Computer-Assisted Cooperative Work) system distributed amongst mobile hosts, knowledge workers can work where they wish and still have access to shared data, whether stored remotely (requiring at least one host within wired network range), replicated locally, or migrating from host to host. The degree of connectivity and quality of the network will determine the method of interaction between the workers; again, adaptation of the application to the services actually available should be hidden from the users.

Citizens and Society
We all interact with our neighbours and community, whether through direct contact, or through our shared interest in the mass media. Personally tailored news, stock information, music, video and other entertainment items, have great appeal. Within homes, TV-top boxes or cards within the TV set which facilitate WWW access are poised to make great inroads in 1997 consumer electronic sales; given suitable wireless bandwidth, these same browsers could literally rest in the hands of the users. Experiments have shown that wireless browsers are feasible, with the primary problems being bandwidth and mobile host processing power [3]. Hence there will be a very heterogeneous set of clients, ranging from clients realized by powerful desktop machines to clients embedded in TV sets. Clearly the server should not be designed for the weakest, so extensive adaptation of the service by the server to cater to the weaker clients will be required.

Control
In some environments, wiring for networks is either undesirable (old buildings, temporary office space) or impossible (emergency areas; locales with little or no existing telecommunications infrastructure). Wireless systems can also be valuable for the quick deployment of surveillance and security systems for homes and offices, especially where the time between recognition of a threat and the deadline for action are beyond the capability of wired systems. Requirements for these quickly approach those of hard-real time systems, where adaptation, and especially QoS, must be strictly specified, controlled, and administered.

4 Challenges

The limits of mobile computers are well known: these devices have less memory, disk space, battery life, and less power for wireless transmitters. The demands of connection management upon networks consumes scarce bandwidth. We suggests that *an environment treating adaptation as a core concept* is a valuable tool for dealing with these challenges.

Varieties of wireless communication

There are many wireless media: infrared for line-of-sight transmission delivering up to 2Mb/s; gigahertz radio such as that for PCS and ISM communication promising 2; and cellular networks running at up to 19.6 Kb/s. In Canada, licenses for Local Multipoint Communication Systems (LMCS) have just been introduced. LMCS systems will operate in the 28 GHz range, providing 0.5 or 1 GHz bandwidths, and a signal radius of 4.5 km. (The initial application domain is cable TV service.)

However, bright sunlight can block infrared; multi-path interference at radio frequencies requires compensation. Significant efforts have been made to isolate applications from the intrinsic defects of the wireless medium, leading to data link layer standards [15]. What remains are media which have bandwidths and loss rates differing from each other by orders of magnitude; this cannot be hidden from applications without significant effort. *How* and *where* that effort is expended can determine ease of system adaptation. Adding support for an order-of-magnitude improvement in media performance must not translate into an order-of-magnitude larger maintenance effort.

Mobility and the Internet

Wired networks are ubiquitous, and users are beginning to expect the same ubiquity for wireless. Network communication combined with mobility leads to expensive network management: maintaining mobile IP addresses; dealing with domain changes [21, 20]; forwarding packets; managing handoffs (an increasing important problem as cell sizes shrink [1]); and supporting "degrees of connectedness" from zero to maximum data rate. Clients may originate sessions in a Mobile IP-supported network, move to one supporting Virtual IP [21, 20], or in the absence of a base stations and wired routers, into an ad-hoc network supporting some arbitrary topology [11, 18].

One solution to disconnections is for the mobile computer to operate asynchronously, i.e. it can send multiple requests before waiting for acknowledgment. Pre-fetching and lazy write-back can decouple the act of communication from the act of consuming data, thus making it possible to do useful work during network disconnections [8]. A good example is the Coda file system, which has been designed for notebook computers [14].

To handle the disconnection problem, both the applications and runtime systems for mobile computers must be designed for adaptation at run-time. Run-time systems should mask the effects of network disconnections and deliver the quality-of-service requested by user applications. On the other hand, user applications must accomplish their objectives with the varying resources provided at run-time. Adaptive solutions exist, but their complexity is also acknowledged [7]. A metaspace-based architecture divides functionality and adaptation concerns between an "interface" and "metainterface", therefore reducing this complexity.

Constraints placed by mobility on physical design

Not much has changed since the 1994 survey of mobile devices by Forman and

Zahorjan [8]. Battery capacity, physical interface size, processing power and transceiver range are traded off against each other.

There is room for adaptation even here. For instance, the HIPERPLAN wireless standard has provisions for quasi-dormant operation. Devices may sleep and only operate when expecting a multicast; another provision has packets structured such that devices can "listen to the header", and quickly decided whether or not to listen to the rest [15]. Since transceivers consume the bulk of electrical power, these provisions in the standard can significantly extend battery life. Services must adapt to the behavior of their clients who need these modes of operation, but also maintain full service for those clients without power constraints.

The asymmetry between server and client processing power has already suggested a similar breakdown in communication processing. However, a server cannot deal with disconnected clients without risking exhausting its memory and CPU resources (e.g. buffering packets, maintaining timing information and windows). More wireless clients can be serviced by an adaptive asymmetric server, and again we see that metaspaces can separate the concerns of service and adaptation. This relative ease in implementing adaptability, plus the reduced effort to introduce new policies, translates into increased revenue for the service provider.

5 Approaches

Here we take a closer look at some of the mobile computing applications mentioned in the previous two sections: *mobile video conferencing* and *mobile user objects*.

Mobile Video Conferencing

Cellular phones are ubiquitous today, so a natural extension is to think of mobile systems as platforms for video conferences. A platform such as an automobile would provide considerable power and space, eliminating two traditional constraints of these systems. Sessions will involve other participants who might be wireless or wired.

However, the demands placed on wired networks by multimedia are already high, and will stretch wireless media to their limits. Limited bandwidth, high error rates, and very long latencies, combined with a potentially insecure broadcast medium, all create challenges for system designers. Adaptability is equated with scalability, and should be implemented within either the network or data link layers; however, applications adapt by interacting with these layers, or are themselves accessed by lower layer upcalls. Therefore, layered multimedia systems are a misnomer: adaptability issues exist in several layers. Alwan et. al. [2] have investigated some of the needs of these adaptive mobile multimedia networks, and report mixed performance for admittedly difficult problems.

A metaspace-based architecture for multimedia [12] is shown in Figure 5. A microphone and camera are used to gather sound and images of the user; a

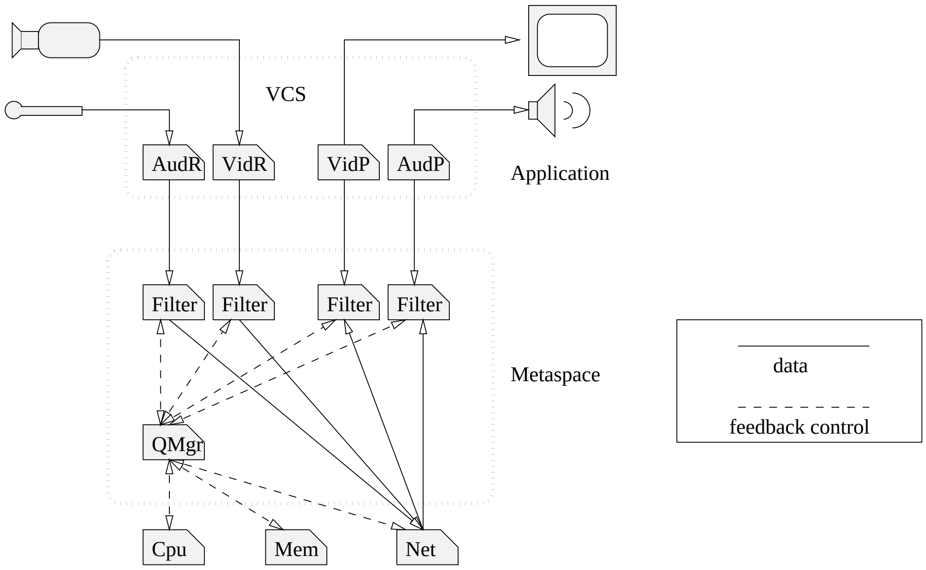

Fig. 5. A metaspace-based video-conferencing system.

speaker and CRT display play the remote signals. There are four corresponding objects at the application level: audio recorder (AudR), video recorder (VidR), audio player (AudP) and video player (VidP). These objects are not concerned with quality control issues as these are provided by the underlying metaspace. The quality manager (QMgr) is at the heart of the metaspace, tracking system dynamics and taking adaptive measures to make the best use of system resources. It therefore processes signals triggering adaptation, such as network congestion, and runs an adaptation engine to take control decisions, e.g. which video stream should be filtered, which task should be allocated more CPU cycles.

Mobile User Objects

Mobile hosts may range from powerful laptops powered by a vehicle power supply, to handheld devices relying on internal batteries which severely limit wireless transmitter output and airtime. For any given task, how much of the processing can be performed on the mobile host, and how much farmed out to neighbouring stations? There is now recognition that *mobile objects* (also called *mobile agents*) can be an important element in mobile computing's toolkit: they can provide high bandwidth remote interaction, support for disconnected operation, support for weak clients, can ease distribution of individual service clients, can route based on message semantics, are scalable, and have robust remote interaction [9]. For all of their power, however, mobile objects have weaknesses, the most significant being the many ways in which their security may be compromised.

One way of dealing with these issues of security is to view them as one of adaptation: objects are granted reduced use of system resources when they reside on a untrusting (or untrusted) platform; hosts, whether fixed or mobile, will grant privileges to visiting objects only as qualifications or identity are proved. For instance, messaging between untrusted objects on a platform may be given specially tuned semantics, such as those in Secure Object Spaces [22] where untrusted objects communicate by exchanging tuples through a passive data area.

Metaspaces provide a natural expression of both mobility and the acquisition of properties by objects; in Figure 6, an object's migration from one system to another is shown (arrow #1). After arriving, the object is bound to the metaspace containing SOS messaging properties. These message facilities may provide enough functionality, and the object will continue processing in that metaspace. However, if more privileges are needed, and if the object succeeds in authentication, it can move into the more privileged metaspace (arrow #2). All bindings for application method calls are handled by the system and the metaspace interface (reflector).

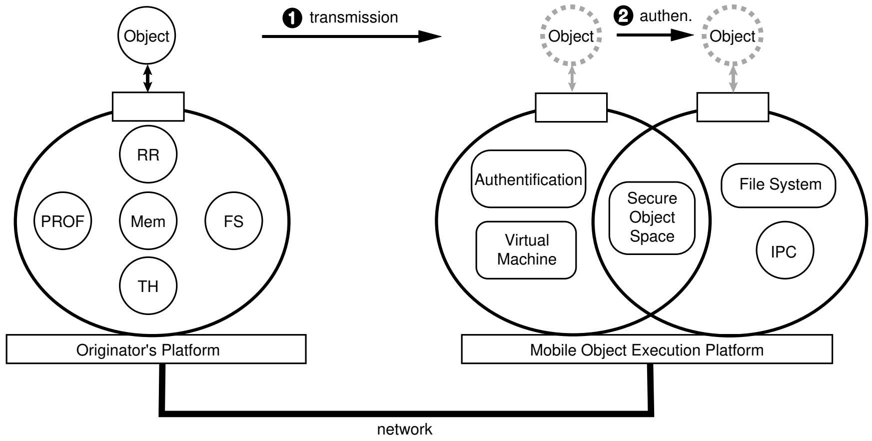

Fig. 6. Mobile user objects

The other marriage between metaspaces and mobile objects reveals its power when adaptation is performed by system users or application developers. For instance, the distribution of data over the machines on a mobile network may change, making it more logical, or better for system performance, if some code is executed at the data site. It would be easier and perhaps even more accurate for the mobile user to perform data acquisition and to initiate processing. Metaspaces ensure that this is clearly expressed. As an another example, enforcing consistency of database views at different mobile hosts can be done by storing data centrally, or by replicating it with all of the associated headaches of maintaining consistency. However, a more appropriate implementation decision may be to migrate data to the mobile host [6]. The user of the data may be a sales representative (or building inspector, or architect, etc.) dealing on-site with a client; there would be no reason for all other mobile hosts to always have up-to-date views of everyone's data. These decisions are best made by someone other than the original file or database system vendor. Metaspaces ensure that *the location of objects may adapt to the changing needs of the system.*

6 Conclusion

We have surveyed some of the problems facing mobile computing, arguing that adaptation within these systems is an important requirement. We then suggested that metaspace concepts can be a valuable tool to realize system design. Some examples were shown, exploring what is possible if adaptation is a core concept of an OS supporting mobile servers or mobile hosts.

There are many areas for research:

- Demonstration of an ATM driver and QoS manager working in a metaspace architecture.
- Development of metaobject protocol designs for mobile systems.
- Investigate each resource type (CPU, primary memory, secondary memory, ATM streams, etc.) and find ways in which to scale them in the context of QoS. Scalable codes such as MPEG are essential.
- Multiple instances of one application executing within different metaspaces, and all executing with respectable performance.
- Holy Grail: demonstration of a suite of applications written assuming a fixed environment and then migrated into a mobile system with *no code changes*: an ambitious target indeed, and one worth striving for, even though it may only be achievable in part.

Acknowledgments

This paper has benefitted from comments and conversations between members of the Parallel and Distributed Computing Group (PANDA), a research group at the University of Victoria in the Faculty of Engineering. Its members investigate distributed system design and implementation, including operating system support and extensibility, programming languages, compiler technology, computational reflection, mobile objects, and support for adaptive multimedia applications. Collaborations with industry include projects with the Sony Computer Science Lab, Tokyo, and with Nortel Technologies (formerly Bell-Northern Research), Ottawa.

The authors especially wish to thank Robert Bryce, Glen Chen, John Foxgord, James Pang and Kin Li for their participation in many discussions on mobile computing's future. The PANDA web site is at: `http://panda.csc.uvic.ca/`

Michael Zastre is supported by a Natural Sciences and Engineering Research Council (NSERC) scholarship for post-graduate studies. Shahadat Khan is supported by a Canadian Commonwealth Scholarship and an NSERC fellowship.

References

1. A. S. Acampora and M. Naghshineh. Quality-of-Service Provisioning in Microcellular Networks. In *IEEE INFOCOM 1995*, volume April, pages 1075–1084, 1995.
2. A. Alwan, R. Bagrodia, N. Bambos, M. Gerla, L. Kleinrock, J. Short, and J. Villasenor. Adaptive Mobile Multimedia Networks. *IEEE Personal Communications*, pages 34–51, April 1996.
3. Joel F. Bartlett. Experience with a Wireless World Wide Web Client. Technical Report TN-46, Digital Western Research Laboratory, March 1995.
4. R. W. Bryce, M. Horie, E. G. Manning, and G. C. Shoja. The Mirrors System. Technical Report DCS-250-IR, Department of Computer Science, University of Victoria, 1996.
5. Robert W. Bryce. Enhancing Real-Time Performance of an Object-Oriented Operating System. Master's thesis, Department of Computer Science, University of Victoria, 1995.
6. Paul Dourish. Developing a Reflective Model of Collaborative Systems. *ACM Transactions on Computer-Human Interaction*, 2(1):40–63, March 1995.
7. Dane Dwyer and Vaduvur Bharghavan. A Mobility-Aware File System for Partially Connected Operation. *ACM SIGOPS Operating Systems Review*, 31(1):24–30, January 1997.
8. George Forman and John Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, April 1994.
9. Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM T. J. Watson Research Center, 1995.
10. M. Horie, E. G. Manning, and G. C. Shoja. Exploring the Apertos Operating System. Technical Report DCS-238-IR, University of Victoria, September 1995.
11. David B. Johnson and David A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications*, pages 34–42, February 1996.
12. M. S. Khan. Quality Adaptation in a Distributed Adaptive Multimedia System: Model and Architecture. Ph.D. Proposal, January 1997.
13. Gregor Kiczales, J. des Rivieres, and D. Bobrow. *The Art of the Meta-object Protocol*. MIT Press, 1993.
14. J.J. Kistler and M. Satyanarayanan. Disconnected Operations in CODA File System. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.
15. Richard O. LaMaire, Arvind Krishna, Pravin Bhagwat, and James Panian. Wireless LANs and Mobile Networking: standards and future directions. *IEEE Personal Communications*, pages 86–94, August 1996.
16. P. Maes. Computational reflection. Technical Report Tech. Rep. 87.2, Vrije Universitaet, Brussels, Belgium, 1987.
17. Kenichi Murata, R. Nigel Horspool, Yasuhiko Yokote, Eric G. Manning, and Mario Tokoro. Cognac: a Reflective Object-Oriented Programming System using Dynamic Compilation Techniques. In *Proceedings of the annual conference of Japan Society of Software Science and Technology 1994 (JSSST94)*, October 1994.
18. Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *SIGCOMM Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.
19. B. Smith. Reflection and semantics in a procedural language. Technical Report MIT-TR-272, Laboratory for Computer Science, MIT, Cambridge, Mass., 1982.

20. F. Teraoka, Y. Yokote, and M. Tokoro. Virtual Network: Towards location transparent communication in large distributed systems. Technical Report TR 90-005, Sony Computer Science Laboratory, June 1990.
21. Fumio Teraoka and Keisukie Uehara. Mobility Support in IPv6 based on the VIP Mechanism. http://info.isoc.org/HMP/PAPER/042/html/paper.html, April 1995.
22. Jan Vitek. Secure object spaces. In *2nd International Workshop on Mobile Object Systems, ECOOP '96*, 1996.
23. Yasuhiko Yokote. The Apertos Reflective Operating System: the concept and its implementation. In *Conference on Object Oriented Programming Systems Languages and Applications*, pages 414–434, 1992.

This article was processed using the LaTeX macro package with LLNCS style