Challenges and Solutions to Adaptive Computing and Seamless Mobility over Heterogeneous Wireless Networks

Vaduvur Bharghavan
Center for Reliable and High-Performance Computing
Coordinated Sciences Laboratory
University of Illinois at Urbana-Champaign
bharghav@crhc.uiuc.edu

Abstract

Recent years have witnessed the rapid evolution of commercially available mobile computing environments. This has given rise to the presence of several viable, but non-interoperable wireless networking technologies - each targeting a niche mobility environment and providing a distinct quality of service. The lack of a uniform set of standards, the heterogeneity in the quality of service, and the diversity in the networking approaches makes it difficult for a mobile computing environment to provide seamless mobility across different wireless networks. Besides, inter-network mobility will typically be accompanied by a change in the quality of service. The application and the environment need to collaboratively adapt their communication and data management strategies in order to gracefully react to the dynamic operating conditions.

This paper presents the important challenges in building a mobile computing environment which provides seamless mobility and adaptive computing over commercially available wireless networks. It suggests possible solutions to the challenges, and describes an ongoing research effort to build such a mobile computing environment.

1 Introduction

Recent years have witnessed explosive growth in the field of mobile computing, resulting in the development of mobile computing environments which can provide a very respectable level of computing and communications capabilities to a mobile user. Unfortunately, one consequence of this rapid evolution has been the emergence of several viable, but non-interoperable wireless networking technologies, each targeting a specific operation environment and providing a distinct quality of service (QoS). The lack of a single set of standards, the heterogeneity in QoS, and the diversity in networking approaches makes it difficult for a mobile computing environment to provide *seamless mobility* across different wireless networks.

Since inter-network migration may result in significant changes in bandwidth or other QoS parameters, *adaptive computing* is necessary in a mobile computing environment which provides seamless mobility over multiple wireless networks. A graceful reaction to sudden QoS changes can only happen when both the environment and the applications collaboratively adapt to the

dynamic operating conditions. Seamless mobility and adaptive computing are thus related and complementary goals.

The ideal scenario in mobile computing is the following: a mobile user carries a portable computer with one or more built-in wireless network interfaces (ranging from O(Kbps) CDPD WMAN to O(Mbps) 2.4 GHz WLAN), and moves around both indoors and outdoors while executing applications in a uniform working environment (oblivious of the dynamics of the underlying inter-network migration). In order to achieve this scenario, the mobile computing environment needs to provide at least the following functionality:

- *seamless mobility* across different wireless networks.
- graceful *adaptation* to dynamic QoS variations.
- a *uniform framework* for executing applications across diverse underlying environments.
- *backbone support* for mobile applications.
- *seamless recovery* in the presence of failures.

In summary, the goal of a mobile computing environment is to provide the illusion of a uniform working environment (with possibly dynamically varying QoS) on top of underlying heterogeneous technologies.

Inherent limitations of wireless and imposed limitations of the state-of-the-art technology currently restrict the realizability of the provisions above (example of the former: the bandwidth of a wireless MAN is one to two orders of magnitude lower than a wireless LAN; example of the latter: switching between RAM and CDPD may require the portable computer to reboot). The focus of this paper is to distinguish the inherent scientific limitations from the state-of-the-art limitations, and identify the challenges which need to be solved in order to approximate the ideal scenario.

In contemporary research, there is no working implementation of a mobile computing environment which provides seamless mobility on top of heterogeneous commercial networks. Several critical research issues in both seamless mobility and adaptive computing have not been addressed, or even identified. In order to understand the challenges in this field, and also experiment with some preliminary solutions, we started building the PRAYER[1] mobile computing environment, which provides a platform for seamless mobility and adaptive computing on top of diverse commercially available wireless networks. We achieve seamless mobility by providing the 'software glue' to hold together diverse commercial networks, and build an adaptive computing framework on top of the seamless network. Three important components of PRAYER relevant to this paper are connection management, adaptive file system support, and OS/language support for application-level adaptation. The connection manager enables seamless mobility over diverse networks and provides coarse estimates of available network QoS. The file system supports application-directed adaptation for partially connected operation. PRAYER provides simple OS/language support to structure applications in an adaptive environment. Preliminary results indicate that PRAYER offers an effective environment for adaptive computing and seamless mobility over commercial networks.

[1] Our system is named PRAYER after the most popular type of wireless uplink transmissions to an infinite server.

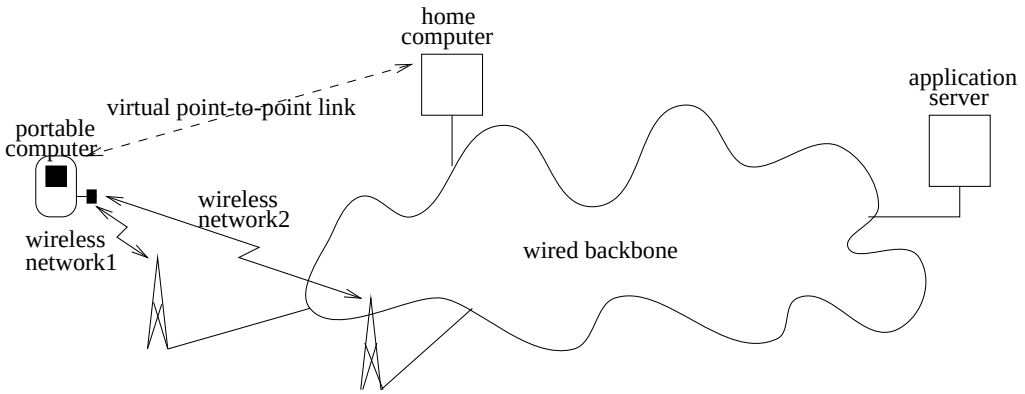

Figure 1: Mobile Computing Model

The rest of the paper is organized as follows. Section 2 describes the target mobile computing model. Section 3 identifies the research goals. Section 4 explores the research challenges and evaluates different approaches. Section 5 describes the PRAYER mobile computing environment. Section 6 summarizes related work, and Section 7 concludes the paper.

2 Mobile Computing Model

Our mobile computing model is fairly standard, as shown in Figure 1. Five components are of relevance: (a) *portable computer*, equipped with multiple wireless network interfaces (b) *home computer*, the repository of data and computing resources for a mobile user on the backbone network (c) *application servers*, e.g. database server, file server etc. (d) *wireless networks*, typically autonomously owned and managed, and (e) *backbone network*.

In our environment, the following points are distinctive.

1. A wireless network may be autonomously owned and operated. It may use proprietary network protocols. Its base stations will not be accessible for software changes to support seamless mobility. Therefore, it is important to operate on top of off-the-shelf commercial networks.

2. The wireless network may vary in its offered quality of service (QoS) (Figure 2) and can be parametrized by the following: bandwidth, latency, channel error, range, access protocol, connection failure probability, connection cost, pricing structure and security.

3. The portable computer may vary in sophistication from a PDA to a high-end notebook, and can be parametrized by the following: compute power, memory, available network interfaces, disk space, available compression techniques, battery power and native OS.

4. The dynamics of the underlying environment are primarily influenced by user mobility. A portable computer will see a variable QoS networking environment depending on its current connections (disconnection is the extreme case).

In the PRAYER model, the home computer plays an important role. It runs the application stubs and file system clients on behalf of the portable. Essentially, it performs the role of a client with respect to the application servers, and the server with respect to the portable. In this

Technology	Range	Bandwidth	Latency	Access	Cost
Ethernet (3Com)	km	10Mbps	100us	CSMA/CD	setup
T1 (OARNet)	country	1.5Mbps	10ms	TDMA	$1000/mth
2.4 GHz radio (Proxim)	500ft	1Mbps	5ms	CSMA/CA	setup
Infrared	30ft	1Mbps	1ms	point-to point	setup
Satellite (Microspace)	country	100Kbps	100ms	FDMA	$10000/mth
Radio mdm (Mobitex)	MAN	10Kbps	100ms	CSMA/CA	$100/MB
Cellular mdm (Motorola)	MAN	10Kbps	50ms	TDMA	40c/min
CDPD (AT&T)	MAN	10Kbps	500ms	CSMA/CA between AMPS	$80/MB

Figure 2: Comparison Chart of Wired/Wireless Networking Technologies (numbers represent orders)

model, the goal of seamless mobility is to establish a 'virtual point-to-point' connection between the portable computer and the home computer over multiple wireless networks. Likewise, the adaptive computing solutions focus on the consistency management issues between the home and the portable. Though restrictive, this model simplifies the systems architecture while still allowing us to study the issues in seamless mobility and adaptive computing.

3 Research Goals

The fundamental goal is to provide a mobile computing environment which enables a portable computer with multiple wireless networking interfaces to seamlessly move between the different proprietary networks without disturbing the application, and to provide systems support for applications to adapt to dynamic QoS variations gracefully. The technical goals fall into two classes: *service goals* - what services need to be provided to the applications, and *system goals* - what systems issues need to be solved to provide the services.

3.1 Service Goals

Three major types of activity from the portable computer include computation, communication, and information access. The mobile computing environment should provide services for each of the above within a uniform framework independent of the dynamics of the underlying system.

Computation: Delegating compute intensive and non-interactive communication-intensive tasks to a backbone computer saves critical wireless bandwidth, and also portable compute cycles. Remote *agent invocation* at the backbone is a general and flexible mechanism to provide backbone computation services [19].

Communication: Efficient, medium-transparent, secure communication is a primary service requirement. Since a portable computer may have multiple wireless connections, the efficient choice for communication depends on the application and network characteristics (e.g. CDPD

for short bursts of data, cellular modem for periodic data traffic). Medium-transparent communication requires *seamless migration* [4] (with possible notification of QoS change to applications) between autonomously managed and potentially non-interoperable networks.

Information access: Filtering information destined for the portable at the backbone network saves wireless bandwidth. This is traditionally achieved by application-level filtering [40]. An alternative approach, which we pursue, is to have applications use the file system as a filtering mechanism by imposing different semantics-driven consistency policies.

Uniform framework: User mobility, migration between wireless networks, and network failures/partitions cause the underlying environment to change dynamically. The mobile computing environment should provide a uniform framework for the abstraction of the current capabilities of the environment, and a simple mechanism for applications to be notified upon change in the environment.

3.2 System Goals

This section identifies the systems goals needed to support the services listed in Section 3.1 within the constraints imposed by the mobile computing environment in Section 2.

Seamless mobility: Medium-transparent data transport in the presence of network connections/disconnections requires seamless migration between the cells of a wireless network (common case), and between autonomously managed wireless networks with possibly very different QoS (general case).

Multiple connections: A portable computer may have simultaneous access to multiple wireless networks. Depending on the type of application data traffic, required QoS, and priority of transmission, the mobile computing environment should be able to make an appropriate choice of wireless network for data transmission. The portable should thus be provided with the ability to use different wireless networks for different types of application traffic.

Caching and consistency mechanism: Since the wireless medium is a scarce resource, caching data at the portable may significantly reduce access time and communication traffic, thereby improving performance and reducing cost. However, caching also introduces data consistency issues. While contemporary research has typically concentrated on caching and consistency issues for disconnected operation, partial connection (with varying degrees of QoS) will soon become a common mode of operation. Caching and consistency policies should adapt with the network QoS. We argue for application-directed consistency policies and simple APIs for applications to interact with the mobile computing environment in order to enforce these policies.

Seamless recovery: When network connections fail, it may be possible to use alternative redundant network connections in order to recover from primary link failure. Such recovery should be seamless, but may still change the caching/consistency policies due to a change in

QoS. Previously cached data needs to be reintegrated according to the revised consistency policies. Recovery of essential state and the mechanism for migration between different consistency policies should be transparent to the application.

Backbone support: Providing agent invocation and application-directed caching/consistency policies requires backbone support. We suggest the use of a dedicated home computer in order to provide the backbone support for executing agents, application stubs, and file system clients. Although this approach may induce greater overhead, it reduces the consistency management problem to just keeping the portable consistent with its home, and permits supporting different consistency policies on the backbone.

Application support: In order to effectively adapt to QoS changes in the network, applications need to make 'aware' decisions [31, 39]. The trade-off between providing applications the flexibility to adapt based on context-awareness, and introducing complexity in the application logic in order to handle such adaptation, is a delicate one. Ideally, the applications will control the *policies* for consistency management of their data, while the system will provide the *mechanisms* to realize these policies. The split between policies and mechanisms will enable the applications to adapt to the dynamic QoS while not being concerned with the actual mechanisms of imposing consistency.

4 Research Challenges and Solutions

The mobile computing environment consists of five entities: portable computers, home computers, application servers, wireless networks, and the backbone wired network. The wireless network is typically proprietary, and the base station is thus inaccessible for customization to support seamless mobility. The service goals in Section 3.1 all require some form of backbone support. The home computer is the natural entity to provide this support, since we cannot provide systems software support on the base stations or mobile service stations[2]. The system architecture at a very high level is thus fairly obvious: the computing entities are the portable computer (one per user), home (one per user) and application servers (one per application)[3]. The home and the portable computer are connected by a dynamically changing set of network connections with diverse QoS - each with a wireless and wired component. The home and application servers are connected via the backbone network, and application servers are oblivious of user mobility. The home interacts with the application server on behalf of the portable. The home and the portable together provide the functionality required by the goals in Section 3.

Within the above framework, a number of research challenges need to be addressed. We classify them into three broad areas:

- application structure
- seamless mobility and management of multiple connections

[2]Henceforth, the term base station generically refers to the backbone agent which supports mobility in a network.

[3]In practice, an application server may serve several applications, and a home computer may be a dedicated computer from which mobile users can lease home service.

- adaptive computing and consistency management

4.1 Application Structure

Applications need to adapt to the dynamic network conditions because the mobile computing environment has no knowledge of the application semantics. While the environment can perform some application-independent adaptation (such as compression, batching writes, etc.), semantics-dependent adaptation - such as deciding which part of the data is critical and needs to be kept consistent over a low QoS network - needs to be performed by the application. The application structure thus depends on the type of adaptation support provided by the environment to applications. In particular, two issues are of interest: (a) whether the environment notifies the applications of QoS changes, and (b) whether the environment provides support for notification handling and dynamic QoS negotiation.

4.1.1 Transparency versus Notification

The major source of problems in our mobile computing environment is the dynamics of the network connections. Due to user mobility, connections may be set-up/torn-down, typically accompanied with QoS changes. Even within the same network, mobility may cause a user to move from uncongested cell to a congested cell. The issue is whether, and how, to provide a framework for applications to react to the dynamic network QoS. There are three broad approaches to this problem:

1. Applications are provided with a seamless mobile computing environment without notifications of QoS change. This is consistent with a purely networking solution to seamless cross-network migration, and does not work well when QoS changes by orders of magnitude (e.g. indoor to outdoor mobility).

2. Applications are allowed to dynamically (re)negotiate QoS with the network. If the pre-negotiated QoS is violated by the network, the application is notified. The onus of QoS negotiation and reaction to notifications is on the applications. Several emerging adaptive computing solutions fit into this approach[31, 39].

3. Applications specify a sequence of acceptable QoS classes, the procedure to execute if a QoS class is granted, and the exception handling policy to execute if the network violates its QoS contract. The mobile computing environment handles the dynamic QoS (re)negotiation and reaction to notification. If the network notifies the application of a failure to deliver a pre-negotiated QoS class, the system executes the exception procedure and then renegotiates a lower acceptable QoS class from the list. PRAYER follows this approach, as described in Section 5.

4.1.2 Adaptation Support

The application structure will depend on the level of adaptation support provided by the mobile computing environment. In case the QoS changes are transparent to the application, no special support needs to be provided for application-level adaptation. However, 'aware' applications will adapt better to the dynamics of the environment [39]. Support for QoS-awareness in

mobile computing environments is a very complex task. While there are systems which provide application-level reaction to measured coarse-grain QoS, we are not aware of a working system which supports dynamic QoS (re)negotiation by the application. Even application-level support for reaction to QoS-notification (upon migration of networks, for example) is a challenging task. There are three broad approaches:

1. The mobile computing environment provides the measured QoS in a structure which can be retrieved by applications. Applications essentially poll the environment periodically in order to retrieve current QoS value and then adapt accordingly.

2. The mobile computing environment provides the measured QoS in a structure which can be retrieved by applications. In addition, an application registers with the environment and specifies acceptable QoS bounds. If the QoS bounds are violated, the application is notified, and may then adapt accordingly [31].

3. The application program is split into ranges separated by QoS system calls. In each call, the application specifies a sequence of acceptable QoS classes, the procedure to execute if each class is satisfied, and the exception handling if the measured QoS class changes. In this case, notification is handled by the exception handling procedure, which may pursue one of four actions: block, abort, rollback, or continue. Section 5 describes this approach further.

.

4.2 Seamless Mobility and Management of Multiple Connections

At any time, a portable computer may have access to multiple wireless networks. The appropriate choice for the wireless network for data transport depends on the application traffic characteristics, wireless network characteristics, priority of transmission, and cost (making this trade-off is in itself a challenging task). Migration between networks may be induced by a variety of reasons, such as user mobility, network failure/partition, or application-dependent trade-offs. Seamless mobility requires that when connections are broken or established, the process of switching between networks should be done transparently (with possible QoS notifications).

An important distinction between the considerations of seamless mobility in this paper and related work [4, 26] is that we address the issue of providing seamless mobility over autonomously owned and operated, possibly non-interoperable wireless networks. This imposes the constraint of not being able to access or modify base station software.

There are four levels of mobility, as described below.

1. Handoff within the organization and the network: a mobile user moves between two cells of the same service-provider within a network (e.g. handoff between two RAM cells).

2. Handoff between organizations but within the same network: a mobile user switches between service-providers on the same network (e.g. switches carriers while using the same cellular phone line).

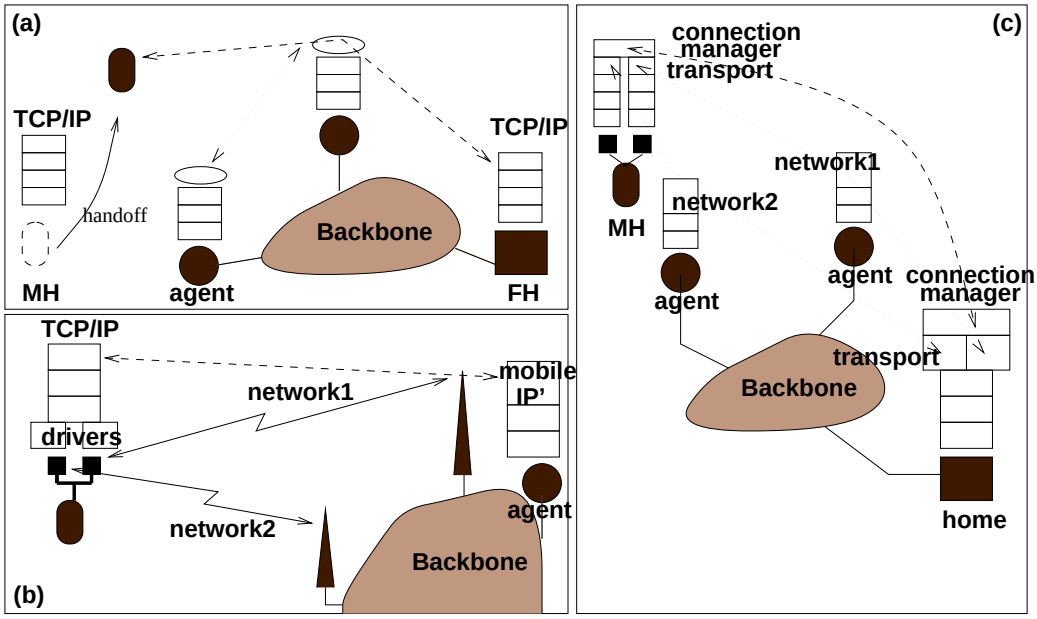

Figure 3: Network Support for Seamless Mobility

3. Migration within the organization but between networks: a mobile user switches between networks of the same service-provider (e.g. switches between cellular phone and CDPD of same carrier).

4. Migration between organizations and networks: a mobile user switches both networks and service-provider (e.g. switches between locally owned WLAN to RAM).

Handoff between the cells of the same network and organization is the common case, and has been solved at the network layer [25]. Handoff between cells of different organizations introduces the problem of authentication and accounting [9]. Handoff across the cells of different networks and organizations is the general case, and is addressed in this paper. In the general case, the following constraints apply: (a) a mobile user may migrate between different networks which are owned and operated autonomously, (b) the networks may use different protocol stacks (typically the lower 3 layers of the stack), (c) mobile service stations for the different networks will not communicate with each other directly for mobility support, and (d) each network will require a unique address for the computer.

Several important issues arise in this case: (a) how to structure the network, (b) when to migrate between networks and (c) how to make application-related trade-offs. Authentication, accounting and security in inter-network mobility are other major issues, but are not discussed in this paper. A discussion of these issues, and a preliminary solution are proposed in [9].

4.2.1 Network Structure

Figure 3 shows alternative network structures for handling mobility.

Figure 3.a shows the standard Mobile-IP structure which supports handoffs between two cells of the same network. Agents provide backbone support for mobility. Performance enhancements may include multicasting between the old and new base station in order to reduce handoff latency [10], and snoop caches in base stations in order to eliminate transport layer

retransmissions [8]. This network structure uses a single network address and only considers handoff between cells of the same network.

Figure 3.b shows a network structure which supports inter-network mobility at the network layer by enhancements through the Mobile-IP protocol. It uses different drivers for the different networks, and provides a network level solution to the mobility problem [5]. It is still possible to provide some notion of QoS, by having the network layer measure the QoS over each network and then propagate it to higher layers. Likewise, application-level trade-offs and choice of the wireless network can be achieved by informing the higher layers of the wireless networks currently accessible to the portable computer. We are not, however, aware of any implementation which provides adaptation support over a network level solution for seamless mobility across different wireless networks. While network level solutions may be the eventual goal for seamless mobility, they do involve making changes at the base station and cooperation between the different networks. In the current scenario, providing seamless mobility over autonomously owned and operated wireless networks is not possible using this approach.

Figure 3.c shows a network structure which handles inter-network mobility at the transport layer and connection manager in the context of the PRAYER model. The connection manager maintains individual connections for each network, and can make the choice of network for data transport. The connection manager can also measure the end-to-end QoS parameters for each network between the portable and the home. The advantage of this architecture is that the choice of the wireless network for data transport is made outside the network. Thus seamless mobility is provided without the necessity of software support at the base stations. There are two advantages in not requiring base stations to different networks to co-operate in order to provide seamless mobility the networks: (a) many autonomous networks follow their own protocol standards, and (b) authentication and trust between the base stations of different organizations is avoided. The disadvantage is performance degradation. PRAYER uses a similar solution, but it merges the transport layer and connection manager.

4.2.2 Choice of Wireless Network for Data Transport

Wireless networking resources being scarce and expensive, the choice of wireless network for data transport can significantly affect the performance and cost of an application. Typical network related trade-offs include bandwidth, delay, medium access patterns, security, and pricing structure. Primarily due to bandwidth considerations, the choice of networks is typically wire, indoor wireless and outdoor wireless in descending order of preference. The interesting trade-offs happen in the outdoor wireless networks, where access patterns and pricing structure play important roles. For example, CDPD and RAM support packetized data transport and are susceptible to much larger bursts of throughput, delay and jitter than cellular modems, which have periodic medium access patterns. If charged by data size, short packet bursts can cost up to an order of magnitude more in cellular modems as compared to RAM, while large data transmissions can cost an order of magnitude more in RAM than cellular modems. However, several packet data wireless providers also support monthly rates, in which case the pricing trade-off is irrelevant. Security is another important issue. For example, cellular modems are insecure while CDPD offers 6 levels of security.

We are not aware of any good solution to the problem of selecting a medium for transport based on an application-directed tradeoff. In PRAYER, the current approach is to pre-specify a descending order of network choices based on raw data rates, and try to satisfy the application

connection request through the 'best' available network. Clearly, this is inadequate, because the dynamically available data rate may be significantly lower than the raw data rate. Besides, bandwidth is an important, but by no means only criterion for network selection.

4.3 Adaptive Computing and Consistency Management

Seamless mobility across different wireless networks is typically accompanied with a change in QoS. For example, mobility from indoor to outdoor wireless networks may result in a bandwidth decrease by two orders of magnitude. In order to provide a graceful degradation of the operating environment, mechanisms for system level and application level adaptation are necessary. This section explores the issues in caching and consistency management for adaptive mobile computing environments.

Caching data and meta-data at the portable computer reduces access time and offered wireless traffic, but introduces a related problem of consistency when multiple copies of shared data are maintained. In mobile computing environments, disconnection is always a possibility. Thus most approaches to caching 'hoard' data aggressively, and allow the mobile user to manipulate the cached copy at the portable when disconnected. The modified data is reintegrated with the server copy upon reconnection, and update conflicts are typically reconciled by human intervention in the worst case [27]. This approach is suitable for disconnected operation on mostly private data. Given the increasing availability of wide area wireless connectivity and also the potential for collaborative applications in mobile computing, there arise four scenarios for caching and consistency management of data.

1. Disconnected operation on private data
2. Disconnected operation on shared data
3. Partially connected operation on private data
4. Partially connected operation on shared data

In the above classification, indoor and outdoor wireless network connectivity are termed as 'partially connected' because (a) disconnections are possible at any time, (b) network errors are orders of magnitude higher than on wire, and (c) the significant cost associated with data transmission over the scarce resource may induce voluntary intermittent connectivity on the part of the mobile user.

Distributed file systems which support disconnected operation typically assume that most of the data is private and unshared. The general approach in this case is to hoard data while in connected mode [27, 42]. Just prior to voluntary disconnection by the user, explicit user-directed hoarding is allowed. Once disconnected, the user is allowed to access and update the hoarded local copy, and all update operations are logged. Upon reconnection, the hoarded files are checked into the server. Update conflicts are resolved by log replay and in the worst case, user intervention [27, 34].

While file systems assume that most of the files are private (user data) or shared read-only (program binaries), this is not true of other data repositories such as databases. In such cases, update conflicts upon reconnection cannot be assumed to happen rarely, and automatic mechanisms for update conflict resolution need to be provided [15]. The possibility of conflict resolution voiding a previously concluded transaction also gives rise to the notion of provisional

and committed transactions. Thus distributed databases which support disconnected operation must also support multiple levels of reads and writes.

Related work has often assumed the two extreme modes of operation - connection or disconnection. The emergence of wide area wireless networks provides intermediate modes of connection - where communication is expensive, but possible. Partial connection is particularly useful in two situations in the context of data management: (a) when files which were not hoarded in the connected mode are required, and (b) when certain parts of the application data need to be kept consistent with the data on the backbone server. In an environment which supports seamless mobility over heterogeneous wireless networks, the QoS may change dynamically. Thus the caching and consistency policies need to adapt to change in the network QoS in order to efficiently exploit the benefits of partial connectivity without incurring a significant communication cost.

In our model, the home retains the 'true' copy of the cached data at any time. Since the home is in a connected mode on the backbone, it can execute any application dependent consistency policy on the backbone. Essentially, the onus of keeping its cached data consistent with the home is on the portable computer. This approach is at variance with contemporary schemes such as Coda [27] or Bayou [15], which do not have an intermediate home computer. The advantage of having a home that is known to maintain the true version of the data is twofold: (a) the consistency management in the mobile computing environment is now restricted to two known endpoints connected by a variable QoS network, and (b) distributed applications which support different types of consistency policies on the backbone can be supported, since the applications on the backbone need only care about keeping their data consistent with the home computer. The disadvantage of this model is poor availability - as far as the portable is concerned, the home is the only server for its cached data.

Within this framework, a number of caching and consistency issues arise: (a) what data should be hoarded, (b) how consistency will be maintained, and (c) how applications will interact with the mobile computing environment in order to adapt the consistency management policies upon dynamic QoS changes. These issues are discussed below in the context of file system support for partially connected operation.

4.3.1 Hoarding Policy

What to hoard is a non-trivial question in mobile computing. The factors involved are: (a) the nature of the data - ownership, mutability, and level of consistency, (b) currently available network QoS, (c) portable computer characteristics - available disk and battery power, and (d) predicted future connection or disconnection.

Privately owned data or read-only data can be cached without involving any communication overhead for consistency management. Cached shared read/write data may involve high communication overhead for consistency management, depending on the type of consistency guarantees provided on shared data. Data which is loosely consistent or data which is not modified often can be cached with low overhead. Based on the these observations, a user who voluntarily plans on initiating a disconnection or migration to a low QoS network (e.g. indoor to outdoor) may choose to hoard private and read-only files, and flush the dirty cached data for shared read/write files. Most of the caching decisions mentioned above are highly application dependent. Ideally, the mobile computing environment will be smart enough to provide some assistance in predictive caching [28, 42]. For example, a request for caching an application

binary will also cache the files it has frequently accessed during its previous executions (e.g. resource files) [42].

The current approach in PRAYER is simplistic - caching files which have been accessed in the recent past, and allowing the user to explicitly select files for caching. The fact that partial connectedness (as opposed to disconnectedness) is the common mode of operation reduces the negative impact of such a simple predictive caching approach.

4.3.2 Caching Granularity

The tradeoff in caching granularity is that a large grain size may induce false sharing while a small grain size may induce higher processing overhead at the portable and the home. There are three broad alternatives for the grain size of caching:

1. Whole file caching: The whole file is cached upon file open. In most current distributed file systems which support disconnected operation [20, 23, 27], whole files are cached during connection.

2. Block caching: Caching is at the granularity of file system blocks. A variation of block-size caching is to have applications vary the block size, depending on the available network QoS.

3. Semantic record caching: An application imposes a semantic structure on its files. The semantic structure is contained in a pre-defined template, which specifies record and field formats in terms of regular expressions. The application is allowed to specify (by means of a file system interface) the cache size to be per-record level or per-field level. The advantage of this scheme is that it allows an 'aware' application to adapt its caching granularity to both the network QoS and application-semantics. The disadvantage is the increased complexity in caching/consistency management. The PRAYER caching approach implements a variant of this approach.

An important point to note in both application-defined block level caching and semantic record caching is that the caching/consistency is being done here between the home and the portable. In the absence of a home, different portable clients may have different cache granularities, which will make providing consistency management incredibly hard for the distributed file system. In our model, the distributed file system provides whatever consistency policy it may choose to, with respect to the home. The block and semantic record caching schemes refer to the ways the portable keeps itself consistent with the home.

4.3.3 Consistency Management

One of the advantages of partial connectedness is the option of providing a variable level of consistency on a whole file or parts of it. File systems which support disconnected operation must inevitably provide some form of session semantics, wherein a disconnection period is treated as a session. Since disconnection is a special case of partially connected operation, the consistency management for partially connected operation must reduce to session semantics in the event of disconnection.

PRAYER supports semantic record caching and consistency. An aware application opens a file and imposes a template structure on it. The open file is thus treated as a sequence of

(possibly multi-level) records. An application can specify certain fields in all records, or certain records in the file to be kept consistent with the home. For each cached data element, there are two possible consistency options: *reintegrate* and *invalidate*. Reintegration keeps data consistent, but requires communication between the home and the portable in order to propagate updates. Invalidation tolerates inconsistencies, but requires no communication. Depending on the available QoS, the application can dynamically choose to reintegrate or invalidate each record or field. Note, that invalidate still allows the application to access the local copy, but with no guarantees on consistency.

In connected mode, the whole file is kept in the reintegration mode (between the portable and the home). In disconnected mode, the whole file is kept in the invalidation mode. In partially connected mode, the application has the flexibility to keep critical fields or records consistent while accepting inconsistencies for the rest of the file.

In addition to maintaining consistency on certain parts of the file, there also needs to be support for explicit consistent reads and writes. PRAYER supports two types of reads: *local read* and *consistent read*. A local read is the default operation, and reads the local copy. A consistent read checks the consistency between the portable and home, and retrieves the copy from the home if the two copies are inconsistent. PRAYER supports three types of writes: *local write*, *deferred write* and *consistent write*. A local write updates only the local copy. A deferred write batches write updates. A consistent write flushes the write to the home. In disconnected mode, consistent read and write return errors.

Two simple examples illustrate the operation of the application-directed adaptive consistency management.

- *Calendar*: If a user goes on a trip and maintains a distributed calendar with the secretary, the user would keep the 'time' and 'place' fields of appointments in reintegration mode, but the 'content' field in invalidation mode. This will enable the user and the secretary to prevent scheduling conflicts, though the content fields of the appointments may not be consistent.

- *Email*: If a user goes on a trip, the email application could keep the 'sender' and 'subject' fields in reintegration mode, but the 'content' field in invalidation mode. If the user wants to read a particular email, an explicit 'consistent read' will be issued in order to access the contents.

5 The PRAYER Mobile Computing Environment

Seamless mobility across diverse indoor and outdoor wireless networks is a very desirable goal, since it will enable a user to operate in a uniform mobile computing environment anytime, anywhere. However, lack of inter-operability standards pose a significant challenge in building such an environment. Even if seamless mobility is provided across different networks, the wide variation in bandwidth and other QoS parameters imply that the systems software and application will have to collaboratively adapt to the dynamic operating conditions in order to gracefully react to inter-network migration. Simple, yet effective mechanisms need to be provided to applications for adaptive computing.

In order to explore the challenges in building a uniform operating environment which provides adaptive computing and seamless mobility on top of commercially available wireless net-

works, we are building the PRAYER mobile computing environment. A preliminary PRAYER prototype has been operational for six months, and has served as a platform to test some of our solutions. While the focus of this paper has been to identify the major challenges and discuss possible solutions for providing seamless mobility and adaptive computing, a brief discussion of the PRAYER environment will serve to provide an overall context for our approach.

There are three important components in PRAYER: connection management, data management, and adaptation management. Connection management provides seamless mobility over multiple wireless networks, and provides the abstraction of a virtual point-to-point connection between the portable computer and the home computer. Data management provides filtered information access on top of a file system which implements application-directed caching and consistency policies. Adaptation management provides language and systems support to applications for dynamic QoS (re)negotiation and reaction to notifications by the network (though at this point, we do not perform end-to-end QoS negotiation in the network).

5.0.4 Seamless Mobility across Multiple Wireless Networks

We adapt TCP/IP in order to provide support for a virtual point-to-point connection over multiple wireless networks between the portable computer and the home computer. The portable computer may have different IP addresses, corresponding to the different networks. Each TCP virtual connection is identified by a 4-tuple consisting of a logical IP address for the portable, a port at the portable, the IP address of the home, and a port at the home. The logical IP address for the portable for a TCP connection is the IP address of the portable corresponding to the wireless network over which the connection is first set up. Thus, the logical IP address, which serves to identify the connection, can be different from the IP address of the wireless network over which the portable actually transmits the packets.

All TCP connections to or from the portable pass through the home if seamless mobility over multiple wireless networks is desired. The home is bypassed if a TCP connection over a single wireless network is desired. In order to establish a virtual connection between the home and the portable, the application pre-specifies the sequence of acceptable wireless networks in a descending order of priority. When a connection request is initiated at the home or the portable, the networks are polled for access in the descending order of priority. We define a socket interface to applications for setting up virtual connections over multiple wireless networks. The details of the implementation and the programming interface are described in [18].

5.0.5 Caching and Consistency Management in the File System

The file system uses the home as the server and the portable as the client, and provides strong consistency on application-specified portions of files cached at the client (note that the home itself may be an NFS client). The key feature of the file system is the support for application-directed caching and consistency policies.

When an application opens a file at the portable, it imposes a template on the open file. Basically, a template specifies the semantic structure of the file. For example, a mailbox is a sequence of mail records, where a mail record has some pre-defined fields such as sender, subject, content, etc. It is possible for a template to specify records with variable length fields, optional fields, or fields appearing in different orders (all of which occur in the mailbox template). Once the application imposes a semantic structure for a file, the file system at

the home and the portable create a sequence of objects for the file, each object representing a record. The application may then specify a subset of the fields of every record, or a subset of the records of the file to be kept consistent with the home through a `pconsistency()` system call, which causes these fields/records to be marked at both the portable and the home. Whenever a marked data element is updated at either the home or the portable, it is propagated to the other entity.

In addition to requiring consistency on parts of the file, the application may also perform explicit consistent reads and writes (through the `pread()` and `pwrite()` system calls, which basically read/write through the home if the data at the portable is not consistent.

The goal of the PRAYER file system is to facilitate adaptive application-directed consistency policies, while shielding the application from the mechanisms of keeping parts of the file consistent. When the portable is fully connected, the whole file may be kept consistent with the home (i.e. reintegration mode). When the portable disconnects, the whole file now operates in cached-only (invalidation) mode. When the portable has connectivity through wide area wireless networks and communication is expensive, only the critical parts of the file are kept consistent, and the rest of the file is accessed in a cached-only mode. Supporting this level of adaptation has a definite penalty in terms of file system performance, though we are yet to perform a quantitative evaluation of the overhead. A detailed description of the file system design and implementation is available in [16].

5.0.6 Application Support for Adaptation

We provide simple language and OS support for modifying existing applications or building new applications in our adaptive mobile computing environment. We classify QoS requests into commonly used *QoS classes*. A program is divided into *regions*, and may explicitly initiate QoS re-negotiation between the regions. Within a region, the application expects the network to provide a fixed QoS class. If the network is unable to do so, it notifies the application, which causes pre-specified exception handling procedures to be executed (as described below).

Exceptions are handled in one of four ways: *best effort*, *block*, *abort* or *rollback*. 'Best effort' ignores the notification and continues with the task. 'Block' suspends the application till the desired QoS class is available. 'Abort' aborts the rest of the task within the region and moves to the next region. 'Rollback' aborts the rest of the task, and reinitiates QoS negotiation within the same region. Ideally, we would like rollback to also undo the actions taken thus far in the region before re-negotiation.

The QoS negotiation is performed by a system call `getQoS()`, which takes in a sequence of `options`. Each option is a 3-tuple, consisting of the desired QoS class, the procedure to execute if that class is granted, and the exception handling policy.

At the start of a region, the `getQoS()` call measures the network QoS, and returns the highest desired QoS class it could satisfy. When a QoS class is granted, the corresponding procedure is executed. If during the execution of the procedure, the network is unable to sustain the QoS class, then the exception handling routine is invoked with the application-defined policy. In this framework, both QoS negotiation and notification handling are supported by the system; the application need only specify the policy, and not bother about the mechanism in order to achieve adaptation.

We expect applications to use the adaptation support and consistency management policy in concert. We expect that an application will initially open a file and impose a template structure

on it. Then, depending on the QoS class granted through the `getQoS()` call, the application can change the fields/records it keeps consistent through the `pconsistency()` call. While we have not yet written a large application in this environment, we expect that the mechanisms for adaptation are simple, yet sufficiently powerful to support adaptive computing.

6 Related Work

Mobile computing has witnessed a rapid evolution in the recent past, both in industry and academia. However, related work on seamless mobility and adaptive computing support for applications has just begun to emerge. In this section, we provide an overview of contemporary work in seamless mobility, adaptive computing, consistency management, and disconnected operation. We also identify key work in related areas which motivated several of the design decisions in PRAYER.

Most of the projects which provide seamless mobility across heterogeneous wireless networks provide a network level solution. While this is a desirable goal, it will require mobile service stations of different networks to interact and trust each other. MosquitoNet [4, 5] and Barwan [26] address seamless mobility issues as described above.

Wit [43, 46] addresses data consistency in variable QoS environments. For high-QoS, caching/consistency is as in backbone networks. In low-QoS, nothing is cached. In variable QoS, there is a two level caching scheme: on the backbone between homes, and on the wireless network between the home and the portable. Bayou [15, 43, 44] provides a replicated weakly-consistent distributed database to support shared data-driven mobile applications and supports per-application consistency. Bayou deals with small-to-medium database applications (calendars, etc) since it assumes that a large part of the database may be cached at the mobile. Odyssey [31, 36] provides a framework for adaptive applications to react to QoS changes. The applications can specify QoS bounds to the network. If the network is unable to satisfy these bounds, it notifies the applications, which can then adapt to the dynamic QoS change.

Coda [27] provides disconnected access of file systems. When the portable is in connected mode, it hoards files by periodic 'hoard walks'. Upon disconnection, the portable accesses the cached files, and logs the updates. Upon reconnection, it checks the mutated cached files for potential conflict, which is then resolved by the user. Disconnected AFS [23] preserves the AFS semantics for disconnected operation. A user explicitly disconnects from the network, upon which the callbacks are retrieved by the server. Seer [28] and MFS [42] propose sophisticated hoarding mechanisms for disconnected operation.

PRAYER uses several ideas from current and past related work, which are mentioned below.

I-TCP [6] (intermediate host), Daedelus [8] (snoop cache) and [11] provide approaches for efficient wireless TCP. NFS [35], Sprite [30] and Andrew [21] provided distributed file systems caching approaches, which may be used for backbone consistency. Distributed databases establish consistency among replicated data by clustering [33], tokens [24] and partitioning [14]. [48] discusses fundamental issues in ubiquitous and mobile computing. [45] highlights OS issues in mobile computing.

7 Conclusions

The ideal scenario in mobile computing is when a user equipped with a portable computer with multiple wireless interfaces roams around between different indoor and outdoor networks while operating in a seamless computing environment which gracefully adapts to the dynamically changing quality of service. In order to achieve this scenario, at least two important components need to be satisfied: *seamless mobility*, and *adaptive computing*.

The emergence of several viable wireless networking technologies with different standards, networking architectures and protocol stacks makes the problem of seamless mobility across different wireless networks a very challenging task. Contemporary research typically proposes network level solutions to this problem. Such a solution, though scientifically desirable, poses some serious problems because the commercial networks cannot inter-operate. This paper identifies the challenges in providing seamless mobility on top of commercial networks.

Seamless mobility across wireless networks is typically accompanied with a change in QoS. In order to react gracefully to the change in QoS, both the mobile computing environment and the application need to collaboratively adapt to the dynamic operating conditions. Adaptation to QoS changes introduces challenges in data management and caching/consistency. It is the application which can best make semantics-based decisions on adaptation. However, burdening the application with dynamic QoS negotiation and reaction to network notifications will complicate application logic, and lead to an unviable environment to develop real-world programs. This paper identifies the challenges in language and operating systems support for applications to interact with the underlying file system.

With the increasing popularity for mobile computing, the importance of a uniform mobile computing environment providing seamless mobility across commercial wireless networks and graceful adaptation to dynamic operating conditions cannot be over emphasized. However, there are significant challenges which need to be overcome before such an environment can be effectively deployed. This paper explores the issues and offers some preliminary solutions, which are being implemented in the PRAYER mobile computing environment.

Acknowledgements

I am grateful to Dane Dwyer for providing multiple reviews of this paper and for implementing the PRAYER file system.

References

[1] B.R. Badrinath, A. Acharya, and T. Imielinski. *Structuring Distributed Algorithms for Mobile Hosts*, **The 14th International Conference on Distributed Computing Systems**, June, 1994.

[2] B.R. Badrinath, and T. Imielinski *Replication and Mobility*, **The 2nd International Workshop on Management of Replicated Data**, November, 1992.

[3] B.R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz *Handling Mobile Clients: A Case for Indirect Interaction*, **The 4th Workshop on Workstation Operating Systems**, October, 1993.

[4] M.G. Baker. *Changing Communication Environmeents in MosquitoNet*, **The 1994 IEEE Workshop on Mobile Computing Systems and Applications**, December, 1994.

[5] M. G. Baker, X. Zhao, S. Cheshire, and J. Stone. *Supporting Mobility in MosquitoNet* **Proceedings of the 1996 USENIX Technical Conference**, January 1996.

[6] A. Bakre, and B.R. Badrinath. *I-TCP: Indirect TCP for Mobile Hosts*, **Technical Report**, Rutgers University, October, 1994.

[7] A. Bakre, and B.R. Badrinath. *Handoff and System Support for Indirect TCP/IP*, **Proceedings of the Second USENIX Mobile and Location- Independent Computing Symposium**, Ann Arbor, MI, April, 1995.

[8] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. *Improving TCP/IP Performance over Wireless Networks*, **To appear in the Proceedings of the 1st ACM International Conference on Mobile Computing and Networking (Mobicom)** November, 1995.

[9] V. Bharghavan. *A Protocol for Authentication, Data and Location Privacy, and Accounting in Mobile Computing Environments*, an earlier version appeared in the **Proceedings of the ACM Conference on Computers and Communications Security**, Fairfax, Virginia, November 1994.

[10] R. Caceres, and L. Iftode. *The Effects of Mobility on Reliable Transport Protocols*, **Proceedings of the 14th Internetional Conference on Distributed Computing Systems**, June, 1994.

[11] R. Caceres, and L. Iftode. *Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments*, **IEEE Journal on Selected Areas in Communication**, Volume 13, Number 5, June, 1994.

[12] S. Cheshire, and M. Baker. *Experiences with a Wireless Network in MosquitoNet*, **IEEE Hot Interconnects Symposium '95**, August, 1995.

[13] D. Cox. *Wireless Network Access for Personal Communications*, **IEEE Communications Magazine**, Volume 30, Number 12, December 1992.

[14] S.B. Davidson, H. Garcia-Molina, and D. Skeen. *Consistency in Patitioned Networks*, **ACM Computing Surveys**, Volume 17, Number 3, September, 1985.

[15] A. Demers, K. Petersen, M. Spreitzer, D. Terry, et al. *The Bayou Architecture: Support for Data Sharing Among Mobile Users*, **Proceedings of the Workshop on Mobile Computing Systems and Applications**, Santa Cruz, California, December 1994.

[16] D. Dwyer and V. Bharghavan. *Mobile File Systems for Partially Connected Operation, Internal Technical Report*, TIMELY Research Group, University of Illinois, April 1996.

[17] G.H. Forman and J. Zahorjan. *The Challenges of Mobile Computing*, **IEEE Computer**, Volume 27, Number 6, April, 1994.

[18] V. Gupta and V. Bharghavan. *Seamless Mobility across Commercial Wireless Networks*, **Internal Technical Report**, TIMELY Research Group, University of Illinois, April 1996.

[19] F. Guterl. *Smart Networks*, **IBM Research Magazine**, Number 4, 1994.

[20] J.S. Heidemann, T.W. Page, R.G. Guy, and G.J. Popek. *Primarily Disconnected Operation: Experience with Ficus*, **The 2nd International Workshop on Management of Replicated Data**, November, 1992.

[21] J.H. Howard, M.L. Kazar, S.G. Menees, D.A. Nichols, et al. *Scale and Performance in a Distributed File System*, **ACM Transactions on Computer Systems**, Volume 6, Number 1, February 1988.

[22] Y. Huang, P. Sistla, and O. Wolfson. *Data Replication for Mobile Computers*, **Proceedings of the 1994 SIGMOD Conference**, May, 1994.

[23] L.B. Huston, and P. Honeyman. *Disconnected Operation for AFS*, **Proceedings of the USENIX Mobile and Location- Independent Computing Symposium**, August, 1993.

[24] T. Imielinski, and B.R. Badrinath. *Data Management for Mobile Computing*, **SIGMOD Record**, Volume 22, Number 1, March, 1993.

[25] J. Ioannidis, D. Duchamp, and G.Q. Maguire Jr.. *IP-based Protocols for Mobile Internetworking*, **Proceedings of the SIGCOMM Conference on Communications Architectures and protocols**, September, 1991.

[26] Randy H. Katz, Eric A. Brewer, Elan Amir, Hari Balakrishnan et al. *The Bay Area Research Wireless Access Network(BARWAN)*, **Proceedings of the Spring COMPCON Conference**, 1996.

[27] J. Kistler, and M. Satyanarayanan. *Disconnected Operating in the Coda File System*, **Proceedings of the 13th SOSP**, October, 1991.

[28] G.H. Kuenning. *The Design of the SEER Predictive Caching System*, **Proceedings of the Workshop on Mobile Computing Systems and Applications**, Santa Cruz, California, December 1994.

[29] L. Mummert, and M. Satyanarayanan. *Large Granularity Cache Coherence for Intermittent Connectivity*, **Proceedings of the 1994 Summer USENIX Conference**, June, 1994.

[30] M. Nelson, B. Welch, and J. Ousterhout. *Caching in the Sprite Network Filesystem*, **ACM Transactions on Computer Systems**, Volume 6, Number 1, February 1988.

[31] B. Noble, M. Price, and M. Satyanarayanan. *A Programming Interface for Application-Aware Adaptation in Mobile Computing*, **Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing**, Ann Arbor, Michigan, April 1995.

[32] B.D. Noble, and M. Satyanarayanan. *An Empirical Study of a Highly Available File System*, **Performance Evaluation Review**, Volume 22, Number 1, May, 1994.

[33] E. Pitoura, and B. Bhargava. *Maintaining Consistency of Data in Mobile Distributed Environments*, **Proceedings of the 15th International Conference on Distributed Computing Systems**, May, 1995.

[34] P. Reiher. *Resolving File Conflicts in the Ficus File System for a Distributed Workstation Envirоonment*, **Proceedings of the 1994 Summer USENIX Conference**, June, 1994.

[35] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, et al. *Design and Implementation of the Sun Network Filesystem*, **Proceedings of the USENIX Summer Conference**, Portland, Oregon, June 1985.

[36] M. Satyanarayanan, B. Noble, P. Kumar, and M. Price. *Application-Aware Adaptation for Mobile Computing*, **Proceedings of the 6th ACM SIGOPS European Workshop** Dagstuhl, Germany, September 1994.

[37] M. Satyanarayanan, J.J. Kistler, L.B. Mummert, M.R. Ebling, P. Kumar, and Q. Lu. *Experience with Disconnected Operation in a Mobile Computing Environment*, **Proceedings of the USENIX Mobile and Location- Independent Computing Symposium**, August, 1993.

[38] B. Schilit, M. Theimer, and B. Welch. *Customizing Mobile Applications*, **Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing**, August, 1993.

[39] B. Schilit, R. Want, and N. Adams. *Context Aware Computing Applications*, **Proceedings of the Workshop on Mobile Computing Systems and Applications**, Santa Cruz, California, December 1994.

[40] P. Schwartz, A. Luniewski, K. Shoens, J. Stamos, et al. *Information Organization using Rufus*, **Proceedings of the ACM SIGMOD Conference on Management of Data**, Washington DC, May 1993.

[41] C.D. Tait, and D. Duchamp. *Service Interface and Replica Management Algorithm for Mobile File System Clients*, **The First International Conference on Parallel and Distributed Information Systems**, 1991.

[42] C.D. Tait, H. Lei, S. Acharya, and H. Chang. *Intelligent File Hoarding for Mobile Computers*, **Proceedings of the ACM Conference on Mobile Computing and Networking**, Berkeley, California, November 1995.

[43] D.B. Terry, M.M. Theimer, K. Petersen, Alan J. Demers, et al. *Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System*, **Proceedings of the ACM Symposium on Operating Systems Principles**, Copper Mountain Resort, Colorado, December 1995.

[44] D. Terry, A. Demer, K. Petersen, M. Spreitzer, M. Theimer, and B. Welch. *Session Guarantees for Weakly Consistent Replicated Data*, **Proceedings of the Third International Conference on Parallel and Distributed Information Systems**, September, 1994.

[45] M. Theimer, A. Demers, and B. Welch. *Operating System Issues for PDA's*, **Proceedings of the Fourth Workshop on Workstation Operating Systems**, October, 1993.

[46] T. Watson. *Effective Wireless Communication through Application Partitioning*, **Proceedings of the Fifth Workshop on Hot Topics in Operating Systems**, 1994.

[47] T. Watson. *Application Design for Wireless Computing*, **Proceedings of the Workshop on Mobile Computing Systems and Applications**, Santa Cruz, California, December 1994.

[48] M. Weiser. *Some Computer Science Issues in Ubitquitous Computing*, **Communications of the ACM**, July, 1993.