COMPUTATIONAL MODELS FOR WIRELESS AND MOBILE ENVIRONMENTS

G. SAMARAS, E. PITOURA

9-98

Preprint no. 9-98/1998

Department of Computer Science
University of Ioannina
451 10 Ioannina, Greece

Computational Models for Wireless and Mobile Environments

George Samaras*
Evangelia Pitoura+

*Department of Computer Science, University of Cyprus
CY-1678 Nicosia, Cyprus, <csamara@turing.cs.ucy.ac.cy>

+ Department of Computer Science, University of Ioannina
GR 45110, Ioannina, Greece, <pitoura@cs.uoi.ge>

Abstract

The mobile computing environment is constrained in many ways. Mobile elements themselves are resource-poor and unreliable. Their network connectivity is often achieved through low-bandwidth wireless links. Furthermore, connectivity is frequently lost for variant periods of time. The difficulties raised by these constraints are compounded by mobility that induces variability in the availability of both communication and computational resources. These severe restrictions have a great impact on the design and structure of mobile computing applications and motivate the development of new computing models. These mobile computing models must cope with the special characteristics and limitations of the mobile/wireless environment while providing efficient access to both existing and new applications which is a key requirement for the wide acceptance of mobile computing. This paper discusses the shortcomings of the wireline client/server model and presents a number of computational models aimed to specifically support, in various degrees, mobility and wireless connectivity making it thus possible to efficiently run distributed applications in wide area wireless networks. A discussion of the various computational representations of a Web Browsing application demonstrates and compares the capabilities of the different models.

Keywords: Mobile Computing, Mobile Architectures, Wireless Architectures, Mobile Agents, Peer to Peer, Client Server, Wireless Web, Transaction Processing

1. Introduction

Universal access and management of information has been one of the driving forces in the evolution of computer technology. Central computing and advances in network technology led to distributed computing. Internet and web technology went even further to provide hyper-linked information access and global computing. However, the road towards the vision of unrestricted access and the ability to compute from anywhere and any time is circumscribed by bounding access stations to physical locations. The growth, however, in mobile computing devices[1] and the emergence of wide area wireless technologies (e.g., Ardis[2] [ARDIS92], Mobitex[3] [RAM94], Cellular Digital Packet Data (CDPD) [IBM95], GSM [MoPa92], Personal Communication Service [Kob93]) offer the potential for unprecedented access to data and applications by mobile workers and paves the way towards the real universal access.

[1] e.g., Laptops, notebooks, personal digital assistances (PDAs), personal communications assistants (PCAs).
[2] Ardis is a registered service mark of the Ardis Co.
[3] Mobitex is a registered trademark of Telia Mobitel.

Mobile computing brings about a new paradigm of distributed computing in which communications may be achieved through wireless networks and users can compute even as they relocate from one support environment to another. The impact of wireless computing on system design goes beyond the networking level and directly affects data management and data computational paradigms. Infrastructure research on communication and networking is essential for realizing wireless and mobile systems. Equally important, however, is the design and implementation of data management applications for these systems, a task directly affected by the characteristics of the wireless medium, the limitations of mobile computing devices, and the resulting mobility of resources and computation.

These limitations require a more flexible computational model than the one supporting wireline communications. One of the most popular models of the wireline communication and computational paradigm is based on the client/server model [GrRe93] (see Figure 1), where the client directly communicates various requests to the server(s). There are precise and decisive assumptions influencing the definition and evolution of this model: (a) fast, reliable and cheap communications, (b) robust and resource rich devices, and finally (c) stationary and fixed locations of the participating devices.

Antithetically, wireless links have major negative characteristics that in practice render the existing client/server paradigm inadequate. Wireless networks are more expensive, offer less bandwidth, and are less reliable than wireline networks. Indeed, wireless communications face many obstacles because the surrounding environment interacts with the signal. Thus, while the growth in wired network bandwidth has been tremendous (in current technology Ethernet provides 10 Mbps, FDDI 100 Mbps and ATM 155 Mbps), products for wireless communication achieve only 19 Kbps for packet radio communications, and 9-14 Kbps for cellular telephony. Similarly, the typical bandwidth of wireless LANs ranges from 250 Kbps to 2 Mbps and it is expected to increase to 10 Mbps [Math94]. Radio transmission error rate is so high that the effective bandwidth is limited to less than 10 Kbps [Miller94]. Since the bandwidth is divided among the users sharing a cell, the deliverable bandwidth per user is even lower. Thus, bandwidth is a scarce resource. Furthermore, an additional reason that makes bandwidth consumption a major concern of mobile computing designs is that data transmission over the air is currently monetarily expensive [Hayden92]. Consequently, connectivity is weak and often intermittent since mobile elements may voluntary operate disconnected for long periods of time for reasons of cost.

Moreover, mobile elements are resource poor when compared to static elements [PiSa98]. Mobile elements must be light and small to be easily carried around. Such considerations, in conjunction with a given cost and level of technology, will keep mobile elements having less resources than static elements, including memory, screen size and disk capacity. This results in an asymmetry in mobile computing systems: fixed hosts have sufficient resources, while mobile elements are resource poor. Furthermore, since mobile elements must rely for their operation on the finite energy provided by batteries, new modes of operations are needed to sustain mobile computations. Finally, mobile elements are easier to be accidentally damaged, stolen or lost thus, arguing for more dependency on the static hosts.

The consequences of mobility itself are also numerous [PiSa98]. Having the mobile element roaming around, changing their location and therefore their point of attachment to the fixed

network, puts new requirements on system design. The center of activity, the system load and the notion of locality changes dynamically. The search cost to locate mobile elements is added to the cost of each communication involving them. Moreover, as mobile elements enter regions in which communication is provided by different means, connectivity becomes highly variant in performance and reliability. Different connectivity assumptions, ranging from high bandwidth to low bandwidth to involuntary disconnections, are now in place to further obstacles and requirements.

A collaborative effect of these characteristics and a serious limitation to a functional system is *disconnections* [PiSa98]. Disconnections are very frequent in wireless computing and thus a crucial parameter in the design of a viable wireless system. Supporting *disconnected operation*, that is allowing the mobile unit to operate even when disconnected, is a central design goal in mobile computing.

These limitations of mobility and wireless communications result in a form of distributed computing with drastically different connectivity assumptions than in traditional distributed systems. It is necessary to employ new computational paradigms to achieve a usable system. These new paradigms must cope with the special characteristics and limitations of the mobile/wireless environment. They should also provid e efficient access to both existing and new applications which is a key requirement for the wide acceptance of mobile computing. In summary, the appropriate mobile computing models must efficiently deal with:

- *disconnections*, to allow the mobile unit to operate even when disconnected
- *weak connectivity*, to aleviate the effect of low bandwith
- *both light and heavy mobile clients*, i.e., clients of different resource capacity
- *dynamic adjustment of the mobile host functionality*, i.e., the type and degree of functionality assign to mobile hosts
- *the variability of the wireless environment* which is the result of mobility
- *multiple types and application models*, including: current, and future TCP/IP applications, and emerging mobile application paradigms.

To this end, a number of models have been introduced latelly such as the *client/agent/server* [BaBIM93, FoGBA96,Ora97, ZeDu97, TeSSM96], the *client/intercept* [SaP97, HoSaL97], the *peer-to-peer*, [RePGSR96, AtDu93] and *the mobile agents model* [ChGHLPT95, White96]. This paper identifies the shortcomings of the wireline client/server model and its inabilities to support mobile computing, presents various mobile computational models and discusses and compares the degree in which each one of them supports mobility and wireless connectivity. The requirements pertaining to wireless/mobile computational models are clearly stated and the positive and negative aspects of the various models are presented in relation to these requirements. A concrete example that demonstrates the capabilities of each model is presented. Besides serving in demonstrating the characteristics of the various model, the example has its own value, since it provides various insights in building web applications for mobile computing. Defining a taxonomy of models and providing a methodology for building software for mobile computing is critical, since it provides the framework for the development of future applications and a yardstick to measure their effectiveness in addressing the challenges of mobile/wireless computing. It also provides a framework for the realization of advanced issues such as transactional support.

The rest of this paper is organized as follows. Section 2 introduces the underlying wireless architecture, basic definitions and concepts. Section 3 discusses the challenges presented by the mobile/wireless environment and the inability of the classical client/server model to deal with them. Section 4 presents the various mobile computational models and how each one of them copes with the limitations of wireless connectivity and mobility. It also provides a taxonomy and a comparison of the identified models. Section 5 presents an example of a web application that clarifies the various considerations surrounding these new mobile models by showing their applicability and limitations. Within the defined models, Section 6 presents issues on mobile transactions. Section 7 concludes the paper.

2. Wireless Architectures and their Special Characteristics

The networking infrastructure for providing ubiquitous wireless communication coverage, known by a number of different names one of which is Personal Communications Networks (PCN), is expected to be partially based on the existing digital cellular architecture (see figure 1 adapted from [ImBa94]) where the whole geographic area is divided into cells with a mobile service base station (MSS) in the center of the cell. The MSS is serving the mobile host through wireless communications and is attached to the fixed network. The mobile host can communicate with other units, mobile or static, only through the base station of the cell in which it resides. In general, the architectural configuration of a wireless network consists of fixed information networks extended with wireless network elements (the base stations) that support the mobile client units.

Cell sizes vary widely [FoZa94]. For instance, a satellite beam can cover an area more than 400 miles in diameter, a cellular transceiver has a range of few miles, and wireless LANs just cover communication in a building. With the base stations getting smaller and cheaper, as the need for an increase in capacity becomes essential, it is expected that the size of cells will decrease. The trend towards smaller and smaller cells is also justified by the requirement to support an increasing number of users via frequency reuse schemes that better utilize the limited allocated frequency spectrum.

As a mobile host (MH) moves, it may cross the boundary of a cell, and enter an area covered by a different base station. This process is called handoff. During handoff an MH may be disconnected from the network for a finite, but arbitrary period of time. Thus, to maintain the computations at the mobile host uninterrupted, handoffs must be executed in a timely fashion. While this is straightforward for voice communications, due to the higher loss of information that can be tolerated, it becomes very tricky for data transfer where the information loss must be extremely low.

The characteristics of mobile host vary. The PCN architecture must be able to support both smaller units, that are called light-weight clients, and larger more powerful units, called heavy-weight clients. A mobile host is battery depended. Receiving and sending messages as well as CPU processing and disk and memory access drain the battery that has to be recharged. Such battery power restrictions will cause mobile hosts to be grequently, voluntarilly or unwillingly, disconnected (powered off). To preserve energy, especially important for heavy weight mobile

units, the mobile host may also enter a special mode of operation called *doze mode*. While in this mode, the clock speed is reduced and no user computation is performed. A specific event or signal can later awake the "sleeping" client.

There is a need to make a distinction between mobility and wireless computing. The clarification is that mobility can exist without wireless communication by having, for example, a process surfing the net to optimize its computations. Wireless communications simply exacerbate the complexity of mobile computing.

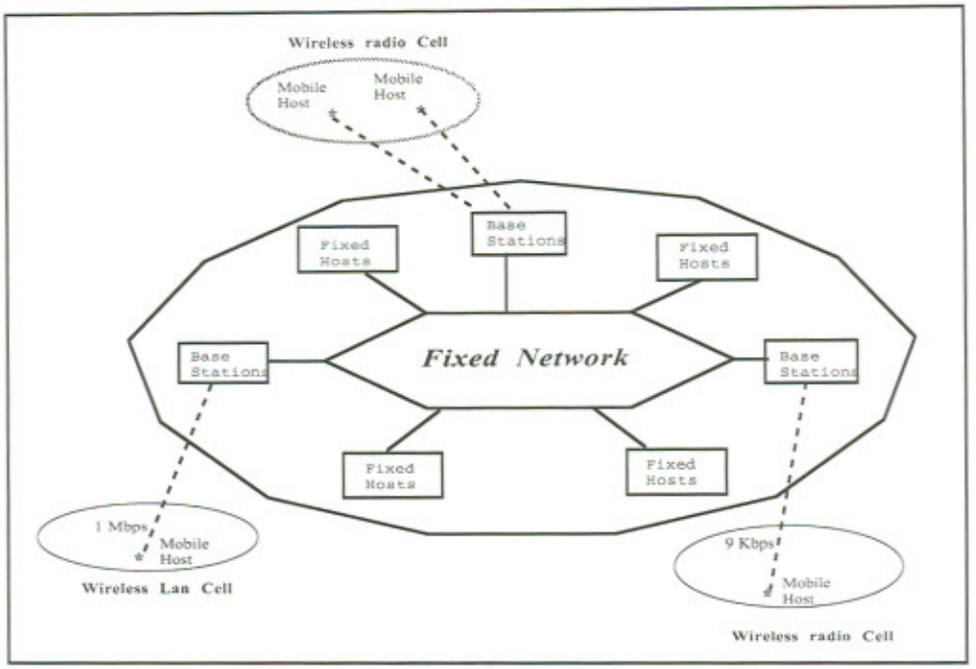

Figure 1: Network Architecture

It is speculated that ubiquitous communications will be provided by PCNs in a hybrid fashion: heavily populated areas will be covered by cheap base stations of small radius (picocells); less populated areas will be covered by bigger radius base stations and farm land, remote areas and highways wil be covered by satellites that will provide the bridge between these different islands of population density. This architecture increases the challenges for the definition of a PCN since it now has to take into consideration the different power and transmission requirements impose by these different islands of communications.

2.1 Issues and Limitations of the Wireless Environment

We survey some of the research challenges induced by (a) the wireless medium, (b) mobility and (c) the portability of wireless elements [Satya96b, FoZa94, ImBa94c, AlKo93]. In section 3, we present the impact of these challenges on defining an appropriate model for mobile applications. Of course, some models may put greater emphasis on some of the characteristics while ignoring others, depending on the application for which they are built and the specific requirements of the particular environment.

Wireless Medium

Wireless networks are more expensive, offer less bandwidth, higher latency, and are less reliable than wireline networks.

- A reason that makes bandwidth consumption a major concern for wireless computing designs is that data transmission over the air is currently monetarily expensive [BaIm94]. The cost per byte transmitted is orders of magnitude greater than traditional wireline (LAN/WAN) networks. Sending a 10K page costs from one to two dollars over current wireless networks. For example, accessing[4] a popular quote server (for a simple query) over a packet radio network required the transmition of 137,649 bytes and incurred a charge of $27 [HoSaL97]. Even after establishing a local cache the second quote was still very expensive, about $0.50. Consequently, connectivity is weak and often intermittent since mobile elements may voluntary operate disconnected for long periods of time for reasons of cost.

- The response time for wireless links in WANs is very slow compared to their wireline or wireless LAN counterpart, thus latency is a factor affecting the performance of wireless applications. Many systems [i.e., HoSaL97] depend on communications support (e.g., CDPD [IBM95], ARTour[5] [IBM95a]) that provides TCP/IP over wireless links. A typical TCP/IP request takes around 15 seconds [Ora97, HoSaL97].

- The capacity of WAN wireless links is very limited compared to most wireline links, since wireless communications face many obstacles because the surrounding environment interacts with the signal. Thus, while the growth in wired network bandwidth has been tremendous (in current technology Ethernet provides 10 Mbps, FDDI 100 Mbs and ATM 155 Mbps), products for wireless communication achieve much lower rates. In particular, depending on the technology, the channel bit rate of WAN wireless links ranges from 4800 bps (e.g., Ardis MDC4800) to 19200 bps (CDPD, RD-LAP [ARDIS97]) with the effective data rate being substantially lower due to latency and retransmission. For radio transmission the error rate is so high that the effective bandwidth is limited to less than 10 Kbps [Miller94]. A WAN wireless link is also ***shared*** among all the mobile terminals within range of the wireless base station. Thus, the data rate realized by any given wireless device is the link speed divided by the number of devices simultaneously using the link (i.e., similar to a multi-drop link). This affects not only bandwidth but latency as well.

- Wireless WAN connections are significantly less reliable than wireline connections. Wireless connections may be intermittently or permanently disrupted for various reasons: the mobile device goes out-of-range; the mobile device goes behind a barrier that blocks the signal; or, higher layer communications or applications protocols time out because of delayed responses. Disruptions may be hidden from the wireless application by error recovery procedures in the link or higher layer communications protocols. However, this often results in excessive retransmissions (and extra cost), not to mention user frustration that may result in request resubmissions.

---

[4] As described in [HoSaL97], these tests were performed on a production Mobitex [RAM97] network operated at 8,000 bps, connected to an enterprise network which was connected to the Internet.

[5] ARTour is a registered trademark of the IBM Corp.

This results in a form of distributed computing with drastically different connectivity assumptions than in traditional distributed systems, where disconnections are rare. In addition, the accepted wireless infrastructure provides *variant connectivity* since wireless technologies vary on the degree of bandwidth and reliability they provide.

Mobility

The location of mobile elements and therefore their point of attachment to the fixed network change as they move. The consequences of mobility are numerous.

- Mobility introduces various forms of heterogeneity. Thus, connectivity becomes highly variant in performance and reliability. For instance, outdoors, a mobile client may have to rely on low bandwidth networks, while inside a building it may be offered reliable high-bandwidth connectivity or even operate connected via wireline connections. Moreover, there may be areas with no adequate coverage resulting in disconnections of various durations. Then, the number of devices in a network cell changes with time, and so do both the load at the base station and bandwidth availability. There may be also variability in the provision of specific services, such as in the type of available printers or weather reports. Finally, the resources available to a mobile element vary, for example, a docked computer has more memory or is equipped with a larger screen.

- The configuration of a system that includes mobile elements is not static. Consequently, in designing distributed algorithms, we can no more rely on a fixed topology. In general, the center of activity, the system load, and the notion of locality change dynamically.

- The location of moving objects changes with time. Thus, the search cost to locate mobile elements is added to the cost of each communication involving them. Efficient data structures, algorithms, and query execution plans must be devised for representing, managing, and querying the location of mobile elements, which is a fast changing data.

Mobility also raises very important security and authentication issues. However, we will not treat such issues in this paper.

Portability of Mobile Elements

Mobile elements must be light and small to be easily carried around. Such considerations, in conjunction with a given cost and level of technology, will keep mobile elements having less resources than static elements, including memory, screen size and disk capacity. This results in an asymmetry in mobile computing systems: fixed hosts have sufficient resources, while mobile elements are resource poor.

Furthermore, mobile elements must rely for their operation on the finite energy provided by batteries. Even with advances in battery technology, this concern will not cease to exist. Concern for power consumption must span various levels in hardware and software design. For example, software systems must take into consideration in their design doze mode

operation. Finally, mobile elements are easier to be accidentally damaged, stolen, or lost. Thus, mobile elements are less secure and reliable than static elements.

3. Challenges for Mobile Computing Models

The characteristics of the mobile and wireless environment place a number of constraints that define the capabilities of the desired mobile computational model. In this section we present the various challenges imposed by the wireless environment and describe why the client/server mode is inadequate to cope with them.

3.1 Requirements

Functionality of the Mobile unit

An important design consideration is the type of functionality assigned to mobile hosts. Mobile units are still characterized as unreliable and prone to hard failures, i.e., theft, loss or accidental damage. Mobile elements are also resource-poor relative to static hosts. For these reasons, there are approaches that treat the mobile unit as a dumb terminal running just a user-interface. The [NaSe96] and ParcTab [ScAGTW93] projects employ such a dump terminal approach and off-load all functionality from the mobile unit to the fixed network. On the other hand, slow and unreliable networks argue for placing more functionality at the mobile hosts so that they are less dependent on remote servers. Although, there is no consensus yet on the specific role the mobile host must play in distributed computation, the above contradictory considerations tend to favor models that provide for a flexible adjustment of the functionality assigned to mobile clients.

In general, the type of functionality assigned to a mobile unit must be a function of its capabilities and the characteristics of the connectivity. Light weight clients (i.e., vulnerable and poor in resources) argue for more dependency on the fixed network while heavy-weight clients (i.e., secure and rich in resources) for less dependency. Taking into account the capabilities of a specific mobile unit, connectivity conditions are dealt differently. For example, in anticipation of a disconnection, a disconnected heavy-weight mobile host may be assigned additional functionality to operate autonomously while, a disconnected light-weight one off-loads most of its functionality to the fixed network.

Disconnections

Disconnections can be categorized in various ways. First, disconnections may be *voluntary*, e.g., when the user deliberately avoids network access to reduce cost, power consumption, or bandwidth use, or *forced* e.g., when the portable enters a region where there is no network coverage. Then, disconnections may be *predictable* or *sudden*. For example, voluntary disconnections are predictable. Other predictable disconnections include those that can be detected by changes in the signal strength, by predicting the battery lifetime, or by utilizing knowledge of the bandwidth distribution. Finally, disconnections can be categorized based on their duration. Very short disconnections, such as those resulting from handoffs, can be masked by the hardware or low-level software. Other disconnections may either be handled at various levels, e.g., by the file system or an application, or may be made visible to the user. *Since disconnections are very common, supporting disconnected operation, that is allowing the mobile unit to operate even when disconnected, is a central design goal in mobile computing.*

Weak Connectivity

Weak connectivity is the cumulative effect of the unreliable and relativelly low bandwidth wireless media, mobility itself and the number of mobile units sharing the media within a particular cell. For example, since the number of the mobile units can dynamically change due to mobility, the effective data rate is similarly affected often resulting in weak connectivity. Since weak connectivity is a fact of life in wireless environments, measures must be taken by the various computing models to deal with such connectivity conditions.

Mobility

Mobility results in an environment with various connectivity and computational assumptions (i.e., disconnections, weak connectivity, hand-offs, location searches etc.) requiring the participating devices to dynamically adapt to it. In addition, the server distribution on the fixed network might need to change to facilitate a more efficient data access for the mobile client. For example, in the mobile environment, dynamic server replication might be employed for performance and scalability. Widely scattered replicated servers can be used for efficiently accommodating highly mobile clients.

Adaptivity

The mobile environment is a dynamically changing one. Connectivity conditions vary from total disconnections to full connectivity. In addition, the resources available to mobile computers are not static either, for instance a "docked" mobile computer may have access to a larger display or memory. Furthermore, the location of mobile elements changes and so does the network configuration and the center of computational activity. Thus, a mobile system is presented with resources of varying number and quality. *Consequently, a desired property of software systems for mobile computing is their ability to adapt to the constantly changing environmental conditions* [Katz94, Satya96a, JoTK97, FoZa94].

But how can adaptivity be captured and realized? A possible answer is by varying the partition of duties between the client and the server. For instance, during disconnection, a mobile client works autonomously, while during periods of strong connectivity, the client depends heavily on the fixed network sparing its scarce local resources. Another way to realize adaptivity is by varying the quality of data available at the clients depending on connectivity. One way to quantify quality is the notion of *fidelity* introduced in [NoPS96]. Fidelity is defined as the degree to which a copy of data presented for use at a client matches the reference copy at the server. Fidelity has many dimensions [NoPS96]. One universal dimension is consistency. Other dimensions depend on the type of data in question. For example, video data has at least the additional dimensions of frame rate and image quality, while spatial data have dimensions of resolution or minimum feature size. Finally, the form adaptation takes depends not only on the type of data but also on the application. Take for example colored images. In cases of weak connectivity, a web browser may sacrifice both color and pixel resolution of an image when used for web surfing. However, a viewer used in a medical application cannot tolerate losing the detail of an image used for medical diagnosis.

Besides the partition of data and computation of mobile applications between mobile and static elements, an issue germane to system design is where should support for mobility and adaptivity be placed [Satya96b, NoPS96]. Should applications be aware of their environment? Mobile computing models should support various degrees of awareness.

Upward Compatibility

The various computing models should be flexible enough to accommodate not only one new type of applications, but multiple types and application models, including: current and emerging TCP/IP applications, terminal emulation and existing applications, applications using dialup services and new mobile application paradigms. A key point for mobile computing's wide acceptance is the user's efficient access to applications [PiSa98]. *This includes access to legacy and existing applications with minimum effort and cost, and cost effective and reliable support for new applications.*

3.2 The Inadequacy of the Traditional Client/Server model

With client/server (c/s) computing in its simplest form, an application component executing in one computing system, called the client, requests a service from an application component executing in another computing system, called the server. In a wireless setting, the mobile host acts as the client requesting services from servers located at the fixed network (Figure 2). In some cases, functionality and data are distributed across multiple servers at different fixed hosts that may have to communicate with each other to satisfy the client's request.

The assumptions under which the traditional client/server paradigm has been developed are very clear; resource rich and reliable clients, fast and reliable communications. The funtionality of the client and server is clearly defined, by the context of the application. There is a rigid division of responsibilities among the client and the server and the inability to fulfill them implies the end of execution. These assumptions are quiet incompatible to those present in the wireless and mobile environment.

In the wireline c/s paradigm disconnections and weak connectivity are rare and fare no special treatment. Disconnections, for example, are not an issue in the traditional client/server environment and are not considered in the design of most wireline c/s application. In the rare cases of disconnection (commonly detected as a communication time-out) the execution of the application is, usually, discontinued. Similarly rare is weak connectivity. It seriously affects the application by resulting into unacceptable performance simply because the environment did not anticipated it.

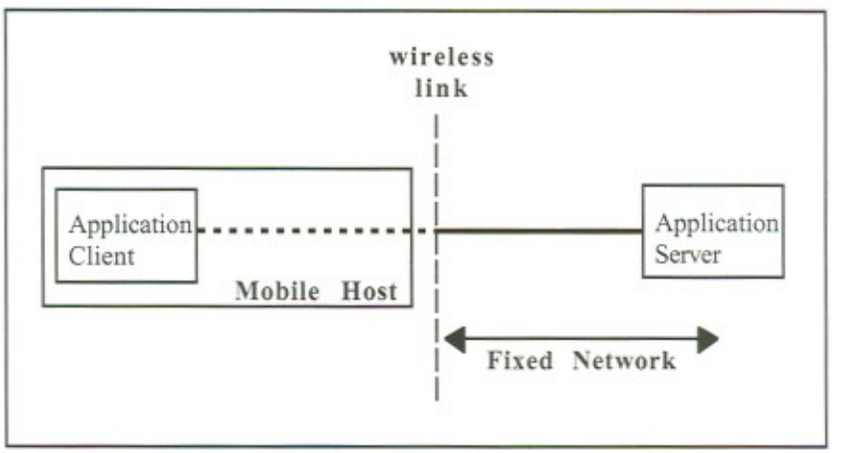

Figure 2: Simple Client/Server Model

Adaptivity is viewed as an answer to the variable characteristics of the wireless media and the capabilities of the mobile client. Facilitating adaptivity in the wireline c/s model implies that the client and server site of the application must be designed as such. That is, ths application must be able to monitor the environment and dynamically adjust its code at both the server and client side. However, c/s applications (in general) never faced the need for adaptivity and traditionally are uninterested to the changes of the environment. One of the few environments in which the need for wireline adaptivity appeared is within the context of WWW where due to its popularity, bandwidth consumption increased dramatically and weak connectivity became a serious issue. Web proxies and a number of optimizations were used to alleviate the problem. Similar approaches are proposed for the mobile/wireless environment.

While mobility is central to the design of mobile computational models, in the c/s paradigm the location of the client and servers is fixed simplifying the requirement of locating and communicating with each other. Thus, the challenges imposed by mobility are not an issue for the wireline environment and no special measures are taken to alleviate its impact.

In general, there should not be a clear division of functionality between the mobile client and the server at the fixed network. [Satya96a] talks about an extended client/server model where the roles of clients and servers are temporarily blurred. The most evident such case is during disconnections, when a disconnected client has to emulate the functionality of a server to continue operating.

An important parameter of client/server architectures is the type of communication mechanism used to exchange information between the client and the server. The traditional approach is the remote procedure call (RPC) mechanism. In this paradigm, a client calls a procedure which is executed at a remote server. However, synchronous RPCs are inadequate since the client is blocked in cases of disconnections. Moreover, traditional RPCs leave small chance of reducing communication cost. One other possibility is the direct exchange of messages between the client and the server. This approach is also not adequate for slow and unreliable networks such as in wireless computing. In such environments, a less direct mechanism, in which messages are queued at the two ends is more appropriate. This leads to various levels of indirection.

To address these problems, there have been various proposals for an asynchronous RPC [JoTK97, BaBa97, AtDu93] mechanism. In asynchronous queued RPC [JoTK97], when an application issues an RPC, the RPC is stored in a local stable log and control is immediately returned to the application. When the mobile client is connected, the log is drained in the background and any queued RPCs are forwarded to the server. Queuing RPCs leaves space for performing various optimizations on the log. For instance, the Rover toolkit [JoTK97] reorders logged requests based on consistency requirements and application-specified operation priorities. Delivering any replies from the server to the client at the mobile host may require multiple retries [JoTK97, BaBa97]. Specifically, if a mobile host is disconnected between issuing the request and receiving the reply, the server may periodically attempt to contact the mobile host and deliver the reply. This extended RPC mechanism enables applications to use different communication channels for the requests and responses and to close channels during the intervening period [JoTK97].

Extensions to the traditional client/server model, such as queuing RPC, are necessary to sustain disconnected operation and weak connectivity. Further optimizations, such as filtering or compression, are also important to further reductions of the effect of weak connectivity. For these reasons the traditional client/server model must be extended to provide specila components responsible for implementing such appropriate optimizations and thus imposing minimum changes in clients and servers. In addition, for light-weight and unreliable clients, it is critical to be able to move part of their operation to the fixed network. Such considerations and others led to the extensions of the client/server model presented in the following sections.

4. Mobile Computing Models

Devising appropriate models for structuring applications that involve wireless elements is an issue central to developing software for mobile computing. In this section, we elaborate on such models, their strengths and weaknesses. First, we consider extensions to the popular client/server model, then the use of peer to peer models, and finally the employment of mobile agents.

4.1 Mobile Client/Server Models

The extensions of the client/server model are merely based in the introduction of agents placed between the mobile client and the fixed server. The agents alleviate both the constraints of the wireless link, by performing various communication optimizations, and of any resource constraints, by undertaking part of the functionality of resource-poor mobile clients. The degree in which this is achieved depends on the placement and the functionality of the agents.

4.1.1 The Client/Agent/Server Model

An emerging computational model is a three-tier or client/agent/server (c/a/s) model [BaBIM93, FoGBA96, Ora97, ZeDu97, TeSSM96], that uses messaging and queuing infrastructure for communications from the mobile client to the agent and from the agent to the server (Figure 3). In its most generic form, an *agent* or *proxy* is just a surrogate of the client on the fixed network. This architecture somewhat alleviates the impact of the limited bandwidth and the poor reliability of the wireless link by continuously maintaining the client's presence on the fixed network via the agent. Agents split the interaction between mobile clients and fixed servers in two parts, one between the client and the agent, and one between the agent and the server. Different protocols can be used for each part of the interaction and each part of the interaction may be executed independently of the other.

The c/a/s model is most appropriate for light-weight clients of limited computational power and resources. It moves responsibilities from the client to the agent. For example, a complex client request can be managed by the agent with only the final result transmitted to the client. Located on the fixed network, the agent has access to high bandwidth links and large computational resources, that it can use to benefit its client. Similarly, performing caching at the agent can minimize long round trips on the network and thus reduce application response time. To further enhance the client's benefit, the agent can be used to off-load significant

computational burden from the server. For example, data compression can be performed by the agent instead of the server.

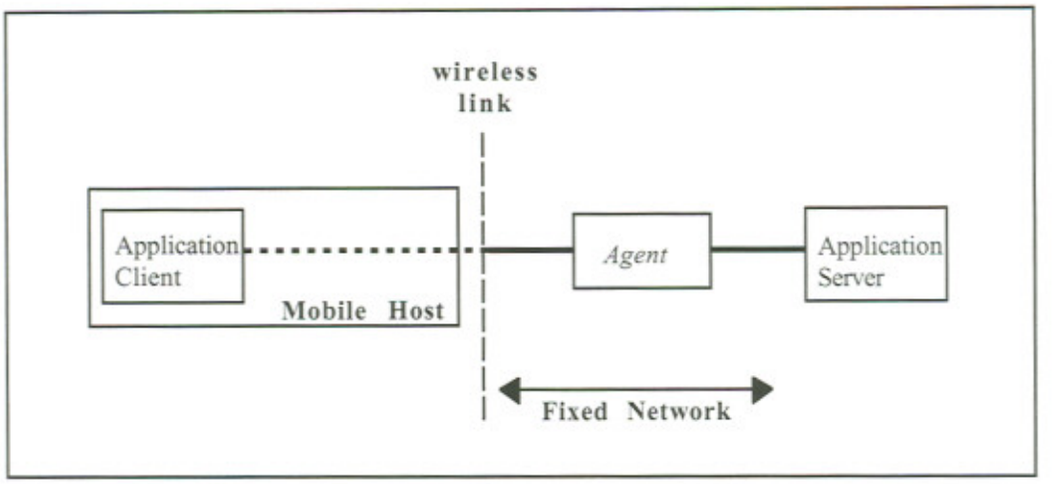

Figure 3: Client-Server based Models: Client/Agent/Server

In general, how the agent's resources are used, that is the functionality of the agent, greatly depends on the specific application the client needs to support. Minimum functionality at the agent includes support for messaging and queuing for communication between the mobile client and the agent so to be able to support weak connectivity and disconnections. The agent can also assume a more active and intelligent role [Ora97, TeSSM96], in which case, when application-specific predefined events occur, it notifies the client appropriately. Furthermore, to limit computations on the mobile unit and thus preserve valuable resources (i.e., battery life), the agent might be responsible for starting/stopping specific functions at the mobile unit or for executing client specific services (that is, untertake a somewhat more coordinating role).

To deal with disconnections, a mobile client can submit its requests to the agent and wait to retrieve the results when connection is re-establish. In the meantime, any requests to the disconnected client can be queued at the agent to be transferred upon reconnection. The agent can be used in a similar way to preserve battery life. The client submits its requests to the agent, enters the doze mode, and waits the agent to wake it up when the response becomes available. Weak connectivity can be effectively handled by the agent in a variety of ways. The agent can employ a number of optimization techniques to minimize the size of data to be transmitted to the client depending on the type of data and on the specific application. The agent can manipulate the data prior to their transmissions to the client [ZeDu97, FoGBA96, TeSSM96], by changing their transmission order so that the most important information is transfered first, by performing data specific lossy compression that tailors content to the specific constraints of the client, or by batching together multiple replies.

Taxonomy of the C/A/S architecture and other features

Agents are used in a variety of forms and roles in this particular architecture. At one extreme, an agent acts as the complete surrogate of a mobile host on the fixed network. In this case, any communication to and from the mobile host goes through the mobile host's agent. The

surrogate role of an agent can be generalized by having an agent acting as a surrogate of multiple mobile hosts [FoGBA96]. At the other extreme, the agent is attached to a specific service or application, e.g., web browsing [HoSaL97] or database access [Ora97]. Any client's request and server's reply associated with this application is communicated through the service-specific agent. In this scenario, a mobile host must be associated with as many agents as the services it needs access to. This second case can be symmetrically generalized by having a service-specific agent servicing multiple clients [HoSaL97].

The situation becomes more involved when a service is associated with multiple agents [Ora97] and this group of service-specific agents serves multiple clients. In this case, any client's request associated with the service can be serviced by any available agent in the group. All this flexibility, however, entails the need of a more complex management function. A service-specific *meta-agent* is required to coordinate clients requests and the service-specific group of agents. Thus, the meta-agent can be viewed as a service-specific agent serving multiple clients with the other agents playing only a supporting role. In [Ora97], the role of the meta-agent is played by an agent management facility, called Message Gateway.

The exact position of the agent at the fixed network depends on its role. Placing the agent at the fringe of the fixed network, i.e., at the base station, has some advantages especially when the agent acts as the surrogate of mobile hosts under its coverage [ZeDu97, BaBIM93]: it is easier to gather informations for the wireless link characteristics; a special link level protocol can be used between the mobile host and the agent; and personalized information about the mobile hosts is available locally. On the other hand, the agent may need to move along with its mobile host, or the current base station may not be trustworthy. In the case of service-specific agents, it makes sense to place them either closer to the majority of their clients or closer to the server.

The introduction of agents affects routing [PiSa98] in various ways depending on the exact role the agent plays. If the agent totally represents one or multiple mobile hosts, then all traffic involving the mobile hosts flows through the agent. Thus, special treatment is required to direct all such communication through the agent. To this end, [ZeDu95] augments the Columbia Mobile IP protocol [IoMa93] to allow a mobile host to choose one certain host to be its agent and guarantee that all traffic will pass through it. If the agent represents services [HoSaL97, Ora97], then only communication related to the service passes through the agent.

While the client/agent/server model offers a number of advantages, it fails to sustain the current computation at the mobile client during periods of disconnection. At the event of a disconnection, the mobile client cannot continue to operate uninterrupted. Furthermore, although the server notices no changes, the model still requires changes to the client code for the development of the client/agent interaction rendering the execution and maintenance of legacy applications problematic. Adaptivity is not well supported by this model since it requires the participation of the (light-weight!) client site application which must be specifically design to deal with it. Finally, the agent can directly optimize only data transmission over the wireless link from the fixed network to the mobile client and not in the opposite direction.

4.1.2 The Client/Intercept/Server Model

To address the shortcomings of the Client/Intercept/Server (c/a/s) model, [SaP97, HoSaL97] propose the deployment of a client-side agent that will run at the end-user mobile device along with the agent of the c/a/s model that runs within the wireline network (Figure 4). The client-side agent intercepts client's request and together with the server-side agent performs optimizations to reduce data transmission over the wireless link, improve data availability and sustain uninterrupted the mobile computation. From the point of view of the client, the client-side agent appears as the local server proxy that is co-resident with the client. Similarly, the server-side agent appears as the local client proxy that resides on the fixed network, most likely, but not necessarily, co-resident with the server representing the client.

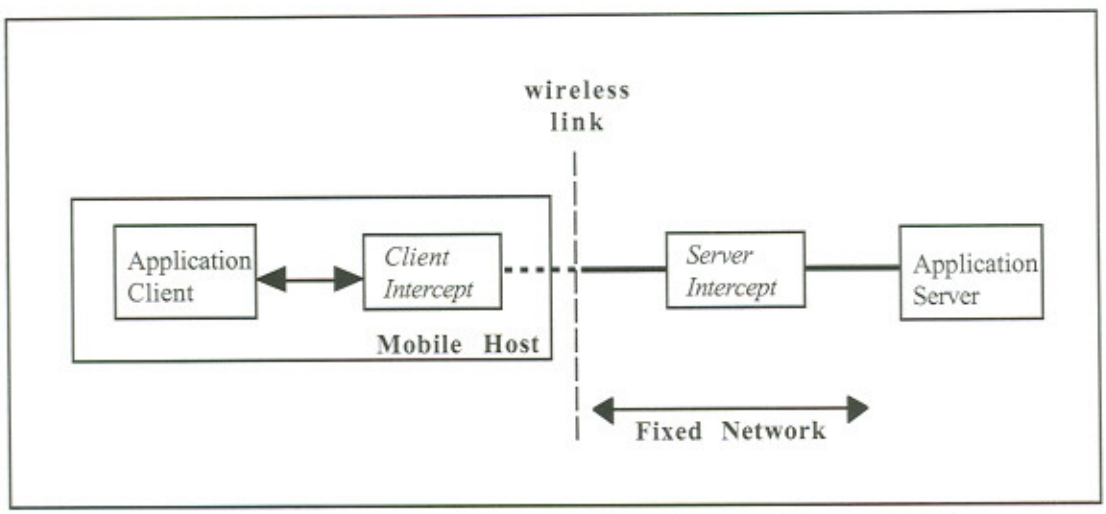

Figure 4: Client-Server based Models: Client/Intercept/Server Model.

This model can be viewed as the adaptation of the client/server model for wireless computing with the pair of agents responsible for minimizing the effect of the wireless link. Since the pair of agents is virtually inserted in the data path between the client and the server, the model is named in [SaP97, HoSaL97] for short client/intercept/server instead of client/agent/agent/server model. This model is more appropriate for heavy-weight clients with enough computational power and secondary storage to support the client-side agent.

The intercept model is transparent to both the client and the server. Therefore, the agent pair can be employed with any client application. The communication protocol between the two agents can facilitate highly effective data reduction and protocol optimization without limiting the functionality or interoperability of the client. The cooperation of the two agents allows for more efficient optimizations of the wireless link for the benefit of different applications. In addition, application specific optimizations can be more effectively realized by the agent pair. The existence of the agent pair facilitates high degree of adaptivity since the two agents can dynamically divide the workload among them based on the various environmental conditions.

The model offers flexibility in handling disconnections. For instance, a local cache may be maintained at the client-side agent to allow it to work autonomously. The cache can be used to satisfy the client's requirements for data during disconnections. Cache misses may be queued

by the client-side agent to be served upon reconnection. Similarly, requests to the client can be queued at the server-side agent and transfered to the client upon reconnection. Weak connectivity can also be handled in a variety of ways. For example, relocating computation from the client-side agent to the server-side agent or vice versa can minimize the effect of weak connectivity. Background prefetching to the client-side agent can reduce communication during weak connectivity. During periods of strong connectivity, on the other hand, the client-side agent can heavily depend on the server-side agent sparing its scarce local resources. For example, to require high degree of data fidelity or no/low compression of the transmitted data.

The intercept model provides high support for upward compatibility. Legacy and existing applications can be executed as before since the agent pair shields them from the limitations of mobility and the wireless media. The main weakness of the model is that every application requires development work both at the server and the client site. However, there is no need to develop a pair of agents for every instance of an application. Instead, since the functionality and optimizations performed by the agent pair is generic enough, it is only required to develop a different pair of agents per application type e.g., file, database, or web application. For example, prefetching documents at the cache of the client-side agent follows the same principles independently of the specific type of web-application while prefetching for different type of applications, i.e., database and web application, might require quite different techiques.

Classifying the roles of the two agents results in a taxonomy similar to that provided in the previous section for the agent role of the client/agent/server model. For example, the server-side agent can be a surrogate of the client or be attached to a specific service.

The intercept model provides for a clear distinction and splitting of responsibilities between the client and server agents. At the same time it provides for transparency and for high degree of collaboration between the pair of agents. The idea of employing proxy pairs has been gaining lately some attention [ZeDu97, FoGBA96]. An intercept approach is employed by WebExpress [HoSaL97], an IBM system for optimizing web browsing in a wireless environment. In this system the client-side agent is called client-side intercept (CSI), while the server-side agent is called server-side intercept (SSI).

4.1.3 Peer-to-Peer Models

In a peer-to-peer distributed architecture, there is no distinction between clients and servers. Ideally, each site has the full functionality of both a client and a server. Adapting such models in mobile computing, makes mobile hosts equal partners in distributed computations and are thus only appropriate for heavy-weight clients. Disconnections have the additional negative effect of making the server unavailable to clients requesting its services. To deal with disconnections and weak connectivity a server-side intercept agent may need to be placed on the mobile host as well (see figure 5). In this case, agents may possess special features to take into account the fact that the server is running on a mobile host. For instance, a server at the mobile host cannot be continuously active because, to conserve power, the mobile host may be switched-off or operating in the doze mode. A mechanism to automatically start applications on demand [AtDu93] is useful in such cases. Obviously mobility does not pose any special problems other than those resulting from the wireless infrastructure itself.

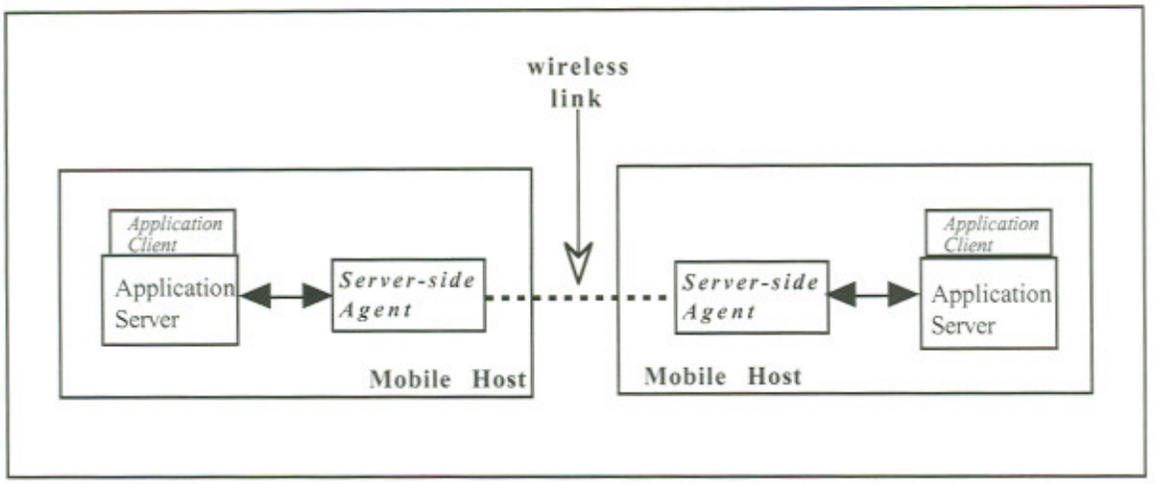

Figure 5: The Peer to Peer model enhanced with a Server-side agent

There are certain applications for which the peer-to-peer model is adequate [RePGSR96]. Consider the case of two partners performing cooperative work on some data using their portable computers. In a client/server model, they cannot trade their updates directly. Instead, each client has to connect with the server machine to be informed of each other's actions. This may incur excessive communication costs, when the server is located far away from the clients. Even worst, in cases of physical disconnection of the clients from the server, there is no way that the two clients can interact with each other, even when a communication path connecting them is available.

4.2 Mobile Agent Models

Mobile agents are processes dispatched from a source computer to accomplish a specified task [ChGHLPT95, White96]. Each *mobile agent* is a computation along with its own data and execution state. In this way, the mobile agent paradigm extends the RPC communication mechanism according to which a message sent by a client is just a procedure call. After its submission, the mobile agent proceeds autonomously and independently of the sending client. When the agent reaches a server, it is delivered to an agent execution environment. Then, if the agent possesses necessary authentication credentials, its executable parts are started. To accomplish its task, the mobile agent can transport itself to another server, spawn new agents, or interact with other agents. Upon completion, the mobile agent delivers the results to the sending client or to another server.

The driving force motivating mobile agent-based computation is twofold. First, mobile agents provide an efficient, asynchronous method for searching for information or services in rapidly evolving networks: mobile clients are launched into the unstructured network and roam around to gather information. Second, mobile agents support intermittent connectivity, slow networks, and light-weight devices. This second property makes the use of mobile agents in mobile computing very attractive [ChGHLPT95, PiBh95a].

Besides their computational components, mobile agents are equipped with intelligence to reason, make decisions, and solve problems. Thus, there are two kinds of languages involved

with an agent. One is a programming language in which the programming content of the agent is written. This is usually a script-like language. The other language is concerned with the intelligent aspects of mobile agents. This is a language for knowledge representation. Languages for knowledge representation provide the means to express goals, tasks, preferences, beliefs and vocabularies appropriate to various domains. A leading approach in defining interoperable languages for knowledge-presentation is the ARPA knowledge sharing effort [ARPA97].

Besides intelligence, some of the key aspects of the mobile agent computing paradigm are:

- the ability of an agent to interact and cooperate with other agents,
- agent autonomy, in the sense that agents' execution proceeds with little or no intervention with the sending client,
- agent interoperability, i.e., agents can be executed in diverse platforms,
- agent reactiveness, the ability of an agent to respond to external events, and
- agent mobility, i.e., their ability to roam among a set of networked servers.

One of the main obstacles to the acceptance of mobile agents in commercial applications is security. Protection against viruses, preventing mobile agents entering endless loops and consuming all available resources, authentication, and privacy are just a few of the implementation challenges to be addressed. General Magic's Telescrtipt (TM) technology [GMAGIC97] is possibly one of the most well-known implementations of the concept of mobile agents in a commercial setting. [MAge97] provides a rich source of information relating to mobile agents.

The agent computational model supports disconnected operation. During a brief connection service, a mobile client submits an agent and then disconnects.The agent proceeds independently to accomplish the delegated task. When the task is completed, the agent waits till reconnection to submit the result to the mobile client. Conversely, a mobile agent may be loaded from the fixed network onto a laptop before disconnection. The agent acts as a surrogate for the application allowing interaction with the user even during disconnections. Weak connectivity is also supported by the model since the overall communication traffic through the wireless link is reduced from a possibly large number of messages to the submission of a single agent and then of its result. In addition, by letting mobile hosts submit agents, the burden of computation is shifted from the resource-poor mobile hosts to the fixed network. Mobility is inherent in the model. Mobile agents migrate not only to find the required resources but also to follow mobile clients. Finally, mobile agents provide the flexibility to load functionality to and from a mobile host depending on bandwidth and other available resources. And, thus, materializing high degree of adaptivity. One might say that, like mobility, adaptivity is too inherent to the model.

Mobile Agents and the Client/Server Variations

The mobile agent computational paradigm is not orthogonal to the client/server model and its extensions. For example, the agent of the c/a/s model may be seen as a form of a static agent, i.e, an agent lacking the ability to migrate to other servers. Alternatively, one can implement the agent of the c/a/s model as a mobile agent and let it move on the fixed network following its associated client.

Mobile agents can be used in conjunction with agents located at the fixed network. Let's call the agents at the fixed network proxies for clarity. In this scenario, a client submits a general mobile agent to its proxy. The proxy refines and extends the mobile agent before launching it to servers on the network. When the mobile agent finishes its task, it first communicates its results to the proxy. The proxy filters out any unnecessary information and transmits to the mobile client only the relevant data. Such an approach entails enhancing the proxies with capabilities to process mobile agents. Building on this approach, a proxy may be *programmable*, that is extended with the ability to execute mobile agents submitted to it by clients or servers. Such an approach is in accordance with current research on *active networks* [TeSSM96].

We can further extend this approach by combining mobile agents with the Client/Intercept model. The Intercept model runs on top of the mobile agent model with the client-side and the server-side intercept agents/proxies utilizing mobile agents to perform the various tasks. This cooperation seems to provide the most advantages. It allows for upward compatibility, higher adaptivity and greater degree of independence between the client side and the server side of an application. The intercept agents/proxies are solely responsible for the submission and refinement of the various mobile agents, thus shielding the client and server applications from the details of adaptivity and awareness.

4.3 Comparison of the Models

Not all models are appropriate for the challenges presented by the mobile and wireless environment. The mobile computational paradigm took an evolutionary path where the capabilities of each model was a function of the characteristics of the mobile unit, the limitations of the wireless media and the challenges of mobility. The capabilities of each model pertaining to the requirements defined in section 3.1 are summarized in Table 1.

	Client/Server	Client/Agent	Client/Intercept	Peer to Peer	Mobile-Agents
Disconnections	NO	partly	YES	partly	partly
Weak Connectivity	NO	partly	YES	partly	partly
Adaptivity	NO	medium	high	NO	medium to high
Type of Clients	heavy	light	heavy	heavy	light
Functionality divisions	low	high	low-high	low-high	low-high
Mobility	not aware	not aware	aware - not aware	aware - not aware	inherent
Upward Compatibility	high	server side only	high	high	server side only

Table 1. Strengths and weaknesses of the mobile computing models

Table 2 presents the capabilities of the various models when combined with the mobile agents model. We thought it appropriate to discuss these hybrid models since mobility is not only inherent to the environment we are discussing but central to the design and evaluation of these new models. Table 2 shows that the addition of mobile agents does not affect the capabilities of the various models significantly other than the mobility issue itself. It does however, provide more flexibility for computations and executions on the fixed network.

	Client/Server	Client/Agent, Mobile agents	Client/Intercept, Mobile agents	Peer to Peer, Mobile agents
Disconnections	NO	partly	YES	partly
Weak Connectivity	NO	partly	YES	partly
Adaptivity	NO	medium	high	NO
Type of Clients	heavy	light	heavy	heavy
Functionality divisions	low	high	low-high **more flexible**	low-high
Mobility	not aware	**inherent**	**inherent**	**inherent**
Upward Compatibility	high	server side only	high	high

Table 2. Strengths and weaknesses of the hybrid mobile computing models

5. An Example: Web Browsing

The goal of this section is to clarify through a concrete example the several architectural alternatives, as well as the various methods for realizing adaptivity. The example application is web browsing. Web technology [BCLNS94] is rapidly being accepted as a universal interface for network access to information. In conjunction with today's mobile devices and the emerging wireless technologies, it offers the potential for unprecedented access to data and applications by mobile users. The popularity of web technology suggests that web browsers offer a compelling end-user interface for many mobile wireless applications.

Web technology is based on the HyperText Markup Language (HTML) [BeCo95] and the HyperText Transport Protocol (HTTP) [BeFF95]. HTML provides a common representation for information, and HTTP defines the common protocol for transporting information between web clients and web servers. The web browser serves as the end-user interface; it is responsible for sending user requests to the appropriate web server often via a web proxy gateway and for formatting and displaying HTML data streams returned to the client device. In addition to information retrieval and browsing, web technology is used as the user interface for many form-processing applications.

In contrast to random, ad hoc web browsing, form-based data management aims at routine repetitive commercial applications such as: visiting nurses or medical personnel; service workers involved with equipment repair, checking warehouses for parts; or mobile sales-persons that need to query product data, place orders, and check credit.

5.1 Inhibitors to Web Browsing

The wireline model for web applications is based on the client/server model, where the client directly communicates the various requests to the server. Such web systems usually choose to trade inexpensive network bandwidth for reduced storage consumption. The serious limitations, however, of wide area wireless communications (i.e., restricted bandwidth, high latency, high cost, and poor reliability) renders such a client/server based web browsing impractical. In addition, the web HTTP protocol presents the following inefficiencies:

- *Connection overhead*: Each request for an HTML page or graphic object (i.e., GIF or JPEG file) requires the browser to open a TCP/IP socket. This operation adds to the data overhead and greatly increases latency. [Ora97, HoSaL97] observed connection times in the neighborhood of 10-15 seconds in low speed (e.g., 4800 bps) WAN wireless networks.

- *Redundant transmission of capabilities*: The HTTP protocol is stateless; this requires that the browser must (re)send its capabilities, a list of 200-400 bytes in length, with each request. These capabilities are normally the same for any given browser, i.e., client device.

- *Verbose protocol*: HTTP control information is coded in standard ASCII and employs human-friendly keywords, which increases the number of bytes transmitted per request.

Even in wireline networks, scalability of HTTP is a problem as evidenced by performance improvements in HTTP/1.1 [FiGMFB97] and active research to improve HTTP [RaMo95]. However, in a wireless environment, unacceptable response times and occasional time-outs renders web access infeasible. To use the example presented in [HoSaL97], for a simple DB2 application including 10 web pages, 7 documents and 3 forms, totaling about 30,000 bytes (including images), to fetch each web page and its corresponding images, generated 56 KB of traffic and took an excess of 20 minutes to complete. These tests were performed on a production Mobitex network connected to an enterprise network which in turn was connected to the Internet. The test environment was not a controlled one. These serious limitations of wireless web communications necessitate the deployment of a number of optimizations to achieve a usable web system.

5.2 Models for Web Computing

The purpose of this section is to provide a concrete example of the mobile computing models by using them in the design of a wireless web browsing system. The type of optimizations employed are independent of the computational model. Their effectiveness, applicability and implementation, however, varies depending on the model. We will not elaborate on these optimizations [PiSa98]; their presentation here is to provide examples of the functionality of the models. We assume that the underlying operating and communication system does not

provide any support for mobility. Which extension of the client/server model is preferable depends on the type of client. For heavy-weight clients, for example, the addition of a client-side intercept agent allows the deployment of additional optimizations without modifying the browser. On the other hand, the peer-to-peer model is used when a specific application requires placing a web server at the mobile host. Finally, the employment of mobile agents adds flexibility, especially in handling mobility and the complexity of information search.

5.2.1 Client/Server Models

Realizing the client/server model is quite simple: it only requires the web browser to be located at the mobile client and communicate directly with the web server via wireless communications. Any browsing optimization requires changes to the client or server code. Even batching web requests to optimize wireless transmission and minimize TCP/IP connections requires modifying the browser [FiGMFB97].

5.2.2 Client/Agent/Server Model

Employing the client/agent/server model implies that a web agent or web proxy is created to represent the client on the fixed network [Ora97, FoBr96, LiAKLR95, VoBe94, Bar95]. For this example application, the client is represented by a service-specific agent, the web agent. The web agent at the fixed network can serve multiple mobile web clients, in which case each client has to register with the agent. All web-related traffic to and from the mobile host goes through the web agent.

A reasonable objective is keeping changes to the web browser and server to a minimum. If the web browser is not allowed to change, then the role of the agent is relatively limited. The optimizations that can be realized by the web agent are mostly optimizations that reduce the client or server computational load and optimizations that minimize the communication overhead only from the server to the client and **not vice versa**. For example, the functionality of the web agent may include:

- *caching* that persists across browser sessions [HoSaL97], so to avoid surfing the net for repeated requests,
- *prefetching* [RaMo95], based on user profiles or on the content of a requested URL populate its cache with items that might be needed, so to optimize net surfing,
- *client-specific procedures* to reduce the client workload, such procedures may compose web documents with information from multiple URLs [Ora97] that are of interest to the user and thus take a more active role without any direct interaction with the client [TeSSM96],
- *queuing responses* [HoSaL97], during disconnections (or weak connectivity) arriving (to the agent) responses are queued to be viewed during re-connection time,
- *application-specific filters*, e.g., for compression, lossy compression, text rearrangement [FoBr96, RaMo95, Bar95] to manipulate the data before their transmission to the mobile client. Such compressions are especially important for inline graphics that add to the annoying latency of loading web pages [FoBr96, RaMo95]. Such compressions, however, must produce representations recognizable by the browser. [FoBr96] implements a www client/agent/server architecture with the agent supporting lossy compression. Depending on

the type of the compression performed, the agent achieves up to 98.2% improvement in byte transmission and up to an order of magnitude reduction in the total perceived latency at the client.

The above optimizations aim at relieving the client and server from computationally intensive tasks, such as compression and prefetching, and at reducing resources at the clients, such as storage for caching. Thus, the web-agent supports light-weight clients.

There are a number of other optimizations that can benefit wireless web browsing. The web browser, however, needs to be modified to facilitate their deployment in cooperation with the web agent. Even for the optimizations mentioned above, the browser code may need modifications, for instance, if the client cannot understand a particular representation or can not appreciate unsolicited arrivals. Even with changes in the browser code, a number of problems related to code maintenance and web server interoperability remain. Such consideration lead to the client/intercept/server model.

5.2.3 Client/Intercept/Server Model

Employing the client/intercept/server model [SaP97] introduces a web-specific agent residing at the mobile client, in addition to the web agent at the fixed network. We will call the web agent at the client, client-side agent and the web agent at the fixed network server-side agent. While the server-side agent at the fixed network serves multiple clients, the client-side agent is unique to the client. The agent pair **cooperates** to intercept and control communications over the wireless link for reducing traffic volume. The pair also optimizes the communication protocol to reduce latency. Functionality at the client-side agent includes:

- *client side caching*, to support disconnections and weak connectivity. The disconnected (or weak connected) client uses the agent's cache to satisfy user's requests,
- *asynchronous-disconnected mode* [HoSaL97], to allow requests that can not be satisfied by the cache) to be automatically queued when connectivity is lost and resumed when connectivity is re-established,
- *client-directed lossy compression*, where the user (via the client-side agent) through modifications to the transmitted HTML document demands a partial refinement of the original representation [FoBr96],
- *protocol reduction* [RaMo95] that minimizes the number of TCP/IP connections from the client-side agent to the server-side agent, and *header reduction* [HoSaL97] that reduces the HTTP header flows from the client-side agent to the server-side agent,
- *dynamic documents* [VoBe94], in which case dynamic URLs are used to allow a single URL to return different documents or execute different commands depending upon the state of the user's dynamic environment at the time the query is executed, and
- *active documents* [VoBe94] which automatically updates its context in response to changes in a user's mobile context and
- *relocatable dynamic objects* that expect the browser to interpreter and execute a message [KaPT94].

The existence of the two agents might result in even more sophisticated optimizations such as *differencing* [HoSaL97]. When the browser requests an object the HTTP request is intercepted by the client-side agent. If the cached object at the client-side agent is outdated the request is forwarded to the server-side agent. The server-side agent using differencing minimizes the utilization of the wireless link by only transmitting the difference bewteen the updated and the cached object. The client-side agent merges the difference with the local cached object to produce the browser's response. Similarly, the agent pair can minimize the number of TCP/IP connections by multiplexing requests through one long-lived connection between the client-side and the server-side agent.

Such optimizations have prove to be very effective. As [HoSaL97] reports for the DB2 example, by employing such methods, updating the query reports produced only 2 KB of network traffic. The complete elapsed time of the job was reduced from 20 minutes to under 3 minutes during peak network utilization.

The intercept model for implementing a web-browsing based system offers a number of advantages. It is transparent to both web browsers and web (proxy) servers and, therefore, can be employed with any web browser. The implementation can be largely insensitive to the development of the rapidly maturing HTML/HTTP technology to the extend where the optimizations used are not sensitive to HTTP changes. Thus, it does not have to be upgraded to run with new or different versions of web browsers that are available in the market place. The agent pair protocols can facilitate highly effective data reduction and protocol optimization without limiting any of the web browser functionality or interoperability. Note that the effectiveness of the client-side agent depends on the successful cooperation with the server-side agent.

5.2.4 Peer-to-Peer Model

In the peer-to-peer model, the mobile hosts are considered equal to the fixed hosts and capable of supporting both web browsers and web servers with locally stored HTML pages. To support such a model, the mobile host must be heavy-weight with enough resources and computational power. Requests submitted by browsers running at the mobile unit for URLs locally available are serviced without any network interaction and thus cause no performance problems. However, requests to the local server originated from browsers located at other sites or requests for URLs that do not reside at the mobile host involve the wireless link. In the first case, a server-side intercept agent needs to be placed at the mobile host to service requests originated from remote browsers. In the second case, a client-side intercept agent is required to effectively facilitate browsing.

5.2.5 Mobile Agent Model

Mobile agents can be used along with the previous models. The main flexibility of mobile agents is the ability given to web (static) agents to create and send a mobile agent around the net to collect information. The mobile agent proceeds asynchronously with minimum interaction with the web agents. Another way that mobile agents can be used is by making the

server-side web agent mobile so that it follows its roaming clients. However, the benefits of such an approach are not clear, since the agent serves many clients.

A more fruitful way to employ this model is to have the web (static) agents to create web client-side agents that are sent onto the mobile unit and thus realizing the client/intercept/server model.

5.2.6 Discussion

We have deliberately chosen to present the various models via the Web example since the implementation of wireless web browsing system provides an interesting and very beneficial level of indirection. In the development of the web-browsing example, mobility and the limitation of the wireless links were handled solely by the web system. Consequently, any application that uses the web browser or web server, e.g., using web for library search, can take advantage of such optimizations without any changes in its code. The optimizations performed at the agents will hide the slow latency or short intervals of disconnections from such applications. Handling mobility exclusively at the web-system level also means that there is no need for specific tools (e.g., provision for the execution of relocatable objects), support from the file system (e.g., caching) or filters (e.g., an RPC filter).

Adaptivity is also provided at the web-system level. There are various ways to make the web-system adaptive. For example, by supporting different types of compression or cache update methods depending on the availability of bandwidth. Adaptivity to location changes is also possible, such an example is given by dynamic documents. Adaptivity requires informing the web-agents about changes in the environment. Thus, a system that monitors the environment and conveys this information to the web-agents is a prerequisite.

6. Issues on Mobile models and Mobile Transactions

Mobile transactions are distributed transactions that involve mobile hosts. Defining mobile transactions is an important topic that needs a careful and methodical study. The intention here is not to elaborate on the definition of mobile transactions but to view transactions in the context of the mobile models introduced in the previous sections. So far there is no adequate definition of an appropriate mobile transactional model mainly due to the unclear role of the mobile host in mobile transactions. To give a single example, two-phase commit processing (2PC) [Gray78, Lamp81] has not yet been applied to the current(!) definition of mobile transactions. Studying mobile transactions within the various computational models provides a much clearer understanding of the issues involved. Commit processing, recovery and concurrency require a treatment that depends on the model and the type of mobile hosts involved. A systematic study of the various transactional models within the frameworks provided by the various computational models clarifies the functionalities assigned to each unit. In addition, it can provide the basis for defining appropriate library support in terms of primitive functions that the transactional units of each of the models must provide.

The solutions provided should always be in relation to the characteristics of the mobile host, its role in the transaction and the mobile model employed. Light-weight mobile transactional participants, for example, are best supported by the client/agent/server model or even the

traditional client/server model. Heavy-weight mobile transactional units, on the other hand, are best supported by the client/intercept model or the peer-to-peer model. With these frameworks as starting points, specific characteristics of transaction processing are better understood and easier implemented.

Light-weight transactional mobile client

In one design extreme, the role of the mobile host is limited to only issuing requests for data to the server, while all data processing and updating is performed at the fixed server. In this case, the mobile client does not own any data. The mobile host may either submit to the fixed server the operations of a transaction sequentially one at a time or the whole transaction as a single atomic unit [JiBE95]. In the former case, a more interactive mode of operation is provided, while in the later case, a better utilization of data transmission may be achieved. In any case, the execution of the transaction is the sole responsibility of the server.

For light-weight mobile transactional clients, the most appropriate model seems to be the client/agent/server or the traditional client/server model. In either case, the control of the transaction is assigned to the fixed network. The server or agent at the fixed network must be equipped with appropriate functions for processing the transactions submitted by the mobile clients. Concurrency control and recovery protocols must be modified to account among others for the mobility and the occasional disconnections of the clients, and the unsafe storage of the mobile unit, while the two-phase commit protocol seems to be applicable as is. If the client/agent/server model is used we might be able to take advantage of the presence of the agent to improve the throughput of the system by minimizing, for example, the number of aborted transactions due to disconnections or weak connectivity. For example, in [YeZa94], where the whole-transaction approach is taken, each mobile client submits a transaction to a coordinating agent. Once the transaction has been submitted, the coordinating agent schedules and coordinates its execution on behalf of the mobile client. Any disconnections that might occur between the invocation of the request and the agent's response have no blocking effect on the client.

Heavy-weight transactional mobile client

In the case of heavy-weight clients, the employment of the client/intercept/server model seems to be more beneficial. We consider first the case in which the mobile client does not own any transactional data. During disconnection, when the mobile client identifies bandwidth degradation, the client intercept cooperates with the server intercept in caching all needed data on the mobile host to allow the computation to continue. A common trend is for the client intercept to tentatively commit transactions executed at the disconnected mobile unit and make their results visible to subsequent transactions in the same unit [Ant95, SaKME93, GKLSL95, Dem95, Wolf95, LuSa94, LuSa95]. Tentative commitment records an intention to commit and allow the client to continue with the next transaction. Upon reconnection, a certification process takes place between the intercept agents, during which the execution of any tentatively committed transaction is validated against an application or system defined correctness criterion. If the criterion is met, the transaction is committed. Otherwise, the execution of the transaction must be aborted, reconciled or compensated. Such actions may have cascaded effects on other tentatively committed transactions that have seen the transaction's results. Reconciling tentative committed data with the fixed server is itself a distributed transaction that needs to be committed. How the actual commit processing is

achieved has not yet been addressed. The intercept agents of the model are the ones responsible for both the certification and commitment processes.

A different approach to the role of the mobile host is to allow data to be stored at the mobile host. Such an approach is necessary to allow autonomous operation during disconnection but complicates data management and may cause unacceptable communication overheads. Placing data at the mobile host raises physical database design issues. Such issues include how to appropriately fragment the database and how to allocate fragments at fixed and mobile hosts. When a fragment is allocated, it may either be replicated at selected or at all sites or maintained as a single copy. In general, maintaining a single copy of a data item at a mobile host is generally inappropriate for reasons of availability and reliability.

Concurrency control in the case of distributed transactions that involve both mobile and fixed hosts is complicated. For transactions that access data at both mobile and stationary hosts accessing the wireless link impose large overheads. Take for instance, the case of a pessimistic concurrency control protocol that requires transactions to acquire locks at multiple sites. In this case, transactions may block if they request locks at sites that get disconnected or if they request locks held by transactions at disconnected sites. On the other hand, techniques such as timestamps may lead to a large number of transactions being aborted because operations may be overly delayed in slow networks. To avoid delays imposed by the deployment of slow wireless links, open-nested transaction models are more appropriate [Chry93]. According to these models, a mobile transaction that involves both stationary and mobile hosts is not treated as one atomic unit but rather as a set of relatively independent component transactions some of which run solely at the mobile host. Component transactions can commit without waiting for the commitment of other component transactions.

In particular, as in the previous approach, transactions that run solely at the mobile host are only tentatively committed at the mobile host by the client intercept and their results are made visible only to subsequent local transactions. These transactions are certified at the fixed hosts, i.e., checked for correctness, at a later time. Fixed hosts can broadcast to mobile hosts information about other committed transactions prior to the certification event, as suggested in [Barb97]. This information can be used to reduce the number of aborted transactions. Transactions that run solely at the mobile host are called *weak* in [PiBh95b, Pit96], while the rest are called *strict*. A distinction is drawn between weak copies and strict copies. In contrast to strict copies, weak copies are only tentatively committed and hold possibly obsolete values. Weak transactions update weak copies, while strict transactions access strict copies located at any site. Weak copies are integrated with strict copies either when connectivity improves or when an application-defined limit to the allowable deviation among weak and strict copies is passed. Applications at weakly connected sites may chose to issue strict transactions when they require strict consistency. Strict transactions are slower than weak transactions since they involve the wireless link but guarantee permanence of updates and currency of reads. During disconnection, applications can only use weak transactions. In this case, weak transactions have similar semantics with *second-class IOTs* [LuSa95] and *tentative transactions* [GrHOS96].

Limitations of the classical 2PC, such as blocking [Skee81, SaBCM95], occur more often and thus solutions, such as heuristic processing (i.e., unilateral committing or aborting the transaction) [SaNi95], to these problems are now more eagerly required. Commit processing

can take advantage of the broadcasting characteristic of the wireless link to optimize its performance. Commit optimizations [SaBCM95] can also be adjusted to deal with mobile transactions. For example, while the mobile client can initiate commit processing it might be prudent to always transfer, through the *last-agent* commit optimization, commit coordination to a more reliable partner, e.g., the server on the fixed network.

In summary, the functionality of the agents of the intercept model in terms of supporting heavy-weight transactional clients is multi-faceted. To name just a few of their roles, the intercept agents cooperate with each other to support caching for weak transactions, provide routines for reconciling mobile transactions with the fixed servers, hide disconnection, and participate in specific commit optimizations.

7. Conclusions

The inherent limitations of the wireless link and the additional complexity imposed by mobility have a great impact on the design and structure of mobile computing applications and motivate the development of new computing models. In this paper, we demonstrated the inability of the traditional client/server paradigm to handle the new requirements and presented a taxonomy of computational models that are specific to mobile computing. These models either build on the traditional client/server model by providing appropriate extensions to it or are based on the new paradigm of mobile agents. These two types of models are not orthogonal, however, but they can be effectivelly integrated to provide additional flexibility. The paper identifies the challenges that a mobile computing model must meet namely mobility, disconnections, weak connectivity, adaptivity, upward compatibility with old and legacy applications and variable types of mobile units. The models are comparatively discussed within this framework. The capabilities and shortcomings of the various models are also demonstrated via a Web Browsing application which is modeled after each one of the models presented.

Mobile computing introduces numerous issues. To construct reliable and robust systems, appropriate frameworks for systematically structuring and designing software applications must be derived. The presented models provide such a framework.

References

[AlKo93] R. Alonso and H. F. Korth, Database System Issues in Nomadic Computing, *Proceedings of the 1993 SIGMOD Conference*, Washington, D.C., May, 1993.

[Ant95] Anthony D. Joseph, et al., Rover: A Toolkit for Mobile Information Access, *Proc. of the Fifteenth Symposium on Operating Systems Principles*, December 1995.

[ARDIS92] ARDIS, ARDIS Network Connectivity Guide, Illinois, ARDIS, March, 1992.

[ARDIS97] *What is ARDIS/DataTAC*, Research in Motion, September 1997. Available at <http:// www.rim.net/networks.html>.

[ARPA97] ARPA Knowledge Sharing Effort, www.cs.umbc.edu/agents/kse/.

[AtDu93] A. Athan and D. Duchamp, Agent-Mediated Message Passing for Constrained Environments, Proceedings *USENIX Symposium on Mobile and Location-Independent Computing*, pp. 103--1070, Cambridge, Massachusetts, August, 1993.

[BaBIM93] B. R. Badrinath and A. Bakre and T. Imielinski and R. Marantz, Handling Mobile Clients: A Case for Indirect Interaction, *Proceedings of the 4th Workshop on Workstation Operating Systems*, Aigen, Austria, October, 1993.

[BandS97] Bandwidth Conservation Society, Bandwidth Conservation Society's Homepage, www.infohiway.com/faster/.

[BaBa97] A. Bakre and B.R. Badrinath, Implementation and Performance Evaluation of Indirect TCP, *Journal IEEE Transactions on Computers*, No. 3, Vol. 46, March, 1997.

[BaBa96] A. Bakre and B.R. Badrinath, Reworking the RPC Paradigm for Mobile Clients, *Journal ACM/Baltzer Journal on Mobile Networks and Applications*, Vol. 1 pp. 371-385, 1996.

[Barb97] D. Barbara, Certification Reports: Supporting Transactions in Wireless Systems, *Proceedings of the IEEE International Conference on Distributed Computing Systems*, 1997.

[BaIm94] D. Barbara and T. Imielinski, Sleepers and Workaholics: Caching Strategies in Mobile Environments, *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 94)*, pp. 1-12, 1994.

[BeCo95] T. Berners-Lee and D. Connolly, Hypertext Markup Language Specification 2.0, Internet Draft, Internet Engineering Task Force (IETF), HTML Working Group. Available at www.ics.uci.edu/pub/ietf/html/html2spec.ps.gz, June, 1995.

[BeFF95] T. Berners-Lee and R. Fielding and H. Frystyk, Hypertext Transfer Protocol - HTTP/1.0 Specification, Internet Draft, Internet Engineering Task Force (IETF). Available at www.ics.uci.edu/pub/ietf/http/draft-fielding-http-spec-01.ps.Z, August, 1995.

[BCLNS94] T. Berners-Lee and R. Caililiau and A. Luotonen and H.F. Nielsen and A. Secret, The World-Wide Web, *Journal Communications of the ACM*, Vol. 37, No. 8, pp.76-82, August, 1194.

[Bar95] J. F. Bartlett, Experienece with Wireless World Wide Web Clients, *Proceedings of the IEEE COMPCON*, San Francisco, CA, March, 1995.

[ChGHLPT95] D. Chess and B. Grosof and C. Harrison and D. Levine and C. Parris and G. Tsudik, Itinerant Agents for Mobile Computing, *Journal IEEE Personal Communications*, Vol. 2, No. 5, October, 1995.

[Chry93] P. K. Chrysanthis, Transaction Processing in Mobile Computing Environment, *Proceedings of the IEEE Workshop on Advances in Paralleland Distributed Systems*, pp. 77-83, Princeton, New Jersey, October, 1993.

[Dem95] Alan Demers, et al., The Bayou Architecture: Suport for Data Sharing among Mobile Users. *Proc. Workshop on Mobile Computing Systems and Applications*, IEEE, December, 1994.

[FiGMFB97] R. Fielding, Jim Gettys, Jeffrey C. Mogul, H. Frystyk and T. Berners-Lee, *Hypertext Transfer Protocol - HTTP/1.1*. RFC 2068, HTTP working Group, Janauary, 1997.

[FoZa94] G. H. Forman and J. Zahorjan, The Challenges of Mobile Computing, *IEEE Computer*, Vol. 27, No. 6, pp. 38-47, April, 1994.

[FoGBA96] A. Fox and S. D. Gribble and E. A. Brewer and E. Amir, Adapting to Network and Client Variability via On-Demand Dynamic Distillation, *Proceedings of the ASPLOS-VII*, Cambridge, MA, October, 1996.

[FoBr96] A. Fox and E. A. Brewer, Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation, *Proceedings of the 5th International World Wide Web Conference*, Paris, France, May, 1996.

[GKLSL95] Gruber R., F. Kaashoek, B. Liskov, L. Shrira, Disconnected Operations in the Thor Object-Oriented Database System. *Proc. Mobile Computing Systems and Applications,* IEEE, Los Alamitos, CA, USA, p 51-56, 1995.

[GMAGIC97]General Magic's Telescript, www.genmagic.com/agents.

[Satya96a], M. Satyanarayanan, Mobile Information Access, *Journal IEEE Personal Communications*, Vol. 3, No. 1, February, 1996.

[Gray78] J. N. Gray, Notes on Data Base Operating Systems, Operating Systems - An Advanced Course, Spinger-Verlag, LNCS 60, (editor) B. R. Graham and G. Seegmuller, 1978.

[GrRe93] Gray, J., and Reuter A., *Transaction Processing: Concepts and Techniques*. Morgan Kaufman Publishers, Inc., 1993.

[GrHOS96] J. Gray and P. Helland and P. O' Neil and D. Shasha, The Dangers of Replication and a Solution, *Proceedings of the ACM SIGMOD Conference*, pp. 173-182, Montreal, Canada, 1996.

[HoSaL97] B C. Housel and G. Samaras and D. B. Lindquist, WebExpress: A Client/Intercept Based System for Optimizing Web Browsing in a Wireless Environment, *Journal ACM/Baltzer Mobile Networking and Applications (MONET), Special Issue on Mobile Networking on the Internet*. To appear. Also, University of Cyprus, CS-TR 96-18, December 1996.

[IBM95] IBM, An Introduction to Wireless Technology, *IBM International Technical Support Center SG24-4465-01*, October, 1995.

[IBM95a] *ARTour Technical Overview Release 1*, IBM Corp. SB14-0110-0, March 1995.

[ImBa94c] T. Imielinksi and B. R. Badrinath, Wireless Mobile Computing: Challenges in Data Management, *Journal Communications of the ACM*, Vol. 37, No. 10, October, 1994.

[IoMa93] J. Ioannidis and G. Q. Maguire Jr, The Design and Implementation of a Mobile Internetworking Architecture, *Proceedings of the 1993 Winter Usenix*, pp. 491-502, San Diego, California , January, 1993.

[JoTK97] A. D. Joseph and J. A. Tauber and M. F. Kaashoek, Mobile Computing with the Rover Toolkit, *Journal IEEE Transactions on Computers*, February, 1997.

[JiBE95] J. Jing and O. Bukhres and A. Elmagarmid, Distributed Lock Management for Mobile Transactions, *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems*, May, 1995.

[KaPT94] M. Frans Kaashoek and Tom Pinckney and Joshua A. Tauber, Dynamic Documents: Mobile Wireless Access to the WWW, *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December, 1994.

[Katz94] R. H. Katz, Adaptation and Mobility in Wireless Information Systems, *Journal IEEE Personal Communications*, Vol. 1, pp. 6-17, 1994.

[Lamp81] B. W. Lampson, Atomic Transactions, Distributed Systems: Architecture and Implemementation, Spinger-Verlag, LNCS 105, (editor) B. W. Lampson, pp. 246-285, 1981.

[LiAKLR95] M. Liljberg and T. Alanko and M. Kojo and H. Laamanen and K. Raatikainen, Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Apprach, *Proceedings of the SDNE*, Whistler, Canada, June, 1995.

[LuSa94] Q. Lu and M. Satyanarayanan, Isolation-Only Transactions for Mobile, Operating Systems Review, pp 81-87, April, 1994.

[LuSa95] Q. Lu and M. Satyanarayanan, Improving Data Consistency in Mobile Computing Using Isolation-OnlyTransactions, *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, Washington, May, 1995.

[MAge97] Mobile Agents, www.agent.org/ and www.cs.umbc.edu/agents.

[Math94] C. J. Mathias, New LAN Gear Snaps Unseen Desktop Chains, Data Communications, volume 23, No. 5, pp.75-80, March, 1994.

[Miller94] K. Miller, Cellular Essentials for Wireless Data Transmission, *Data Communications*, Vol. 23, No. 5, pp. 61-67, March, 1994.

[NaSe96] S. Narayanaswamy and S. Seshan and et. al, Application and Network Support for InfoPad, *Journal IEEE Personal Communications Magazine*, March, 1996.

[NoPS96] B. D. Noble and M. Price and M. Satyanarayanan, A Programming Interface for Application-Aware Adaptation in Mobile Computing, *Journal Computing Systems*, Winter, Vol. 8, No. 4, 1996.

[Ora97] Oracle, Oracle Mobile Agents Technical Product Summary, www.oracle.com/products/networking/mobile/_agents/html/, June, 1997.

[PiSa98] E. Pitoura and G. Samaras, *"Data Management for Mobile Computing"*, Kluwer Academic Publishers, ISBN 0-7923-8053-3, 1998.

[Pit96] E. Pitoura, A Replication Schema to Support Weak Connectivity in Mobile Information Systems, *Proceedings of the 7th International Conference on Database and Expert Systems Applications (DEXA96),* LNCS 1134, Springer Verlag, pp. 510--520, September, 1996.

[PiBh95b] E. Pitoura and B. Bhargava, Maintaining Consistency of Data in Mobile Distributed Environments, Proceedings of the 15th IEEE International Conference on Distributed Computing Systems, pp. 404-413, May, 1995.

[PiBh95a] E. Pitoura and B. Bhargava, A Framework for Providing Consistent and Recoverable Agent-Based Access to Heterogeneous Mobile Databases, *Journal ACM SIGMOD Record*, Vol. 24, No. 3, pp.44-49, September, 1995.

[RaMo95] V. N. Radmanabhan and J. C. Mogul, Improving HTTP Latency, *Journal Computer Networks and ISDN Systems*, Vol. 28, No. 1, December, 1995.

[RAM97] Mobitex Features and Services, *RAM Mobile Data White Paper*, February 1997. Available at <http://www.ram-wireless. com/new/white/mobitex2.html>.

[RAM94] RAM, RAM Mobile Data System Overview, RAM Mobile Data Limited Partnership USA RMDUS 031-RMDSO-RM Release 5.2, October, 1992.

[RePGSR96] P. Reiher and J. Popek and M. Gunter and J. Salomone and D. Ratner, Peer-to-Peer Reconciliation Based Replication for Mobile Computers, *Proceedings of the European Conference on Object Oriented Programming 2nd Workshop on Mobility and Replication*, June, 1996.

[SaP97] G. Samaras and A. Pitsillides, Client/Intercept: a Computational Model for Wireless Environments, *Proceedings of the 4th International Conference on Telecommunications (ICT'97)*, Melbourne, Australia, April, 1997.

[SaBCM95] Samaras G., K. Britton, A. Citron, C. Mohan, "Two-Phase Commit Optimizations in a Commercial Distributed Environment", *Distributed and Parallel Databases Journal*, 3(4): 325-361, October, 1995.

[SaNi95] Samaras, G., S.D. Nikolopoulos, "Algorithmic Techniques Incorporating *Heuristic Decisions in Commit Protocols*". *Proc. 25th Euromicro Conference,* IEEE/CS, Como, September 1995.

[Skee81] Skeen, D., Nonblocking Commit Protocols, *Proc. ACM/SIGMOD International Conference on Management of Data*, pp. 133-142, Ann Arbor, Michigan, 1981.

[SaKME93] M. Satyanarayanan, M., Kistler, , L. Mummert, M. R. Ebling, P. Kumar, and Q. Lu. Experiance with Disconnected Operations in a mobile Computing Environment. *Proc. 1993 Usenix Symposium on Mobile and Location Independent Computing*, CA, November, pp 11-28, Cambridge, MA, August 1993.

[Satya96b] M. Satyanarayanan, Fundamental Challenges in Mobile Computing, *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, Philadelphia, PA, May, 1996.

[ScAGTW93] B. N. Schilit and N. Adams and R. Gold and M. Tso and R. Want, The ParcTab Mobile Computing System, *Proceedings of the 4th IEEE Workshop on Workstation Operating Ssytems (WWOS-IV),* pp. 34-39, October, 1993.

[TeSSM96] D. L. Tennenhouse and J. M. Smith and W. D. Sincoskie and G. J. Minden, A Survey of Active Network Research, *Journal IEEE Communication Magazine*, Vol. 35, No. 1, pp. 80--86, January, 1996.

[YeZa94] L. H. Yeo and A. Zaslavsky, Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment, *Proceedings of the 14th International Conference onDistributed Computing Systems*, Poznan, Poland, June, 1994.

[VoBe94] G. M. Voelker and B. N. Bershad, Mobisaic: An Information System for a Mobile Wireless Computing Environment , *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, pp. 185-190, December, 1994.

[ZeDu97] B. Zenel and D. Duchamp, General Purpose Proxies: Solved and Unsolved Problems, Proceedings of the Hot-OS VI, 1997.

[ZeDu95] B. Zenel and D. Duchamp, Intelligent Communication Filtering for Limited Bandwidth Environments, Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems (HOT-OS V), Rosario WA, May, 1995.

[White96] J. E. White, Mobile Agents, General Magic White Paper, www.genmagic.com/agents, 1996.

[Wolf95] O. Wolfson, Prasad Sistal, Son Dao, Kailash Narayanan, Ramya Raj, View Maintenance in Mobile Computing, Sigmod Records, 1995.