Contents

X-XXXX-XXXX-X/XX/$X.XX+$X.XX

1

Data and Services for Mobile Computing

Sasikanth Avancha, *University of Maryland Baltimore County*

Dipanjan Chakraborty, *University of Maryland Baltimore County*

Filip Perich, *University of Maryland Baltimore County*

Anupam Joshi, *University of Maryland Baltimore County*

CONTENTS

Abstract The advent and phenomenal growth of low-cost, lightweight, portable computers concomitant with that of the Internet has led to the concept of Mobile Computing. Protocols and mechanisms used in Internet computing are being modified and enhanced to adapt to *mobile* computers. New protocols and standards are also being developed to enable mobile computers to connect to each other and the Internet through both wired and wireless interfaces. The primary goal of the mobile computing paradigm is to enable mobile computers accomplish tasks using all possible resources, i.e., data and services, available in the network, anywhere, anytime. In this chapter we survey the state-of-the-art of mobile computing and its progress toward its goals. We also present a comprehensive, flexible framework to develop applications for mobile computing. The framework consists of protocols, techniques and mechanisms that enable applications to discover and manage data and services in wired, infrastructure supported wireless and mobile ad hoc networks.

X-XXXX-XXXX-X/XX/$X.XX+$X.XX

1.1 Introduction

The term "computing device" or "computer" usually evokes the image of a big, powerful machine, located in an office or home, that is always on and possibly connected to the Internet. The rapid growth of lightweight, easily and constantly available devices, even when one is on the move, has dramatically altered this image. Coupled with the potential for easy network access, the growth of these *mobile* devices has tremendously increased our capability to take computing services with us, wherever we go. The combination of device mobility and computing power has resulted in the Mobile Computing paradigm. In this paradigm computing power is constantly at hand, irrespective of whether the mobile device is connected to the Internet or not. The smaller the devices, the greater their portability and mobility, but the lesser their computing capability. It is important to understand that the ultimate goal of the mobile computing paradigm is enabling people to accomplish tasks using computing devices, *anytime*, *anywhere*. To achieve this goal, network connectivity will become an essential part of mobile computing devices. The underlying network connectivity in mobile computing is typically wireless. *Portable Computing* is a variant of mobile computing that includes the use of wired interfaces (e.g., a telephone modem) of mobile devices. For instance, a laptop equipped with both a wireless and a wired interface connects via the former when the user is walking down a hallway (mobile computing), but switches to the latter when the user is in her office (portable computing).

The benefits of mobility afforded to computing devices are greatly reduced, if not completely eliminated, if devices can only depend on a wired interface for their network connectivity (e.g., telephone/network jack). It is more useful for a mobile computing device to use wireless interfaces for network connectivity when required. Additionally, networked sources of information may also become mobile. This leads to a related area of research called *ubiquitous computing*.

Let us now discuss the hardware characteristics of current-generation mobile computing devices. The emphasis in designing mobile devices is to conserve energy and storage space. These requirements are evident in the following characteristics, which are of particular interest to mobile computing:

Size, form factor, and weight. Mobile devices, with the exception of high-end laptops, are handhelds (e.g., cell phones, PDAs, pen computers, tablet PCs). They are lightweight and portable. Mobility and portability of these devices are traded off with greater storage capacity and higher processing capability.

Microprocessor. Most current-generation mobile devices use low-power microprocessors, such as the family of ARM and XScale processors, in order to conserve energy. Thus, high performance is traded off for energy consumption, because the former is not as crucial to mobile devices as the latter.

Memory size and type. Primary storage sizes in mobile devices range anywhere between 8 MB and 64 MB. Mobile devices may additionally employ flash ROMs for secondary storage. Higher-end mobile devices, such as pen computers, use hard drives with sizes in the order of Gigabytes. In mid-range devices, such as the iPAQ or Palm, approximately half of the primary memory is used for the kernel and operating system leaving the remaining for applications. This limited capacity is again used to trade better performance off with lower energy consumption.

Screen size and type. The use of LCD technology and viewable screen diagonal lengths between 2" and 10" are common characteristics of mobile devices. The CRT technology used in desktop monitors typically consumes approximately 120W, whereas the LCD technology used in PDAs consumes only between 300 and 500 mW. As with the other characteristics, higher screen resolution is traded off for lower power consumption; however, future improvements in LCD technology may provide better resolution at little or no increase in power consumption.

Input mechanism(s). The most common input mechanisms for mobile devices are built-in keypads, pens and touch screen interfaces. Usually, PDAs contain software keyboards; newer PDAs may also support external keyboards. Some devices also use voice as an input mechanism. Mobility and portability of devices are primary factors in the design of these traditional interfaces for cell phones, PDAs and pen computers. Human-Computer interaction (HCI) is a topic of considerable research and impacts the marketability of a mobile device. For example, a cell phone that could also be used as a PDA should not require user input via keys or buttons in the PDA mode; rather it should accept voice input.

Communication Interface(s). As discussed above, mobile devices can support both wired and wireless communication interfaces, depending on their capabilities. We shall concentrate on wireless interfaces in this context. As far as mobile devices are concerned, wireless communication is either short-range or long-range. Short-range wireless technologies include infrared (IR), Global System for Mobile Communications (GSM), IEEE 802.11b, and Bluetooth. IR, which is part of the optical spectrum, requires line-of-sight, communication while the other three, which are part of the radio spectrum, can function as long as the two devices are in radio range and do not require line-of-sight. Long-range wireless technologies include satellite communications which are also part of the radio spectrum. While wireless interfaces provide network connectivity to mobile devices, they pose some serious challenges as compared to wired interfaces. Frequent disconnections, low and variable bandwidth, and most importantly, increased security risks are some of these challenges.

The discussion thus far clearly suggests that mobile computing is not limited to the technical challenges of reducing the size of the computer and adding a wireless interface to it. It encompasses the problems, and their solutions, associated with enabling people use the computing power of their devices at anytime, anywhere, possibly with network connectivity.

1.2 Mobile Computing vs. Wired-Network Computing

We now compare mobile computing and wired-network computing from the network perspective. For the purposes of this discussion, we consider only the wireless networking aspect of mobile computing. We shall also use the terms wired-network computing and wired computing interchangeably. We shall compare mobile computing and wired computing based on layers 1 through 4 of the standard 7-layer Open Standards Interconnection (OSI) stack. Figure 1.1 shows the Physical, Data Link (comprising the Link Management and Medium Access Control sub-layers), Network, and Transport layers of the two stacks.

The Physical layer: In the network stack for mobile computing, the physical layer consists of two primary "media" – the radio spectrum and the optical spectrum. The radio spectrum is divided into licensed and unlicensed frequency bands. Cellular phone technologies use the licensed bands whereas technologies such as Bluetooth and IEEE 802.11b use the unlicensed band. The optical spectrum is mainly used by infrared devices. The network stack for wired computing consists of cable technologies such as co-axial cable and optical fiber.

The Medium Access Control (MAC) sub-layer: The most frequently adopted MAC mechanism in the wired computing network stack is the well-known Carrier Sense Multiple Access with Collision Detection (CSMA/CD). It is also well-known that CSMA/CD cannot be directly applied to the mobile computing stack because it would cause collisions to occur at the receiver as opposed to the sender. To prevent this situation, researchers and practitioners have designed different mechanisms based on collision avoidance and synchronizing transmissions. CSMA with Collision *Avoid-*

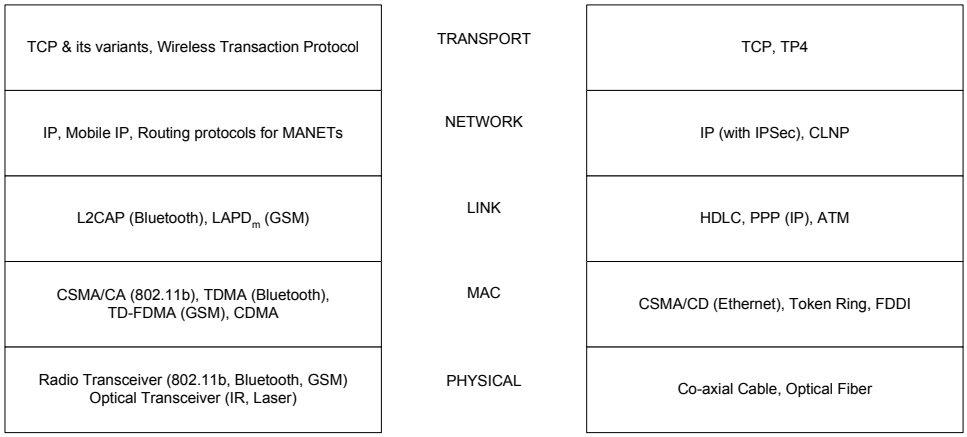

Figure 1.1: Network Stack Comparison of Mobile and Internet Computing

ance (CSMA/CA) helps transmitters determine if other devices around them are also preparing to transmit and therefore avoid collisions by deferring transmission. Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA) and Digital Sense Multiple Access with Collision Detection (DSMA/CD) are other popular MAC protocols used by mobile device network stacks to co-ordinate transmissions.

The Link Management sub-layer: This layer is present in only a few network stacks of mobile devices. For example, the IEEE 802.11b standard describes only the Physical and MAC layers as part of the specification. Some of the link management protocols on mobile device network stacks are required to handle voice connections (usually connection-oriented links), in addition to primarily connection-less data links. The Logical Link Control and Adaptation Protocol (L2CAP) in Bluetooth is an example of such a protocol. GSM uses a variant of the well known Link Access Protocol D-channel (LAPD) called LAPD$_m$. High-level Data Link Control (HDLC), Point-to-Point Protocol (PPP), and Asynchronous Transfer Mode (ATM) are the most popular data link protocols used in wired networks.

The Network layer: Mobility of devices introduces a new dimension to routing protocols, which reside in the network layer of the OSI stack. Routing protocols for mobile networks, both ad-hoc and infrastructure supported, need to be aware of mobile device characteristics such as mobility and energy consumption. Unlike static devices, mobile devices cannot always depend on a static address, such as an IP address. This is because they need to be able to attach to different points to the network, public or private. Routing protocols such as Mobile IP [Perkins, 1997], enable devices to dynamically obtain an IP-address and connect to any IP-based network, while they are on the move. This solution requires the existence of a central network (i.e., home network), that tracks the mobile device and knows its current destination network. Routers are the linchpins of the Internet. They decide how to route incoming traffic based on addresses carried by the data packets. In ad-hoc networks, no static routers exist. Many nodes in the network may have to perform the routing function, because the "routers" may be mobile and thus move in and out of range of senders and receivers. All of these considerations have focused research on developing efficient routing protocols in mobile ad hoc networks (MANET).

The Transport layer: TCP has been the protocol of choice for the Internet. TCP performs very well on wired networks, such as the Internet, which have high bandwidth and low delay. However, research on TCP performance over wireless networks has shown that it typically fails if non-

congestion losses (losses due to wireless channel errors or client mobility) occur on the wireless link. This is because TCP implicitly assumes that all losses are due to congestion and reduces the window on the sender. If the losses are not due to congestion, then TCP unnecessarily reduces throughput leading to poor performance. Solutions to this problem include designing new transport protocols, such as CentaurusComm [Sasikanth Avancha and Vladimir Korolev and Anupam Joshi and Timothy Finin and Y. Yesha, 2002], that are more mobile-aware and modifying TCP to make it more mobile-aware. Modified versions of TCP [Barke and Badrinath, 1995; Brown and Singh, 1997; Goff et al., 2000] are well-known in the research community. The Wireless Transaction Protocol (WTP) is part of the well-known Wireless Application Protocol (WAP) stack and provides reliable data transmission using retransmission, segmentation and reassembly as required.

Both, academia and industry, have contributed significantly to mobile computing research aimed at designing the best possible network stack that takes into account the challenges of reduced computer size and computing power, energy conservation, and low-bandwidth, high-delay wireless interfaces. The discussion in this section provides a glimpse of the solutions applied, to the most significant layers of the wired-network stack, to address these challenges.

1.3 M-Services Application Architectures

Mobile computing applications can be classified into three categories – client-server, client-proxy-server and peer-to-peer – depending on the interaction model. Evolution of mobile applications started from common distributed object-oriented systems like CORBA and DCOM [Sessions, 1997] which primarily follow client-server architecture. The emergence of heterogeneous mobile devices with varying capabilities has subsequently popularized the client-proxy-server architecture. Increasing computational capabilities of mobile devices and the emergence of ad-hoc networks is leading to a rapid growth of peer-to-peer architectures, similar to Gnutella in the Internet.

In the client-server architecture, a large number of mobile devices can connect to a small number of servers residing on the wired network, organized as a cluster. The servers are powerful machines with high bandwidth wired network connectivity and the capability to connect to wireless devices. Primary data and services reside on and are managed by the server, while clients locate servers and issue requests. Servers are also responsible for handling lower level networking details, such as disconnection and retransmission. The advantages of this architecture are simplicity of the client design and straightforward cooperation among cluster servers. The main drawbacks of this architecture are the prohibitively large overhead on servers to handle each mobile client separately, in terms of transcoding and connection handling, severely affecting system scalability.

In the client-proxy-server architecture, a proxy is introduced between the client and the server, typically on the edge of the wired network. The logical end-to-end connection between each server and clients is split into two physical connections, server-to-proxy and proxy-to-client. This architecture increases overall system scalability because servers only interact with a fixed number of proxies, which handle transcoding and wireless connections to the clients. There has been substantial research and industry efforts [Brooks et al., 1995; Zenel, 1995; Bharadvaj et al., 1998; Joshi et al., 1996] in developing client-proxy-server architectures. Additionally, intelligent proxies [Pullela et al., 2000] may act as a computational platform for processing queries on behalf of resource-limited mobile clients.

Transcoding, i.e., conversion of data and image formats to suit target systems, is an important problem introduced by client-server and client-proxy-server architectures. Servers and proxies are powerful machines that can handle data formats of any type and image formats of high resolution. However, mobile devices cannot. Therefore, data on the wired network must be transcoded to suit different mobile devices. It is therefore important for the server or proxy to recognize the characteristics of a client device. Standard techniques of transcoding, such as those included in the

WAP stack, include XSLT [Muench and Scardina, 2001] and Fourier transformation. The W3C CC/PP standard [Klyne et al., 2001] enables clients to specify their characteristics when connecting to HTTP servers using profiles.

In the peer-to-peer architecture, all devices, mobile and static, are peers. Mobile devices may act servers and clients. Ad-hoc network technologies such as Bluetooth allow mobile devices to utilize peer resources in their vicinity in addition to accessing servers on the wired network. Server mobility may be an issue in this architecture and thus, the set of services available to a client is not fixed. This may require mobile devices to implement service discovery [Rekesh, 1999; Chakraborty et al., 2002a], collaboration and composition [Chakraborty et al., 2002b; Mao et al., 2001]. The advantage of this architecture is that each device may have access to more up-to-date location dependent information and interact with peers without infrastructure support. The disadvantage of this architecture is the burden on the mobile devices in terms of energy consumption and network traffic handling.

Client-server and client-proxy-server architectures remain the most popular models of practical use from both, commercial and non-commercial, perspectives. Both these architectures provide users with certain guarantees, such as connectivity, fixed bandwidth and security, because of the inherent power of the proxies and servers. From a commercial perspective, they guarantee increased revenues to infrastructure and service providers, as the number of wireless users increases. Peer-to-peer architectures, which truly reflect the goal of anytime, anywhere computing, are largely confined to academia, but possess the potential to revolutionize mobile computing in the decades to come.

1.4 Mobile Computing Application Framework

In this section, we describe a comprehensive framework for enabling the development of a mobile application using one of the three architectures described above. Figure 1.2 depicts the different components of the framework. Depending on the selected model, some of the components may not be required to build a complete mobile application. However, other components, such as the communications layer, form an intrinsic part of any mobile application. The design of this framework takes into consideration such issues. We describe the different layers and components in the framework in the next few subsections.

1.4.1 Communications Layer

The communications layer in this framework encompasses the physical, MAC, link, network and transport layers of the mobile computing stack illustrated in Figure 1.1. This layer is responsible for establishing and maintaining logical end-to-end connections between two devices, and for data transmission and reception.

The physical and MAC layers are primarily responsible for node discovery, and establishment and maintenance of physical connections between two or more wireless entities. These functions are implemented in different ways in different technologies. For example, in Bluetooth, node discovery is accomplished through the use of the *inquiry* command by the baseband (MAC) layer. In IEEE 802.11b, the MAC layer employs the RTS-CTS (i.e., Request-To-Send and Clear-To-Send) mechanism to enable nodes to discover each other, when they are operating in the *ad hoc* mode. When IEEE 802.11b nodes are operating in *infrastructure* mode, the base station broadcasts beacons which the former use, to discover the base station and establish physical connections with it. The establishment of physical connections is a process in which the nodes exchange operational parameters such as baud rate, connection mode (e.g.: full-duplex or half-duplex), power mode (e.g.: low-power, high-power) and timing information for synchronization, if required. In order to maintain the connection, some or all of these parameters are periodically refreshed by the nodes.

The link layer may not be part of the specifications of all wireless technologies. Some, such as IEEE 802.11b, use existing link layer protocols such as HDLC or PPP (for point-to-point con-

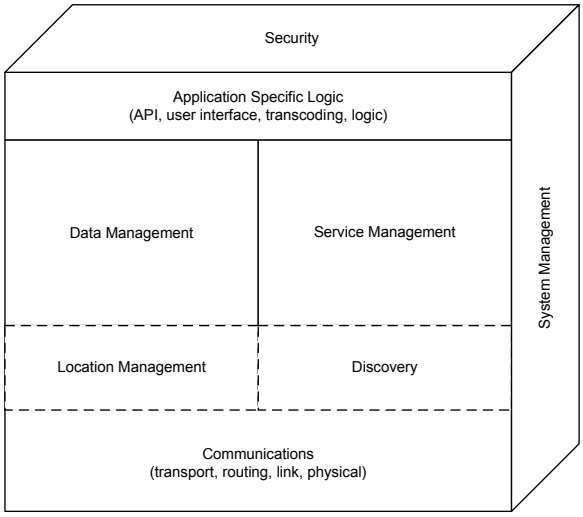

Figure 1.2: Mobile Computing Application Framework

nections) to establish data or voice links between the nodes. Bluetooth, on the other hand, uses a proprietary protocol, L2CAP, for establishing and maintaining links. This protocol is also responsible for other common link-layer functions such as framing, error correction and quality-of-service. The task of the link layer in is more difficult in wireless networks than in wired networks because of the high probability of errors either during or after transmission. Thus, error correction at the link layer must be robust enough to withstand the high bit-error rate of wireless transmissions.

The network layer in mobile computing stacks must deal with device mobility, which may cause existing routes to break or become invalid with no change in other network parameters. Device mobility may also be the cause of packet loss. For example, if the destination device, to which a packet is already enroute, moves out of range of the network, then the packet must be dropped. Thus, both route establishment and route maintenance are important problems that the network layer must tackle. As the mobility of a network increases, so do route failures and packet losses. Thus, the routing protocol must be robust enough to either prevent route failures or recover from them as quickly as possible. In particular, routing protocols for mobile ad hoc networks have received considerable attention in the recent past. Many routing protocols for MANETs have been developed primarily for research purposes. These include Ad hoc On Demand Distance Vector (AODV) routing protocol, Dynamic Source Routing (DSR) and Destination-Sequenced Distance Vector (DSDV) routing protocol. However, for most applications that use some variant of the client-server model, the standard Internet Protocol (IP) is quite sufficient.

Mobile applications, unlike Internet applications, tend to generate or require small amounts of data (of the order of hundreds or at most thousands of bytes). Thus, protocols at the transport layer should be aware of the short message sizes, packet delays due to device mobility and non-congestion packet losses. TCP is ill-suited for wireless networks. Numerous variations of TCP and transport protocols designed exclusively for wireless networks, ensure that both ends of a connection agree that packet loss has occurred before the source retransmits the packet. Additionally, some of these protocols choose to defer packet transmission if they detect that current network conditions are unsuitable.

The functionality of the communications layer in this framework is usually provided by the operating system running on the mobile device. Therefore, the mobile application can directly invoke the lower level system functions via appropriate interfaces.

1.4.2 Discovery Layer

The discovery layer helps a mobile application discover data, services and computation sources. These may reside in the vicinity of the mobile device or on the Internet. Due to resource constraints and mobility, mobile devices may not have complete information about all currently available sources. The discovery layer assumes that the underlying network layer can establish a logical end-to-end connection with other entities in the network. The discovery layer provides upper layers with the knowledge and context of available sources.

There has been considerable research and industry effort in service discovery in the context of wired and wireless networks. Two important aspects of service discovery are the *discovery architecture* and the *service matching mechanism*.

Discovery architectures are primarily of two types: *lookup-registry based* and *peer-to-peer*. Lookup-registry based discovery protocols work by registering information about the source to a registry. Clients query this registry to obtain knowledge about the source (such as, its location, how to invoke it etc.). This type of architecture can be further subdivided into two categories: centralized registry-based and federated or distributed registry-based architectures. A centralized registry-based architecture contains one monolithic centralized registry whereas a federated registry-based architecture consists of multiple registries distributed across the network. Protocols such as Jini [Arnold et al., 1999], Salutation and Salutation-lite, UPnP [Rekesh, 1999], UDDI and Service Location Protocol [Veizades et al., 1997] are examples of a lookup-registry based architecture.

Peer-to-peer discovery protocols query each node in the network to discover available services on that node. These type of protocols treat each node in the environment equally in terms of functional characteristics. Broadcasting of requests and advertisements to peers is a simple, albeit inefficient, service discovery technique in peer-to-peer environments. Chakraborty et al. [2002a] describe a distributed peer-to-peer service discovery protocol using caching that significantly reduces the need to broadcast requests and advertisements. Bluetooth Service Discovery Protocol (SDP) is another example of a peer-to-peer service discovery protocol. In SDP, services are represented using 128-bit unique identifiers. SDP does not provide any information on how to invoke the service. It only provides information on the availability of the service on a specific device.

The service discovery protocols discussed in this section use simple interface, attribute or unique identifier based matching techniques to locate appropriate sources. Jini uses interface matching, SDP uses identifier matching, while the Service Location Protocol and Ninja Secure Service Discovery Systems, discover services using attribute-based matching. The drawbacks of these techniques include lack of rich representation of services, inability to specify constraints on service descriptions, lack of inexact matching of service attributes and lack of ontology support [Chakraborty et al., 2001]. Semantic matching is an alternative technique that addresses these issues. DReggie [Chakraborty et al., 2001] and Bluetooth Semantic Service Discovery Protocol (SeSDP) [Avancha et al., 2002] both use a semantically rich language, called DARPA Agent Markup Language (DAML), to describe and match both services and data. Semantic descriptions of services and data allow greater flexibility in obtaining a match between the query and the available information. Matching can now be inexact. This means that parameters such as functional characteristics, hardware and device characteristics of the service provider may be used in addition to service or data attributes to determine whether a match can occur.

1.4.3 Location Management Layer

Location management layer deals with providing location information to a mobile device. Location information dynamically changes with mobility of the device and is one of the components of context awareness. It can be used by upper layers to filter location-sensitive information and obtain location-specific answers to queries, e.g., weather of a certain area and traffic condition on a road. The current location of a device relative to other devices in its vicinity can be determined using

the discovery layer or the underlying communications layer. Common technologies use methods such as triangulation and signal strength measurements for location determination. GPS [Hofmann-Wellenhof et al., 1997] is a well known example of the use of triangulation based on data received from four different satellites. Cell phones use cell tower information to triangulate their position. On the other hand, systems such as RADAR [Bahl and Padmanabhan, 2000], used for indoor location tracking, work as follows. Using a set of fixed IEEE 802.11b base stations, the entire area is mapped. The map contains (x,y) co-ordinates and the corresponding signal strength of each base station at that co-ordinate. This map is loaded onto the mobile device. Now, as the user moves about the area, the signal strength from each base station is measured. The pattern of signal strengths from the stored map that most closely matches the pattern of measured signal strengths is chosen. The location of the user is that corresponding to the (x,y) co-ordinates associated with the stored pattern. Outdoor location management technologies have achieved technical maturity and have been deployed in vehicular and other industrial navigational systems. Location management, indoor and outdoor, remains a strong research field with the rising popularity of technologies such as IEEE 802.11b and Bluetooth.

The notion of location can be dealt with at multiple scales. Most "location determination" techniques actually deal with position determination, with respect to some global (lat/long) or local (distances from the "corner" of a room) grid. Many applications are not interested in the absolute position as much as they are in higher order location concepts (inside or outside a facility, inside or outside some jurisdictional boundary, distance from some known place, at a mountaintop, in a rain forest region etc.) Absolute position determinations can be combined with GIS type data to infer locations at other levels of granularity.

Expanding the notion of location further leads us to consider the notion of context. Context is any information that can be used to characterize the situation of a person or a computing entity[Dey and Abowd, 2000]. So for instance, context covers things such as location, device type, connection speed, direction of movement. Context even arguably involves a users mental states (beliefs, desires, intentions) etc. This information can be used by the layers described next for data and service management. However, the privacy issues involved are quite complex. It is not clear who should be allowed to gather such information, under what circumstances should it be revealed, and to whom. So for instance a user may not want her GPS chip to reveal her current location except to emergency response personnel. Some of these issues, specifically related to presence and availability, are being discussed in the PAM WG of PARLAY `http://www.parlay.org`. A more general formulation of such issues can be found in the recent work of Chen et. al. [Chen et al., 2003], who are developing OWL based policies and a Decision Logic based reasoner to specify and reason about a users privacy preferences as related to context information.

1.4.4 Data Management Layer

Data management layer deals with access, storage, monitoring, and data manipulation. Data may reside locally and also on remote devices. Similar to data management in traditional Internet Computing, this layer is essential in enabling a device to interact and exchange data with other devices located in its vicinity and elsewhere on the network. The core difference is that this layer must also deal with mobile computing devices. Such devices have limited battery power and other resources in comparison to their desktop counterparts. The devices also communicate over wireless logical links that have limited bandwidth and are prone to frequent failures. Consequently, the data management layer often attempts to extend data management solutions for Internet Computing by primarily addressing mobility and disconnection of a mobile computing device.

Work on data management can be classified along four orthogonal axes [M. Tamer Ozsu and Patrick Valduriez, 1999; Margaret H. Dunham and Abdelsalam (Sumi) Helal, 1995]: autonomy, distribution, heterogeneity, and mobility,. We can apply the classification to compare three architecture models adopted by existing data management solutions.

The client-server model is a two-level architecture with data distributed among servers. Servers are responsible for data replication, storage and update. They are often fixed and reside on the wired infrastructure. Clients have no autonomy as they are fully dependent on servers and may or may not be mobile and heterogeneous. This model was the earliest adopted approach for distributed file and database systems since it simplifies data management logic and supports rapid deployment [Satyanarayanan et al., 1990]. The model delegates all data management responsibility to only a small subset of devices, the servers. Additionally, the model addresses the mobility problem by simply not dealing with it or by using traditional timeout methods.

The client-proxy-server model extends the former approach by introducing an additional level in the hierarchy. Data remains distributed on servers residing on a wired infrastructure. Clients still depend on servers and may or may not be mobile and heterogeneous. However, a proxy, residing on the wired infrastructure, is placed between clients and servers. The proxy takes on a subset of server responsibility, including disconnection management, caching and transcoding. Consequently, servers no longer differentiate among mobile and fixed clients. They can treat all clients uniformly since they communicate with devices on wired infrastructure only. Proxy devices are then responsible for delivering data to clients and for maintaining sessions when clients change locations [Dunham et al., 1997].

The peer-to-peer model takes a completely different approach from the other two models. This model is highly autonomous as each computing device must be able to operate independently. There is no distinction between servers and clients and their responsibility. The model also lies in the extreme of the other three axes since data may reside on any device, and each device can be heterogeneous and mobile. In this model, any two devices may interact with each other [Perich et al., 2002]. Additionally, unlike client-server based approaches the model is open in that there is no strict set of requirements that each device must follow. This may cause the data management layer to be implemented differently on each device. Consequently, each peer must address both local and global data management issues; the latter is handled by servers or proxies in client-server based models.

Local data management, logically operating at the end-user level, is responsible for managing degrees of disconnection and query processing. The least degree of disconnection encourages the device to constantly interact with other devices in the environment. The highest degree represents the state when the device only utilizes its local resources. The mobility of a device can affect both the type of queries as well as the optimization techniques that can be applied. Traditional query processing approaches advocate *location transparency*. These techniques only consider aspects of data transfer and processing for query optimization. On the other hand, in the mobile computing environment, query processing approaches promote *location awareness* [Hans-Erich Kottkamp and Olaf Zukunft, 1998]. For example, a mobile device can ask for the location of the closest Greek restaurant, and the server should understand that the starting point of the search refers to the current position of the device [Perich et al., 2002; Olga Ratsimor and Vladimir Korolev and Anupam Joshi and Timothy Finin, 2001].

Global data management, logically operating at the architecture-level, deals with data addressing, caching, dissemination, replication and transaction support. As devices move from one location to another or become disconnected, it is necessary to provide a naming strategy, to locate a mobile station and its data. There have traditionally been three approaches for data addressing: *location-dependent*, *location-transparent*, and *location-independent*[Pinkerton et al., 1990; Sandberg et al., 1985]. To allow devices to operate disconnected, they must be able to cache data locally. This requirement introduces two challenges: *data selection* and *data update*. Data selection can be explicit [Satyanarayanan et al., 1990] or pro-actively inferred [Perich et al., 2002]. In the former approach, a user explicitly selects files or data that must be cached. The latter approach automatically predicts and pro-actively caches the required information. Data update of local replicas usually requires a weaker notion of consistency as the mobile device may have to operate on stale data without the knowledge that the primary copy was altered. This is especially the case when devices become dis-

connected from the network and cannot validate consistency of their data. Either subscription-based callbacks [Satyanarayanan et al., 1990] or latency- and recency-based refreshing [Laura Bright and Louiqa Raschid, 2002] can address this issue. In subscription-based approaches, a client requests the server to notify it (the client) when a particular datum is modified. In turn, when a server modifies its data it attempts to inform all clients subscribed to that data. In latter approaches, a client or proxy uses timestamp information to compare its local replicas with remote copies in order to determine when to refresh its copy.

Data dissemination models are concerned with read-only transactions where mobile clients can *pull* information from sources, or the sources can *push* data to them automatically [Acharya et al., 1995]. The latter is applicable when a group of clients share the same sources and they can benefit from accepting responses addressed to other peers.

To provide consistent and reliable computing support, the data management layer must support transaction and replica control. A transaction consists of a sequence of database operations executed as an atomic action [M. Tamer Ozsu and Patrick Valduriez, 1999]. This definition encompasses the four important properties of a transaction: atomicity, consistency, isolation, and durability (i.e., ACID properties). Another important property of a transaction is that it always terminates, by either committing the changes or by aborting all updates. The principal concurrency control technique used in traditional transaction management relies on locking [M. Tamer Ozsu and Patrick Valduriez, 1999; Kapali P. Eswaran and Jim Gray and Raymond A. Lorie and Irving L. Traiger, 1976]. In this approach, all devices enter a state in which they wait for messages from one another. Since mobile devices may become involuntarily disconnected, this technique raises serious problems, such as termination blocking and reduction in the availability of data. Current generation solutions to the mobile transaction management problem often relax the ACID [Walborn and Chrysanthis, 1997] properties or propose completely different transaction processing techniques [Margaret Dunham and Abdelsalam Helal and Santosh Balakrishnan, 1997].

Having relaxed the ACID properties, one can no longer guarantee that all replicas are synchronized. Consequently, the data management layer must address this issue. Traditional replica control protocols, based on voting or lock principles [Carla Schlatter Ellis and Richard A. Floyd, 1983], assume that all replica holders are always reachable. This is often invalid in mobile environments and may limit the ability to synchronize the replica located on mobile devices. Approaches addressing this issue include data division into volume groups and the use of versions for pessimistic [Demers et al., 1994] or optimistic updates [Satyanarayanan et al., 1990; Guy et al., 1998]. Pessimistic approaches require epidemic or voting protocols that first modify the primary copy before other replicas can be updated and their holders can operate on them. On the other hand, optimistic replication allows devices to operate on their replicas immediately, which may result in a conflict that will require a reconciliation mechanism [JoAnne Holliday and Divyakant Agrawal and Amr El Abbadi, 2000]. Alternatively, the conflict must be avoided by calculating a voting quorum [Keleher and Cetintemel, 1999] for distributed data objects. Each replica can obtain a quorum by gathering weighted votes from other replicas in the system and by providing its vote to others. Once a replica obtains a voting quorum, it is assured that a majority of the replicas agree with the changes. Consequently, the replica can commit its proposed updates.

1.4.5 Service Management Layer

Service management forms another important component in the development of a mobile application. It consists of service discovery monitoring, service invocation, execution management, and service fault management. The service management layer performs different functions depending on the type of mobile application architecture. In the client-server architecture, most of the management (e.g., service execution state maintenance, computation distribution) is done by the server side of the application. Clients mostly manage the appropriate service invocation, notifications, alerts and monitoring of local resources needed to execute a query. In the client-proxy-server ar-

chitecture, most of the management (session maintenance, leasing and registration) is done at the proxy or the lookup server. Disconnections are usually managed by tracking the state of execution of a service, mostly at the server side, and retransmitting data once connection is established. One very important aspect of this layer is to manage the integration and execution of multiple services that might be required to satisfy a request from a client, referred to as service composition. Such requests usually require interaction of multiple services to provide a reply. Most of the existing service management platforms [Mao et al., 2001; Mennie and Pagurek, 2000] for composite queries are centralized and oriented toward services in the fixed wired infrastructure. Distributed broker-based architectures for service discovery, management and composition in wireless ad-hoc environments are current research topics [Chakraborty et al., 2002b]. Fault tolerance and scalability is another important component, especially in environments with many short-lived services. The management platform should degrade gracefully as more services become unavailable. Solutions for managing services have been incorporated into service discovery protocols designed for wired networks, but not mobile environments.

1.4.5.1 Service Transaction Management

This sub-layer deals with the management of transactions associated with *m-services*, i.e., services applicable to mobile computing environments. We discuss service transaction management as applied to client-server, client-proxy-server, and peer-to-peer architectures. Service transaction management in mobile computing environments is based on the same principles used by e-commerce transaction managers in the Internet. These principles are usually part of a transaction protocol such as the Contract Net Protocol [FIPA, 2001]. A Contract Net Protocol involves two entities, the buyer (aka manager) and the seller (aka contractor), who are interested in conducting a transaction. The two entities execute actions as specified in the protocol at each step of the transaction. Examples of these actions include *Call for Proposal (CFP)*, *Refuse*, *Propose*, *Reject-Proposal*, *Accept-Proposal*, *Failure*, and *Inform-Done*. In the Internet computing environment, the two entities execute all actions explicitly. In a mobile computing environment, complete execution of the protocol may be infeasible due to memory and computational constraints. For example, the *Refuse* action, performed by a seller who refuses the *CFP*, is implicit if the seller does not respond to the *CFP*. Thus, service transaction managers on mobile computing devices use simplified versions of transaction protocols designed for the Internet [Avancha et al., 2003].

In mobile computing environments using the client-server or client-proxy-server architecture, the service transaction manager would choose to use the services available on the Internet to successfully complete the transaction. For example, if a person is buying an airline ticket at the airport using her PDA, she could invoke the airline software's payment service and specify her bank account as the source of payment. On the other hand, in a peer-to-peer environment, there is no guarantee of a robust, online payment mechanism. In such situations, the transaction manager may choose other options, such as *micropayments*. For example, if a person were buying a music video clip from another person for $1.00, he may pay for it using digital cash. Both industry and academia have engaged in core research in the area of *micropayments* in past few years [Benjamin Cox and Doug Tygar and Marvin Sirbu, 1995; Soon-Yong Choi and Dale O. Stahl and Andrew B. Whinston, 1997].

Three of the most important e-service transactional features that must be applied to m-services are: *Identification*, *Authentication*, and *Confidentiality*. Every entity in a mobile environment must be able to uniquely and clearly identify itself to other entities with whom it wishes to transact. Unlike devices on the Internet, a mobile device may not be able to use its IP address as a unique identifier. Every mobile device must be able to authenticate transaction messages it receives from others. In a mobile environment when air is the primary medium of communication, anybody can eavesdrop and mount man-in-the-middle attacks against others in radio range. Confidentiality in a mobile environment is achieved through encryption mechanisms. Messages containing payment

and goods information must be encrypted to prevent theft of the data. However, mobile devices are constrained by computational and memory capacities to perform expensive computations involved in traditional encryption mechanisms. Technologies such as Smartcards [Hansmann et al., 2000] can help offload the computational burden from the mobile device at the cost of higher energy consumption.

1.4.6 Security Plane

Security has greater significance in a mobile environment than in a wired environment. The two main reasons for this are the lack of any notion of security on the transmission medium, and the real possibility of theft of a user's mobile device.

Despite the increased need for security in mobile environments, the inherent constraints on mobile devices have prevented large scale research and development of secure protocols. Lightweight versions of Internet security protocols are likely to fail because they ignore or minimize certain crucial aspects of the latter, in order to save computation and/or memory. The travails of the Wired Equivalent Privacy (WEP) protocol designed for the IEEE 802.11b are well-known [Jesse R. Walker, 2000]. The IEEE 802.11b working group has now released WEP2 for the entire class of 802.1x protocols. Bluetooth also provides a link layer security protocol that consists of a *pairing* procedure, which accepts a user-supplied passkey to generate an initialization key. The initialization key is used to calculate a link key, which is finally used in a challenge-response sequence, after being exchanged. The current Bluetooth security protocol uses procedures that have low computation complexity, so they are susceptible to attacks. To secure data at the routing layer in client-server and client-proxy-server architectures, IPSec [Kent and Atkinson, 1998] is used in conjunction with Mobile IP. Research in securing routing protocols for networks using peer-to-peer architectures has resulted in interesting protocols such as Ariadne [Yih-Chun Hu and Adrian Perrig and David B. Johnson, 2002] and Security-Aware ad hoc Routing [Seung Yi and Prasad Naldurg and Robin Kravets, 2001]. The Wireless Transport Layer Security protocol is the only known protocol for securing transport layer data in mobile networks. This protocol is part of the WAP stack. WTLS is a close relative of the Secure Sockets Layer protocol that is *de jure* in securing data in the Internet. Transaction and application layer security implementations are also based on SSL.

1.4.7 System Management Plane

The system management plane provides interfaces so that any layer of the stack in Figure 1.2 can access system level information. System level information includes data such as current memory level, battery power, and the various device characteristics. For example, the routing layer might need to determine whether the current link layer in use is IEEE 802.11b or Bluetooth to decide packet sizes. Transaction managers will use memory information to decide whether to respond to incoming transaction requests or to prevent the user from sending out any more transaction requests. The application logic will acquire device characteristics from the system management plane to inform the other end (server, proxy, or peer) of the device's screen resolution, size, and other related information. The service discovery layer might use system level information to decide whether to use semantic matching or simple matching in discovering services.

1.5 Conclusions

Mobile devices are becoming popular in each aspect of our everyday life. Users expect to use them for multiple purposes, including calendaring, scheduling, checking e-mail and for browsing the web. Current generation mobile devices, such as iPAQs, are powerful enough to support

more versatile applications that may already exist on the Internet. However, applications developed for the wired Internet cannot be directly ported onto mobile devices. This is because some of the common assumptions made in building Internet applications, such as presence of high bandwidth disconnection-free network connections, resource-rich tethered machines and computation platforms, are not valid in mobile environments. Mobile applications must take these issues into consideration. In this chapter, we have discussed the modifications to each layer of the OSI stack that are required to enable mobile devices to communicate with wired networks and other mobile devices. We have also discussed three popular application architectures, i.e., client-server, client-proxy-server and peer-to-peer, that form an integral part of any mobile application. Finally, we have presented a general framework that mobile applications should use in order to be functionally complete, flexible and robust in mobile environments. The framework consists of an abstracted network layer, discovery layer, location management, data management, service management, transaction management and application specific logic. Depending on the architecture requirements, each application may use only a subset of the described layers. Moreover, depending on the type of architecture, different solutions apply for the different layers. In conclusion, we have presented the reader with a sketch of the layered architecture and technologies that make up the state-of-the-art of mobile computing and mobile applications. Most of these have seen significant academic research, and more recently, commercial deployment. Many other technologies are maturing as well, and will move from academic and research labs into products. We feel that the increasing use of wireless local and personal area networks (WLANs, WPANs), higher bandwidth wireless telephony, and a continued performance/price improvement in handheld and wearable devices will lead to a significant increase in the deployment of mobile computing applications in the near future, even though not all of the underlying problems would have completely wrapped up solutions in the short term.

Acknowledgments

This work was supported in part by NSF awards IIS 9875433, IIS 0209001 and CCR 0070802, the DARPA DAML program, IBM and Fujitsu Labs of America, Inc.

References

Swarup Acharya, Rafael Alonso, Michael Franklin, and Stanley Zdonik. Broadcast disks: Data management for asymmetric communication environments. In Michael J. Carey and Donovan A. Schneider, editors, *ACM SIGMOD International Conference on Management of Data*, pages 199 – 210, San Jose, California, June 1995. ACM, ACM Press.

Ken Arnold, Bryan Osullivan, Robert W. Scheifler, Jim Waldo, Ann Wollrath, Bryan O'Sullivan, and Robert Scheifler. *The Jini Specification (The Jini Technology)*. Addison-Wesley, Reading, MA, June 1999. ISBN 0-201-61634-3.

Sasikanth Avancha, Pravin D'Souza, Filip Perich, Anupam Joshi, and Yelena Yesha. P2P M-Commerce in Pervasive Environments. *ACM SIGecom Exchanges*, 3(4):1–9, January 2003.

Sasikanth Avancha, Anupam Joshi, and Tim Finin. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, 35(6):96–99, June 2002.

Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE INFOCOM*, volume 2, pages 775–784, Tel Aviv, Israel, March 2000.

Ajay V. Barke and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *15th International Conference on Distributed Computing Systems*, pages 136–143, Vancouver, British Columbia, Canada, June 1995. IEEE Computer Society Press.

Benjamin Cox and Doug Tygar and Marvin Sirbu. NetBill Security and Transaction Protocol. In *First USENIX Workshop of Electronic Commerce.*, pages 77–88, Newyork. USA, July 1995.

Harini Bharadvaj, Anupam Joshi, and Sansanee Auephanwiriyakyl. An Active Transcoding Proxy to Support Mobile Web Access. In *17th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 118–123, West Lafayette, IN, October 1998.

Charles Brooks, Murray S. Mazer, Scott Meeks, and Jim Miller. Application-Specific Proxy Servers as HTTP Stream Transducers. In *4th International World Wide Web Conference*, pages 539–548, Boston, Massachusetts, December 1995.

Kevin Brown and Suresh Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM Computer Communications Review*, 27(5):19–43, October 1997.

Carla Schlatter Ellis and Richard A. Floyd. The Roe File System. In *3rd Symposium on Reliability in Distributed Software and Database Systems*, pages 175 – 181, Clearwater Beach, Florida, USA, October 1983. IEEE.

Dipanjan Chakraborty, Anupam Joshi, Tim Finin, and Yesha Yesha. GSD: A novel group-based service discovery protocol for MANETS. In *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*, pages 301–306, Stockholm, Sweden, September 2002a.

Dipanjan Chakraborty, Filip Perich, Sasikanth Avancha, and Anupam Joshi. DReggie: A smart Service Discovery Technique for E-Commerce Applications. In *Workshop at 20th Symposium on Reliable Distributed Systems*, October 2001.

Dipanjan Chakraborty, Filip Perich, Anupam Joshi, Tim Finin, and Yelena Yesha. A Reactive Service Composition Architecture for Pervasive Computing Environments. In *7th Personal Wireless Communications Conference (PWC)*, pages 53–62, Singapore, October 2002b.

Harry Chen, Tim Finin, and Anupam Joshi. Semantic Web in a Pervasive Context-Aware Architecture. In *Artificial Intelligence in Mobile System at UBICOMP*, Seattle, Washington, October 2003.

Alan Demers, Karin Petersen, Mike Spreitzer, Douglas Terry, Marvin Theimer, and Brent Welch. The bayou architecture: Support for data sharing among mobile users. In *Proceedings IEEE Workshop on Mobile Computing Systems & Applications*, pages 2–7, Santa Cruz, California, 8-9 1994.

Anind K. Dey and Gregory D. Abowd, editors. *Towards a Better Understanding of Context and Context-Awareness*, The Hague, The Netherlands, April 2000. GVU, Georgia Institute of Technology.

Margaret Dunham, Abdelsalam Helal, and Santosh Balakrishnan. A Mobile Transaction Model that Captures Both the Data Movement and Behavior. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2(2):149–162, 1997.

FIPA. FIPA Contract Net Interaction Protocol Specification. World Wide Web, `http://www.fipa.org/specs/fipa00029/XC00029F.pdf`, 2001.

Tom Goff, James Moronski, Dhananjay S. Phatak, and Vipul Gupta. Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments. In *INFOCOM*, volume 3, pages 1537–1545, Tel Aviv, Israel, March 2000.

Richard Guy, Peter Reiher, David Ratner, Michial Gunter, Wilkie Ma, and Gerald Popek. Rumor: Mobile data access through optimistic peer-to-peer replication. In *Workshop on Mobile Data Access in conjunction with 17th International Conference on Conceptual Modeling (ER)*, pages 254–265, Singapore, November 1998. World Scientific.

Hans-Erich Kottkamp and Olaf Zukunft. Location-aware query processing in mobile database systems. In *ACM Symposium on Applied Computing*, pages 416–423, Atlanta, Georgia. USA., February 1998.

Uwe Hansmann, Martin S. Nicklous, Thomas Schck, and Frank Seliger. *Smart Card Application Development using Java*. Springer-Verlag, 1st edition, 2000.

Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global Positioning System: Theory and Practice*. Springer-Verlag, 4th edition, May 1997.

Jesse R. Walker. Unsafe at any key size; An analysis of the WEP encapsulation. IEEE Document 802.11-00/362, October 2000.

JoAnne Holliday and Divyakant Agrawal and Amr El Abbadi. Database Replication Using Epidemic Communication. In Arndt Bode and Thomas Ludwig II and Wolfgang Karl and Ronal Wism, editor, *6th Euro-Par Conference*, volume 1900, pages 427–434, Munich, Germany, September 2000. Springer.

Anupam Joshi, Ranjeewa Weerasinghe, Sean P. McDermott, Bun K. Tan, Gregory Bernhardt, and Sanjiva Weerawarana. Mowser: Mobile Platforms and Web Browsers. *Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS)*, 8(1), 1996.

Kapali P. Eswaran and Jim Gray and Raymond A. Lorie and Irving L. Traiger. The Notion of Consistency and Predicate Locks in a Database System. *Communications of the ACM*, 19(11): 624–633, December 1976.

Peter J. Keleher and Ugur Cetintemel. Consistency Management in Deno. *ACM Mobile Networks and Applications*, 5:299–309, 1999.

Stephen Kent and Randall Atkinson. IP Encapsulating Security Payload. World Wide Web, `http://www.ietf.org/rfc/rfc2406.txt`, November 1998.

Graham Klyne, Franklin Raynolds, and Chris Woodrow. Composite Capabilities/Preference Profiles (CC/PP): Structure and Vocabularies. World Wide Web, `http://www.w3.org/TR/CCPP-struct-vocab/`, March 2001.

Laura Bright and Louiqa Raschid. Using Latency-Recency Profiles for Data Delivery on the Web. In *International Conference on Very Large Data Bases (VLDB)*, pages 550–561, Kowloon Shangri-La Hotel, Hong Kong, China, August 2002. Morgan Kaufmann.

M. Tamer Ozsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Inc., New Jersey, 2nd edition, 1999.

Zhuoqing Morley Mao, Eric A. Brewer, and Randy H. Katz. Fault-tolerant, Scalable, Wide-Area Internet Service Composition. Technical report, CS Division, EECS Department, University of California, Berkeley, January 2001.

Margaret Dunham and Abdelsalam Helal and Santosh Balakrishnan. A Mobile Transaction Model That Captures Both the Data and Movement Behavior. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2(2):149–162, October 1997.

Margaret H. Dunham and Abdelsalam (Sumi) Helal. Mobile Computing and Databases: Anything New? In *ACM SIGMOD Record*, pages 5–9. ACM Press, December 1995.

David Mennie and Bernard Pagurek. An Architecture to Support Dynamic Composition of Service Components. In *5th International Worshop on Component-Oriented Programming*, June 2000.

Steve Muench and Mark Scardina. XSLT Requirements. World Wide Web, `http://www.w3.org/TR/xslt20req`, Februrary 2001.

Olga Ratsimor and Vladimir Korolev and Anupam Joshi and Timothy Finin. Agents2Go: An Infrastructure for Location-Dependent Service Discovery in the Mobile Electronic Commerce Environment. In *ACM Mobile Commerce Workshop in Conjunction with MobiCom*, pages 31–37, Rome, Italy, July 2001.

Filip Perich, Sasikanth Avancha, Dipanjan Chakraborty, Anupam Joshi, and Yelena Yesha. Profile Driven Data Management for Pervasive Environments. In *3rd International Conference on Database and Expert Systems Applications (DEXA)*, pages 361–370, Aix en Provence, France, September 2002.

Chalers E. Perkins. *Mobile IP Design Principles and Practices*. Addison-Wesley Wireless Communication Series, Reading, MA, 1997.

C. Brian Pinkerton, Edward D. Lazowska, David Notkin, and John Zahorjan. A Heterogeneous Distributed File System. In *10th International Conference on Distributed Computing Systems*, pages 424–431, May 1990.

Chaitanya Pullela, Liang Xu, Dipanjan Chakraborty, and Anupam Joshi. A Component based Architecture for Mobile Information Access. In *Workshop in conjunction with International Conference on Parallel Processing*, pages 65–72, August 2000.

John Rekesh. UPnP, Jini and Salutation - A look at some popular coordination frameworks for future network devices. Technical report, California Software Labs, 1999. URL `http://www.cswl.com/whitepaper/tech/upnp.html`.

Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and Implementation of the Sun Network Filesystem. In *Summer USENIX Conference*, pages 119–130, Portland, OR, 1985.

Sasikanth Avancha and Vladimir Korolev and Anupam Joshi and Timothy Finin and Y. Yesha. On Experiments with a Transport Protocol for Pervasive Computing Environments. *Computer Networks*, 40(4):515–535, November 2002.

Mahadev Satyanarayanan, James J. Kistler, Puneet Kumar, Maria E. Okasaki, Ellen H. Siegel, and David C. Steere. Coda: A Highly Available File System for a Distributed Workstation Environment. *IEEE Transactions on Computers*, 39(4):447–459, 1990.

Roger Sessions. *COM and DCOM: Microsoft's Vision for Distributed Objects*. John Wiley & Sons, New York, NY, October 1997.

Seung Yi and Prasad Naldurg and Robin Kravets. Security-aware ad hoc Routing for Wireless Networks. In *2nd ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 299–302, Long Beach. California. USA., October 2001.

Soon-Yong Choi and Dale O. Stahl and Andrew B. Whinston. Cyberpayments and the Future of Electronic Commerce. In *International Conference on Electronic Commerce, Cyberpayments Area*, 1997.

John Veizades, Erik Guttman, Charles E. Perkins, and Scott Kaplan. RFC 2165: Service location protocol, June 1997.

Gary D. Walborn and Panos K. Chrysanthis. PRO-MOTION : Management of Mobile Transactions. In *ACM Annual Symposium on Applied Computing*, pages 101–108, San Jose, California. USA., February 1997.

Yih-Chun Hu and Adrian Perrig and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *8th ACM International Conference on Mobile Computing and Networking*, pages 12–23, Atlanta, Georgia. USA., September 2002. ACM Press.

Bruce Zenel. *A Proxy Based Filtering Mechanism for The Mobile Environment*. PhD thesis, Department of Computer Science, Columbia University, New York, NY, December 1995.