

# Contents

<b>1</b>	<b>Data Management for Mobile Ad-Hoc Networks</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Origins of Mobile Peer-to-Peer Computing Model . . . . .	3
1.3	Data Management Challenges in Mobile Ad-Hoc Networks . . . . .	3
1.3.1	Communications Layer . . . . .	4
1.3.2	Discovery Layer . . . . .	6
1.3.3	Location Management Layer . . . . .	8
1.3.4	Data Management Layer . . . . .	9
1.3.5	Transaction Management Layer . . . . .	18
1.3.6	Security and Privacy Plane . . . . .	19
1.3.7	System Management Plane . . . . .	19
1.4	Peer-to-Peer Data Management Model for Mobile Ad-Hoc Networks . . . . .	20
1.4.1	Data Representation Model . . . . .	23
1.4.2	MoGATU Architecture Model . . . . .	24
1.4.3	Application Layer . . . . .	24
1.4.4	Data Management Layer . . . . .	25
1.4.5	Communications Layer . . . . .	29
1.5	Future Work . . . . .	30
1.6	Chapter Summary . . . . .	31

*CONTENTS*

*CONTENTS*

# 1

---

## Data Management for Mobile Ad-Hoc Networks

**Filip Perich**, *Cougaar Software, Inc.*

**Anupam Joshi**, *University of Maryland, Baltimore County*

**Rada Chirkova**, *North Carolina State University*

### CONTENTS

1.1	Introduction . . . . .	1
1.2	Origins of Mobile Peer-to-Peer Computing Model . . . . .	3
1.3	Data Management Challenges in Mobile Ad-Hoc Networks . . . . .	3
1.3.1	Communications Layer . . . . .	4
1.3.2	Discovery Layer . . . . .	6
1.3.3	Location Management Layer . . . . .	8
1.3.4	Data Management Layer . . . . .	9
1.3.5	Transaction Management Layer . . . . .	18
1.3.6	Security and Privacy Plane . . . . .	19
1.3.7	System Management Plane . . . . .	19
1.4	Peer-to-Peer Data Management Model for Mobile Ad-Hoc Networks . .	20
1.4.1	Data Representation Model . . . . .	23
1.4.2	MoGATU Architecture Model . . . . .	24
1.4.3	Application Layer . . . . .	24
1.4.4	Data Management Layer . . . . .	25
1.4.5	Communications Layer . . . . .	29
1.5	Future Work . . . . .	30
1.6	Chapter Summary . . . . .	31

---

### 1.1 Introduction

The overall goal of data management and processing in mobile ad-hoc networks is to allow individual devices to compute *what* information each device needs, *when* the device needs it, and

*how* it can obtain the information. This chapter identifies the fundamental challenges and outlines ongoing and needed future work in order to achieve this goal.

Until recently, the research on mobile data management was dominated by the client-proxy-server model requiring an infrastructure support. In this model, mobile devices connect to the Internet and serve as client end-points. They initiate actions and receive information from servers, which reside on the network and provide the infrastructure support to the clients. This earlier research focuses primarily on the development of protocols and techniques that deal with disconnection management, low bandwidth and device resource constraints. This allows applications built for the wired world, *e.g.*, World Wide Web and databases, to run in wireless domains using proxy based approaches [Bharadvaj et al., 1998; Joshi, 2000]. In systems based on the cellular network infrastructure or wireless local area network infrastructure, the traditional client-proxy-server interaction is perhaps an appropriate model where the *client* database can be extremely lightweight [Bobineau et al., 2000], has a (partial) replica of the main database on the wired side [Imielinski et al., 1997; Tait et al., 1995] or where selected data is continuously broadcast into the environment and cached by the clients [Acharya et al., 1995; Goodman et al., 1997].

With the widespread use of short-range ad-hoc networking technologies, such as Bluetooth [Bluetooth SIG], an alternative data management model becomes necessary. These networking technologies allow spontaneous connectivity among mobile devices, including hand helds, wearables, computers in vehicles, computers embedded in the physical infrastructure, and (nano)sensors. Mobile devices can suddenly become both sources and consumers of information. There is no longer a clean distinction between clients and servers, instead devices are now peers. To further complicate the matter, there also is no longer a guarantee of infrastructure support. Consequently, for obtaining data, devices cannot simply depend on a help of some fixed, centralized server [Perich et al., 2003]. Instead, the devices must be able to cooperate with others in their vicinity in order to pursue individual and collective tasks. This will lead devices to become more autonomous, dynamic and adaptive with respect to their environments.

This chapter describes the origins of this novel mobile peer-to-peer computing model and relates it to the traditional mobile models.

More importantly, this chapter introduces problems that arise in traditional mobile data management systems as well as additional problems specifically related to mobile ad-hoc networks. The three fundamental sources of these problems represent the underlying wireless ad-hoc networking technologies, the traditional issues relating to data management in any mobile computing paradigm, and the problems related to context awareness.

This chapter also surveys proposed solutions to each problem category. Despite the fact that wireless ad-hoc networking technologies and peer-to-peer based data management paradigms attempt to solve similar problems, the chapter will illustrate that there is a very limited effort on cross-layer interaction, which is essential for mobile ad-hoc networks. This gap between the research on networking, data management, and context-awareness in pervasive computing environments is the fundamental problem of allowing a device to compute *what* information the device needs, *when* the device needs it, and *how* it can obtain the information.

To overcome this problem, this chapter then presents the MoGATU model [Perich et al., 2002a,b, 2003, 2004a,b,c] – a novel peer-to-peer data management model for mobile ad-hoc networks. The key focus of MoGATU is to narrow the gap to its minimum by enabling mobile devices to proactively learn their current context and adjust their computing functionality according to their users' needs and preferences. MoGATU abstracts all devices using Communication Interfaces, Information Managers, Information Consumers, and Information Providers. The Information Manager is the fundamental component of the model. It is responsible for majority of the data management and communication functionality. It is composed of multiple components, which are responsible for: (i) data and service discovery, (ii) query processing, (iii) join query processing, (iv) caching, (v) transactions, (vi) reputation, and (vii) for data-based routing among peer devices.

---

## 1.2 Origins of Mobile Peer-to-Peer Computing Model

Mobile computing applications can be classified into three categories – client-server, client-proxy-server and peer-to-peer – depending on the interaction model.

In the client-server model, a large number of mobile devices connect to a small number of servers residing on the wired network, organized in a cluster. This model is a direct evolution of the distributed object-oriented systems like CORBA and DCOM [Sessions, 1997]. Here, mobile devices terminal-like client end-points, initiating actions and receiving information from servers on the network. The servers then represent 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 servers. Servers are also responsible for handling lower level networking details, such as disconnection and retransmission.

The advantages of this model are simplicity of the client design and straightforward cooperation among cluster servers. The main drawback, however, is the prohibitively large overhead on servers to handle each mobile client separately, in terms of transcoding and connection handling, which severely decreases system scalability.

In the client-proxy-server model, 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 a server and a client is split into two physical connections – server-to-proxy and proxy-to-client. This model increases overall system scalability since servers interact only with a fixed number of proxies, which in turn are responsible for handling transcoding and wireless connections to the clients.

There have been substantial research and industry efforts [Bharadvaj et al., 1998; Brooks et al., 1995; Joshi et al., 1996; Zenel, 1995] 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. Unlike mobile devices, servers and proxies are powerful machines that can handle data formats of any type and image formats of high resolution. Therefore, data on the wired network must be transcoded to suit different mobile devices. It is important for a server or proxy to recognize the characteristics of a client device. Standard techniques for 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.

The widespread use of short-range ad-hoc networking technologies created an additional model based on peer interaction. In this peer-to-peer model, all devices, mobile and static, are treated as peers. Suddenly, mobile devices act as both servers and clients. Ad-hoc networking 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 is, however, an important issue in this model. The set of services available to a client is dynamically changing with respect to location and time. Consequently, this requires mobile devices to implement protocols for data and service discovery [Chakraborty et al., 2002a; Rekesh, 1999], collaboration and composition [Chakraborty et al., 2002b; Mao et al., 2001]. Although, the disadvantage of this model is the burden on the mobile devices in terms of energy consumption and network traffic handling, the key advantage is that each device may have access to more up-to-date location dependent information and interact with peers without the need of an infrastructure support. Particularly, the second advantage plays an important role in enabling and revolutionizing mobile ad-hoc networks.

---

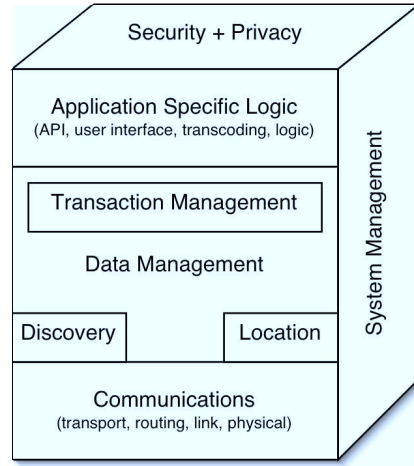


Figure 1.1: Layered data management framework for mobile ad-hoc networks.

### 1.3 Data Management Challenges in Mobile Ad-Hoc Networks

The aim of mobile ad-hoc networks is to extend the vision of mobile computing paradigm and enable people to accomplish their tasks *anywhere* and *anytime*, by using all computing resources, *i.e.*, data and services, currently available in their vicinity. This goal, however, raises many challenges from in multiple research areas.

There are three key sources of these issues and challenges: (i) One set of challenges emanates from the networking component, and includes problems relating to device discovery, message routing, and physical limitation of the underlying networking technology. (ii) A second set of challenges is due to a device's difficulty to be context aware by discovering and maintaining location of other devices and information in a network, since the topology is dynamic. (iii) The third key source of challenges is then the actual data management layer with issues such as transactional support or consistency among data objects.

Figure 1.1 illustrates the various layers that are essential for designing and developing a data management framework for mobile ad-hoc networks. Correspondingly, this section describes each layer individually by identifying key challenges and by offering a survey of existing approaches.

#### 1.3.1 Communications Layer

The communications layer represents wireless ad-hoc networking technologies that enable mobile devices to communicate with other devices in their vicinity. It is responsible for establishing and maintaining logical end-to-end connections between two devices, for data transmission, and for data reception. This layer encompasses the first four layers of the standard 7-layer Open Standards Interconnection (OSI) stack – physical, medium access control (MAC), network, and transport layers.

The primary task of the physical and MAC layers is to provide node discovery, and establish and maintain physical connections between two or more wireless entities. Each ad-hoc networking technology implements these functions differently. 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 in order to enable nodes to discover each other, when they are operating in an *ad-hoc* mode. When IEEE 802.11b nodes are operating in an *infrastructure* mode, a base station periodically broadcasts beacons, which other nodes use to discover the base station and to establish physical connections with it. The establishment of physical connections is a process in which the nodes ex-

change 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 connections) 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 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.

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.

### 1.3.1.1 Routing in Ad-Hoc Networks

Routing is perhaps the most important component of mobile ad-hoc networks. Routing allows devices to communicate with others outside their immediate wireless radio range. In the past few years, there has been significant effort on developing routing algorithms. Three representative algorithms are introduced in this section. Destination-Sequential Distance Vector Routing Algorithm (DSDV) [Perkins and Bhagwat, 1994] is a representative of a table-driven approach. Dynamic Source Routing Algorithm (DSR) [Johnson and Maltz, 1996] represents an alternative source-initiated approach, and Ad-Hoc On-Demand Distance Vector Routing Algorithm (AODV) [Perkins and Royer, 1999] is a hybrid of the two approaches.

Destination-Sequential Distance Vector Routing Algorithm (DSDV) [Perkins and Bhagwat, 1994] is a table-driven algorithm based on the Bellman-Ford routing mechanism [Cormen et al., 2001]. Every node in the network maintains a routing table containing a list of all possible destinations and the number of hops to reach them. Additionally, every entry in the table is assigned a sequence number as obtained from the destination node. Every node periodically transmits routing table updates to maintain consistency in the network topology. Duplicate routes with the same sequence number are rejected and only the shortest route is accepted. The algorithm therefore responds to topology changes by detecting them and by propagating the information to all nodes in the network. Finally, the algorithm attempts to conserve the available bandwidth by propagating only partial routing information whenever possible.

Dynamic Source Routing Algorithm (DSR) [Johnson and Maltz, 1996] is a source-initiated on demand routing protocol – an alternative to table-driven routing algorithms. DSR creates a route

only upon an explicit source initiated request. When a device requires a route to a destination, it initiates a route discovery process within the network. Upon discovering a proper route, a route maintenance procedure is executed to maintain the route until every path from the source is no longer available. DSR consists of a route learning and maintenance policy accompanied by a route discovery process. When a mobile device needs to send a message, it first verifies the destination by matching it with known routes. Alternatively, the device initiates a route discovery and waits for a *route reply* message that is generated by the destination device. The reply is routed back from destination throughout the same path as it was received on. Lastly, route maintenance is accomplished through the use of *route error* and *acknowledgment* messages.

Ad-Hoc On-Demand Distance Vector Routing Algorithm (AODV) [Perkins and Royer, 1999] is an on-demand table-driven algorithm. AODV is an improvement upon DSDV because every node is not required to maintain a complete list of routes for all other nodes in the network. Instead, each node in the network maintains route information for only those paths in which it is actively involved. Similar to DSR, this algorithm consists of two parts: *path discovery* and *path maintenance*. Each node maintains a sequence number and broadcast ID. During the path discovery, a source node broadcasts a route request with a unique ID for the desired destination. When an intermediate node knows a path to the destination it replies with that information by reversing the path, otherwise it broadcasts the request further. This algorithm therefore requires the use of symmetric links because path replies and other messages are sent back along the reverse path of the path discovery messages. To reflect topology changes, the algorithm considers two possibilities. When a source node moves, it re-initiates the *path discovery* procedure. When an intermediate or destination node moves, its upstream neighbor detects the change and propagates *link failure* message to the source node along the reverse path. The source node may then choose to again re-initiate the *path discovery* procedure for the given destination.

### 1.3.2 Discovery Layer

The discovery layer helps a mobile device with discovering data, services and computation resources. 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 then provides upper layers with the knowledge and context of available sources.

There has been a 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 either based on *lookup-registry* or *peer-to-peer* oriented.

Lookup-registry based discovery protocols register information about the source to some centralized or distributed registry. Devices query this registry in order to obtain knowledge about the source, including its location and invocation parameters. 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.

On the other hand, peer-to-peer discovery protocols query each node in the network to discover available services on that node. 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) [Avanacha 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.

The Service Location Protocol (SLP) [Guttman et al., 1999] is a language independent protocol for automatic resource discovery on IP networks utilizing an agent-oriented infrastructure. The basis of the SLP discovery mechanism lies on predefined service attributes, which can be applied to universally describe both software and hardware services. The architecture consists of three types of agents: User Agent, Service Agent and Discovery Agent. The User Agents is responsible for discovering available Directory Agents, and acquiring service handles on behalf of end-user applications that request services. The Service Agent is responsible for advertising the service handles to Directory Agents. Directory Agent is responsible for collecting service handles and maintaining the directory of advertised services. SLP uses multi-casting for service registration and discovery, and unicasting for service discovery responses from Directory/Service Agents.

The Ninja Secure Service Discovery System (SDS) [Czerwinski et al., 1999] is a research level service discovery engine developed at University of California, Berkeley. The architecture consists of clients, services and SDS servers. The SDS server architecture is a scalable, fault-tolerant, secure and highly available service discovery repository. SDS servers are hierarchically arranged for scalability and availability purposes across both local and wide area networks. Service descriptions and messages used to send query and answers between devices are encoded using eXtensible Markup Language (XML). Additionally, the SDS uses encryption to ensure interaction privacy and uses capability-based access control to limit the clients in discovering only permissible services. Ninja services and clients then use well-known global SDS multicast channels to communicate with the service discovery servers.

Universal Plug and Play (UPnP) [UPNP Forum] extends the original Microsoft Plug and Play peripheral model to support service discovery provided by network devices from numerous vendors. UPnP works and defines standards primarily at the lower-layer network protocol suites, so that the devices can natively, *i.e.*, language and platform independently, implement these standards. UPnP uses the Simple Service Discovery Protocol (SSDP) for discovery of services over IP networks, which can operate with or without a lookup service in the network. In addition, the SSDP operates on the top of the existing open standard protocols utilizing HTTP over both unicast (HTTPU) and multicast UDP (HTTPMU). When a new service wants to join the network, it transmits an announcement to indicate its presence. If a lookup service is present, it can record this advertisement to be subsequently used to satisfy clients' service discovery requests. Additionally, each service on the network may also observe these advertisements. When a client wants to discover a service, it can either contact the service directly through the URL that is stored within the service advertisement, or it can send out a multicast query message, which can be answered by either the directory service

or directly by the service.

Jini [Sun Microsystems] is a distributed service-oriented architecture developed by Sun Microsystems. A collection of Jini services forms a Jini federation. Jini services coordinate with each other within the federation. The overall goal of Jini is to turn the network into a easily administered tool on which human and computational clients can find services in a flexible and robust fashion. One of the key components of Jini is the Jini Lookup Service (JLS), which maintains the dynamic information about the available services in the Jini federation. Every service must discover one or more JLS before it can enter a federation. When a Jini service wants to join a Jini federation, it first discovers one or many JLS from the local or remote networks. The service then uploads its service proxy (*i.e.*, a set of Java classes) to the JLS. This proxy can be used by the service clients to contact the original service and invoke methods on the service. Service clients interact only with the Java-based service proxies. This allows various types of services, both hardware and software services, to be accessed in a uniform fashion. For instance, a service client can invoke print requests to a PostScript printing service even if it has no knowledge about the PostScript language.

DReggie [Chakraborty et al., 2001] extends the matching mechanisms in Jini and other service discovery systems by providing a semantic-based matching. The key idea in DReggie is to enable the service discovery systems to perform matching based on semantic information associated with the services as an alternative to strictly syntactic (*i.e.*, string matching) techniques. The semantic information of services consists of their extensive descriptions including, but not limited to, capabilities, functionality, portability, and system requirements. Semantic service matching introduces the possibilities of fuzziness and inexactness of the response to a service discovery request. In the DReggie system, a service discovery request contains the description of an “ideal” service - one whose capabilities match exactly with the requirements. Thus, matching now involves comparison of requirements specified with the capabilities of existing services; however, depending on the requirements, a match may occur even if one or more capabilities does not match exactly.

The Bluetooth Enhanced Service Discovery Protocol (ESDP) [Avancha et al., 2001], similar to DReggie, extends an existing discovery protocol, which in this case uses UUID-based matching, specified in Bluetooth architecture. The Bluetooth architecture is discussed in Section ???. ESDP presents a more sophisticated matching mechanism using semantic information to decide the success or failure of a query. The initial version employed the RDF/RDF-S [Brickley and Guha, 2000; Lassila and Swick, 1999] data model, while the current version utilizes the DARPA Agent Markup Language + Ontology Inference Layer [DARPA] to describe, register and discover services at peer devices.

### 1.3.3 Location Management Layer

The responsibility of the location management layer is to provide location information to a mobile device. Location information changes dynamically with mobility of the device and is one of the key components of context awareness. A location 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], which are 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, and 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. A more general formulation of such issues can be found in [Chen et al., 2003], which defines semantically rich policies and a Decision Logic based reasoner for specifying and reasoning about a users privacy preferences as related to context information.

### 1.3.4 Data Management Layer

The actual 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.

Distributed database systems and distributed file systems in mobile computing environments address the challenges introduced by sharing data that can reside both on devices in a fixed infrastructure and on mobile devices. The systems attempt to provide solutions for two challenges raised by the communication characteristics, mobility, and portability of the environments. The first question relates to the location of the database – on the mobile device or the wired network. The latter is typically assumed. So, a mobile device may require access to data that resides on the wired network, but may not be able to obtain due to network disconnection or low bandwidth. Often, the solution is to replicate or cache data on the mobile device to ensure access. This brings up the issue of updates. For a transaction to succeed, a mobile device must be able to commit its updates at the appropriate data manager residing on the wired network. Additionally, when data is modified at the primary side, all mobile devices should receive corresponding updates for their replicas. Mobile data management solutions thus attempt to extend the traditional distributed database systems by addressing challenges that arise due to the following conditions:

- Wireless networks have limited bandwidth and are prone to frequent failures.
- Channels in wireless networks may be asymmetric.

- Mobile devices have limited battery power.
- Mobile devices have limited resources.

Mobile ad-hoc networks only exacerbate the issues. This can be illustrated by classifying the mobile ad-hoc network model along four orthogonal axes that represent autonomy, distribution, heterogeneity and mobility of mobile databases [Dunham and Helal, 1995]. *Autonomy* refers to control distribution. It indicates the degree to which each mobile device can operate independently from the servers in the fixed infrastructure. It is a function of numerous factors defining the restrictions on execution of transactions as well as consistency requirements. *Dimension* classifies the data distribution model among all mobile and fixed devices in the system. At one extreme, all data can reside only on one device (usually the server), while at the other extreme all data can reside on all devices within one system (*i.e.*, full replication). *Heterogeneity* defines the hardware as well as software (primarily protocol) heterogeneity supported by a system. Lastly, *mobility* defines the degree of mobility that a particular system provides.

Mobile ad-hoc networks are highly autonomous since there is no centralized control of the individual client databases. They are heterogeneous as entities can only *speak* to each other in some neutral format. The mobile ad-hoc networks are clearly distributed as parts of data may reside on different devices and there is replication as entities cache data and their respective metadata. Mobility is of course given – in mobile ad-hoc networks, devices can change their locations and no fixed set of entities is *always* accessible to a given device. The last point is perhaps the most important. It is also the main reason why a direct use of solutions developed for mobile information access is inappropriate. In mobile distributed systems, disconnections of mobile devices from the network are viewed only as temporary events. Additionally, these systems often assume that all data *managers* are located at fixed positions in the wired network and that their locations are known by every client a priori [Bukhres et al., 1997; Kottkamp and Zukunft, 1998; Lauzac and Chrysanthis, 1998]. Finally, an additional limitation is the naming schema for defining data and for locating both data and devices in the traditional system. Here, each client must know the precise server location as well as its corresponding database schema in order to utilize the data properly.

Much like the arguments made in [Dunham and Helal, 1995], the status of data management in wireless networks versus wired networks can be compared to that of distributed data management versus centralized data management in the late 60s. The issues are often the same, but the solutions are different. Therefore, first the traditional challenges of any distributed data management are described, which are then followed by additional challenges specific to mobile ad-hoc networks. This overview is based on [Dunham and Helal, 1995; Franklin, 2001; Imielinski and Badrinath, 1994; Oezsu and Valduriez, 1999; Zaslavsky and Tari, 1998].

**Query Processing and Optimization** Query processing is highly affected by the addition of mobility to distributed data management systems in mobile environments. 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 advocated *location transparency*, where a query should return the same outcome irrespective of the client's location. These techniques thus considered only the aspects of data transfer and processing to optimize a given query. On the other hand, in the mobile computing environment, the query processing approaches promote *location awareness* [Kottkamp and Zukunft, 1998]. For example, a mobile device can ask for a location of the closest Greek restaurant, and the server should understand that the starting search point refers to the current position of the device.

**Caching** With the possibility of disconnection of mobile devices from the wired infrastructure, the mobile devices require data be cached on their locally available storage. Data caching allows mobile devices to operate even in disconnected mode. At the same time, this may require a weaker notion of

consistency as the mobile device may operate on stale data without the knowledge that the primary copy located in the wired infrastructure was altered. Hence, different consistency constraints as well as intelligent caching methods are required to allow a disconnected mode of operation.

**Replication** Another issue arises when one mobile device holds a complete replica of a database. The traditional replica control protocols are often based on implicit assumptions, which are no longer valid in the mobile environment. They assume that the communication among devices is symmetric. They also assume that all replicas are always reachable. This is not the case in the mobile environment. This may limit the ability to synchronize the replica located on the mobile device. It may also limit the ability of accepting data modification even in the wired infrastructure, as one of the replica owners may be unavailable to vote.

**Name Resolution** Name resolution also plays an important role in data management in mobile environments. As devices may move from one location to another or become disconnected, it is necessary to provide a global naming strategy to be able to locate a mobile station, which may hold the required data. This can be solved by broadcasting a request for the device such as a device discovery in Bluetooth networks. This however introduces reachability limitations as well as a high communication overhead. Alternatively, name resolution can be done by creating a “home” base station for each mobile device, which keeps track of the particular mobile device’s location, and can act as proxy to transmit messages to the mobile device if reachable over the network. This solution was studied extensively in [Perkins, 2002].

**Transaction Management** Lastly, the bandwidth limitations and possible long spanned disconnections of mobile devices require a new model for transactions as well as transaction processing techniques. This functionality is further described in the next Section 1.3.5.

Additionally, mobile ad-hoc networks impose the following issues that are primarily related to the randomness of every device’s neighborhood at any instance of time. The neighborhood, also referred to as vicinity, consists of all reachable devices that a particular (mobile) device can communicate with and all available data that is accessible at that time.

**Spatio-temporal variation of data and data source availability.** As devices move, their vicinity changes dynamically affecting data and data source availability. Additionally, current wireless networking technologies cannot support stable connections under high mobility.

**Lack of a global catalog and schema.** As the neighborhood changes dynamically, a mobile device has no prior knowledge of the current set of available data. There is no global catalog that it may contact and ask for a location of a given data item.

**No guarantee of reconnection.** When a device moves away from a current neighborhood it may affect any ongoing interaction among other devices of that neighborhood. As there is no guarantee that the mobile devices will ever again be able to communicate among themselves, this may cause an inconsistent global state.

**No guarantee of collaboration.** The issues of privacy and trust are very important for mobile ad-hoc networks where random devices interact in random transactions [Undercoffer et al., 2003]. A device may have reliable information but refuses to make it available to others. A device may be willing to share information; however that information is unreliable. Lastly, when a device makes information available to other devices questions regarding protection of future changes and sharing of that data arise.

One consequence of these challenges is that query answering is highly serendipitous. The answer obtained will depend on information sources accessible in the current vicinity. Consequently, each device in mobile ad-hoc networks must gather information pro-actively and much of the interaction among devices should happen in the background, without an explicit human intervention [Franklin, 2001; Perich et al., 2004a]. This requires that devices adapt themselves to the needs and preferences of their users and the current context.

#### 1.3.4.1 Approaches for Disconnected Operation

This section presents related work on data management challenges from the disconnected operation perspective. This work is primarily based on client/server model, where clients are mobile devices with intermittent connectivity operating under various processing and energy related constraints. The primary concept is to leverage the traditional client/server model from the wired infrastructure networks to operate also in wireless environments. The authors of the work described below usually relax one of the properties of the wired solutions to allow a seaming-less (yet limited) operation in the new environment. Commonly, a proxy point is added between the client and server or a weaker notion of transactions is introduced allowing mobile clients to operate in a disconnected mode.

Within a mobile database environment, cached data on mobile clients can take the form of materialized views. In order to efficiently maintain such materialized views while taking into consideration disconnected operations, [Lauzac and Chrysanthis, 1998] present a mechanism within the fixed network they refer to as *view holder* that maintains versions of views required by a particular mobile host. A view defines a function from a subset of base tables to a derived table, where (base) tables are the common data structures within a relational database system. A view is materialized by physically storing the derived table in forms of tuples. In distributed environments with client-server configuration, such as mobile databases, materialized views can be stored at the client side to support local query processing. Materialized views operate in a fashion similar to data caches. Available data can be quickly processed through the materialized views without the requirement of accessing a remote database server. Due to the communication costs and frequent disconnections of wireless networks information stored within the mobile computer becomes crucial to maintaining productivity. If the data needed to complete a task are present on the mobile computer, remote access may be eliminated and processing may continue even though disconnection has occurred. The authors argue that most of the transactions in a database environment are read-only, and thus the main focus of this paper was on optimizing read-only transactions on mobile computers. The authors do not consider write transactions, and only suggest that in the occurrence of a write transaction, the transaction should be performed directly with the data sources and not through the materialized views stored on the mobile client. Their proposed layered system architecture for read-only transactions thus consists of four layers: data servers, data warehouses, mobile hosts, and view holders. The data server layer is responsible for periodically constructing a maintenance transaction in order to update the data warehouse. A data warehouse, using a versioning maintenance algorithm, is created where the views are static and the number of consecutive versions of each view also remains static. Hence, the amount of space made available for versions of a particular attribute is known and fixed. A view holder is a mechanism for providing dynamic and customizable view maintenance so that the cache or view consistency achieved between the data stored on the mobile host and the data sources match the availability or cost of the network and the capabilities of the mobile host. A view holder will maintain a version of a view requested by a mobile host for as long as the mobile host needs it. So, the view holder can be seen as a buffer, holding versions of a specialized view for a particular mobile host. Space allocated for the updated attributes of a view must be done dynamically since it is not known beforehand how many versions will be maintained. The assumption is the views requested by an MH are very likely to be a small and specialized amount of the information from within the data servers and/or data warehouses. To avoid these huge storage requirements, versions of the requested data are dynamically maintained by the view holder. It is possible some of the data

sources including data warehouses may not support explicit versions of data. In such a case, the view holder will query the source in order to extract the data at a given moment. A timestamp for this implicit *version* could be the last time the tuple, attribute, or table was modified and found by querying the catalog of the data source.

[Kottkamp and Zukunft, 1998] present optimization techniques of query processing in mobile database systems of queries that include location information. Query processing in mobile database systems is a special challenge due to the resource limitations and constantly changing location of the mobile host. The authors concentrate on the optimization of queries that include location information, and suggest the use of a location management component. The location management component is responsible for updating the information about the actual location of the mobile host and resides on the wired network either at the location of home station for a particular mobile host or at some other centralized and fixed location on the wired network. All ad-hoc queries that use location information about the mobile host have to access the location component. Depending on the used localization strategy, the ad-hoc queries are associated with a different cost. Therefore, the authors develop and present a cost model for query optimization incorporating mobility specific factors like energy and connectivity. Additionally, the authors argue that a query processing in mobile database systems is significantly different from that in stationary systems and must be performed using different techniques. The query processing subsystem of a DBMS is organized into several phases: translation, optimization, and execution. The authors concentrate primarily on query optimization. In the phase of query optimization, potential query execution plans are generated and evaluated using a cost function. The cheapest plan is chosen for execution. In stationary systems, disk and main memory accesses are the foremost optimization criteria. In a mobile environment, additional constraints like the energy consumption of a query have to be taken into account. Mobile database systems must be able to choose an execution site for the different phases dependent on their current environment and should be able to revise that decision as flexible as possible. The authors then examine different localization strategies for mobile users. To validate their optimization strategies, the authors have developed a simulation model of mobile query processing and performed various experiments. They show that no single localization strategy performs acceptably under all conditions and identify the critical factors for adapting a query processing subsystem to the employed location management strategy.

[Bukhres et al., 1997] consider an infrastructure-based mobile network model consisting of mobile hosts and mobile support stations utilizing an IP-type communication and addressing. According to their description, a mobile host is an intelligent device, which can move freely while maintaining its connection to the network. The mobile support station is connected to the network via a wired medium and provides a wireless interface that allows the mobile host to interact with the static network. Each mobile support station is responsible for a geographical cellular region and is required to maintain the addresses of the mobile hosts, which are located within its region. Similar to [Kottkamp and Zukunft, 1998], the authors consider location dependent queries under disconnected operation mode. To overcome the issue of mobility, the authors introduce the concept of a mailbox for each mobile host. The mailbox is the recipient of all the query messages and results from the network and must be always accessible by its respective mobile host owner. Mailboxes thus provide a central repository for all of the mobile host's query responses and logs. This is similar to the concept of voice mailboxes in standard telecommunication cellular and wired networks. Additionally, the mailbox can be used during the process of a mobile transaction recovery, as all messages destined for a particular mobile host are always sent to it. Therefore, any DBMS is able to resolve and recover from any transactional conflicts by simply interacting with the mobile hosts' mailboxes and does not require the mobile hosts to be always accessible over the wireless network. Lastly, it helps reducing the load on the system as the messages may be sent only within the wired network between the query processor and the mailbox for the particular mobile host. The authors then consider issues related to the cellular region based infrastructure, namely: mobile support router hand-off and zone-crossing. Mobile support router hand-off occurs when a

mobile computer moves within a given cellular zone, while zone-crossing occurs when a mobile computer moves between two cellular zones. The authors address these problems using a mobile protocol adapted from the Columbia host protocol proposal [Ioannidis et al., 1991].

[Pitoura, 1996] presents a replication schema appropriate for environments where connectivity is partial, weak and variant such as in mobile computing. She considers a distributed database with data located both at mobile and stationary hosts to allow autonomous operation during disconnections. Transactions are distributed and can be initiated both at mobile and at stationary hosts. As an alternative for requiring a mutual consistency of all copies of data items, their proposed approach allows bounded inconsistencies. Pitoura groups together all data located at strongly connected hosts to form a cluster. While all data inside a cluster are consistent, various degrees of inconsistency are defined for replicas at different clusters. To maximize local processing and to reduce network access, her proposed mechanism enhances the interface offered by the database systems with operations using weaker consistency guarantees, which allow access to data that exhibit bounded inconsistency. Pitoura introduces two new types of operation, weak reads and weak writes. These operations allow users to operate on data with bounded inconsistency. The traditional read and write operations are referred to as strict read and strict write, respectively. Additionally, Pitoura also defines two strict and weak transactions. The strict transaction again represents the traditional notion of transaction in distributed database systems, while weak transaction is defined a transaction consisting of strict operations and at least one weak operation. The weak transaction, however, requires two types of commit: local and global. The local commit point is expressed by an explicit commit protocol, and updates made by locally committed weak transactions are visible only by weak transactions in the same cluster. These updates become permanent and visible by strict transactions only after reconciliation when local transactions become globally committed. Pitoura then continues by presenting an implementation, wherein schema distinguishes copies into quasi and core. Core copies are copies whose values are up-to-date and permanent, while quasi copies are copies whose values may be obsolete and are only conditionally committed. Lastly, Pitoura introduces protocols for enforcing the schema and evaluates the performance of the weak consistency schema for various networking conditions.

[Demers et al., 1994] present the system architecture of the Bayou System which is a platform of replicated, highly available, variable-consistency, mobile databases on which to build collaborative applications. The emphasis is on supporting application-specific conflict detection and resolution, and on providing application-controlled inconsistency. The system is intended to run in a mobile computing environment that includes portable machines with less than ideal network connectivity, and the goal of the system is to support data sharing among mobile users. The authors predominantly consider the issue of disconnected operations, which they call an experience of extended and sometimes involuntary disconnection from many or all of the other devices with which a particular mobile device wishes to share data. The system architecture is based on client-server model, where servers store data, and clients read and write data managed by servers. A server is any machine that holds a complete copy of one or more databases, which loosely denotes a collection of data items instead of the traditional database notion. Clients are able to access data residing on any server to which they can communicate, and any machine holding a copy of a database must also act as a server accessible by others. Therefore, in their architecture, servers do not have to reside on a wired network, and instead any mobile device can also operate as a server. For example, when several users become disconnected from the rest of the system, they can continue to actively collaborate among themselves if at least one user is able to utilize its mobile device as the group's server. To allow any two devices that are able to communicate with each other to propagate updates between themselves, the system employs a peer-to-peer reconciliation. Therefore, even machines that never directly communicate can exchange updates via intermediaries. Reconciliation can be structured as an incremental process so that even servers with very intermittent or asymmetrical connections can eventually bring their databases into a mutually consistent state. To resolve update conflicts, the system detects and resolves them in an application-specific manner. A write operation, thus,

includes not only the data being written or updated but also a dependency set. The dependency set is a collection of queries and their expected results. A conflict is detected if the queries are executed at a server and do not return the expected outcome. The write operation also specifies how to automatically resolve conflicts using a procedure called *mergeproc*. This procedure is invoked when a write conflict is detected. Hence, Bayou's write operation consists of a proposed update, a dependency set, and a *mergeproc*. The dependency set and *mergeproc* are both dictated by an application's semantics and may vary for each write operation issued by the application. The verification of the dependency check, the execution of the *mergeproc*, and the application of the update set are done atomically with respect to other database accesses on the server.

[Walborn and Chrysanthis, 1997] present a mobile transaction-processing system Pro-Motion. The underlying transaction-processing model of Pro-Motion is the concept of nested-split transactions. Nested-split transactions are an example of open nesting, which relaxes the top-level atomicity restriction of closed nested transactions where an open nested transaction allows its partial results to be observed outside the transaction. Consequently, one of the main issue for describing the local transaction processing on the mobile host (MH) is visibility and allowing new transactions to see uncommitted changes (weak data), which may result in undesired dependencies and cascading aborts. At the same time, when an update is made on a disconnected MH, subsequent transactions using the same data would be unable to proceed until a connection occurs and the mobile transactions could commit. Pro-Motion considers the entire mobile sub-system as one extremely large, long-lived transaction, which executes at the server with a sub-transaction executing at each MH. Each of these MH sub-transactions, in turn, is a root of another nested-split transaction. The results of local transactions on MH are automatically made visible for subsequent local access. In this way, local visibility and local commitment can reduce the blocking of transactions during disconnection and minimize the probability of cascading aborts. It is built on generalized client-server architecture with a mobile agent called compact agent, a stationary server front-end called compact manager, and an intermediate array of mobility managers to help manage the flow of updates and data between the other components of the system. Its fundamental building block is the compact, which functions as the basic unit of replication for caching, prefetching, and hoarding. A compact is defined as a satisfied request to cache data, with its obligations, restrictions and state information. It represents an agreement between the database server and the mobile host where the database server delegates control of some data to the MH to be used for local transaction processing. The database server need not to be aware of the operations executed by individual transactions on the MH, but, rather, sees periodic updates to a compact for each of the data items manipulated by the mobile transactions. Compacts are defined as objects encapsulating the cached data, methods for the access of the cached data, current state information, consistency rules, obligations and the interface methods. The management of compacts is performed by the compact manager on the database server and the compact agent on each mobile host cooperatively. Compacts are obtained from the database by requesting when a data demand is created by the MH. If data is available to satisfy the request, the database server creates a compact with the help of compact manager. The compact is then recorded to the compact store and transmitted to the MH to provide the data and methods to satisfy the needs of transactions executing on the MH. It is possible to transmit the missing or outdated components of a compact, which avoids the expensive transmission of already available compact methods on the MH. Once the compact is received by the MH, it is recorded in the compact registry, which is used by the compact agent to track the location and status of all local compacts. Each compact has a common interface, which is used by the compact agent to manage the compacts in the compact registry list and to perform updates submitted by transactions run by applications executing on the MH. Compact agent also performs disconnected processing when the mobile host is disconnected from the network and the compact manager is processing transactions locally. The compact manager maintains an event log, which is used for managing transaction processing, recovery, and resynchronization on the MH. Local commitment is permitted to make the results visible to other transaction on the MH, accepting the possibility of an eventual failure to commit at the

server. Transactions, which do not have a local option, will not commit locally until the updates have committed at the server. As more than one compact may be used in a single transaction, the commitment of a transaction is performed using a two-phase commit protocol where all participants reside on the MH. On the other hand, resynchronization occurs when the MH reconnects to the network and the compact agent is reconciling the updates committed during the disconnection with the fixed database.

[Dunham et al., 1997] define a mobile transaction model, called Kangaroo Transaction (KT), which addresses the movement behavior of transactions in a mobile computing environment. This transaction model incorporates the property that transactions in a mobile environment hop from one base station to another as the mobile device moves. The model captures this movement behavior and the data behavior reflecting the access to data located in databases throughout the static network. The authors assume an architecture where each base station hosts a Data Access Agent (DAA), which is used for accessing data in the database. When DAA receives a transaction request from a mobile user, the DAA forwards it to a specific base station or a fixed host that contains the required data. DAA acts as a Mobile Transaction Manager (MTM) and data access coordinator for the site. It is built on top of an existing Global Database System, which assumes that the local DBMS performs the required transaction processing functions including recovery and concurrency. DAA is, however, unaware of the mobile nature of some nodes or of the implementation details of each requested transaction. A hopping property is added to model the mobility of the transactions. Each subtransaction represents the unit of execution at one base station and is called a Joey Transaction (JT). The authors define a Pouch to be the sequence of global and local transactions, which are executed under a given KT. Each KT has a unique identification number consisting of the base station number and unique sequence number within the base station. When a mobile unit moves from one cell to another, the control of the KT changes to a new DAA at another base station. The DAA at the new base station creates a new JT as the result of the hand-off process. JTs have sequenced identification numbers consisting of both the KT identification number and an increasing number. The mobility of the transaction model is captured by the use of split transactions. The old JT is committed independently of the new JT. If a failure of any JT occurs, which in turn may result in undoing the entire KT, a compensation for any previously completed JTs must be assured. Therefore, a Kangaroo Transaction could be in a Split Mode or in a Compensating Mode. A split transaction divides an ongoing transaction into serialized subtransactions. Earlier created subtransaction may be committed and the remaining ones can continue in its execution. However, the decision on as to abort or commit a currently executing subtransaction is left up to the main DBMS. Previous JTs may not be compensated so that neither Splitting Mode nor Compensating Mode guarantees serializability of kangaroo transactions. Although Compensating Mode assures atomicity, isolation may be violated because locks are obtained and released at the local transaction level. With the Compensating Mode, Joey subtransactions are serializable. The MTM keeps a Transaction Status Table on the base station DAA to maintain the status of those transactions. It also keeps a local log into which the MTM writes the records needed for recovery purposes. Most records in the log are related to KT transaction status and some compensating information.

#### **1.3.4.2 Approaches for Data Dissemination and Replication**

This section presents related work on data dissemination and replication within wireless networks. The work on data dissemination assumes that servers have a relatively high bandwidth broadcast capacity while clients cannot transmit or can do so only over a lower bandwidth link. The data dissemination models are concerned with read-only transactions, where mobile clients usually issue a query to locate particular information or a service based on the current location of the device. Another model for data dissemination can be applied when a group of clients shares the same servers and they can, in general, also benefit from accepting responses addressed to other clients in their group.

[Acharya et al., 1995] present a broadcast-based mechanism for disseminating information in a wireless environment. To improve performance for non-uniformly accessed data, and to efficiently utilize the available bandwidth, the central idea is that servers are repeatedly broadcasting data to multiple clients at various frequencies. The authors superimpose multiple disks of different sizes and speeds to create an arbitrarily fine-grained memory hierarchy, and study client cache management policies to maximize performance. The authors argue that in a wireless mobile network, servers may have a relatively high bandwidth broadcast capacity while clients cannot transmit or can do so only over a lower bandwidth link. Such system have been proposed for many application domains, including hospital information systems, traffic information systems, and wireless classrooms. Traditional client-server information systems employ a pull-based algorithm, where clients initiate data transfers by sending requests to a server. The broadcast disks on the other hand exploit the advantage in bandwidth by broadcasting data to multiple clients at the same, and thus employ a push-based approach. In this approach, a server continuously and repeatedly broadcasts data to the clients, which effectively causes a creation of a disk from which clients can retrieve data as it goes by. The authors then model and study performance of various cache techniques at the client side and broadcast patterns at the server side within their architecture. The inherent limitations of this approach, however, restrict the clients to employ read-only transactions. In addition, it requires the client to wait for incoming data until it appears on the broadcast disk, even though the client may momentarily have a near-perfect wireless connectivity to a particular server.

[Tait et al., 1995] present an intelligent hoarding approach for caching files on the client side for mobile networks. The authors consider the case of a voluntary, client-initiated disconnection as opposed to involuntary disconnection that was under the scrutiny of many approaches described above. Therefore, the authors attempt to present a solution for intelligently caching important data at the client side, in their case files, once the client has informed the system about its planned disconnection. This is known as the hoarding problem, wherein hoarding tries to eliminate cache misses entirely during the period of client disconnection. The authors first describe other approaches consisting of doing nothing, utilizing explicitly user-provided information, logging user's past activity, and by utilizing some semantic information. Their approach is based on the concept of prefetching, and can be referred to as transparent analytical spying. The algorithm relies on the notion of working sets. It automatically detects these working sets for user's applications and data. It then provides generalized delimiters for periods of activity, which is used to separate time periods for which a different collection of files is required.

Infostations [Goodman et al., 1997] is a system concept proposed to support *many time, many where* wireless data services including voice mail. It allows mobile terminals to communicate to Infostations with variable data transmission rate to obtain the optimized throughput. The main idea is to use efficient caching techniques to hoard as much data as possible when connected to services within an island of high bandwidth coverage, and use the cached information when unable to contact the services directly. This idea is very similar to the previously described work by [Tait et al., 1995].

[Guy et al., 1998] discuss an optimistically replicated file system designed for use in mobile computers. The file system, called Rumor, uses a peer model that allows opportunistic update propagation among any sites replicating files. This work describes the design and implementation of the Rumor file system, and feasibility of using peer optimistic replication to support mobile computing. The authors discuss the various replication design alternatives and justify their choice of a peer-to-peer based optimistic replication. Replication systems can usefully be classified along several dimensions based on update type, device classification and propagation methods. Conservative update replication systems prevent all concurrent updates, causing mobile users who store replicas of data items to have their updates frequently rejected, particularly when connectivity is poor or non-existent. Optimistic replication on the other hand allows any device storing a replica to perform a local update, rather than requiring the machine to acquire locks or votes from other replicas. Optimistic replication minimizes the bandwidth and connectivity requirements for performing updates. At the same time, optimistic replication systems allow conflicting updates to occur. The devices can

be classified either into client and servers to as peers. In the client-server replication, all updates must be first propagated to a server device that further propagates them to all clients. Peer-to-peer systems, on the other hand, allow any replica to propagate updates to any other replica. Although, the client-server approach simplifies the system design and maintenance, the peer-to-peer system can propagate updates faster by making the use of any available connectivity. Lastly, the last dimension differentiates between an immediate propagation versus a periodic reconciliation. In the first case, an update must be propagated to all replicas as soon as it is (locally) committed, while in the latter case a batch method can be employed to conserve the constrained resources, such as bandwidth and battery. The authors, therefore, decided to design Rumor as an optimistic, peer-to-peer, reconciliation-based replicated file system. Rumor operates on file sets known as volumes. A volume is a continuous portion of the file system tree, larger than a directory but smaller than a file system. Reconciliation then operates at the volume granularity, which increases the possibility of conflicting updates and large memory and data requirement for storage and synchronization. At the same time, this approach does not introduce a high maintenance overhead. Additionally, the Rumor system employs a selective replication method and a per-file reconciliation mechanism to lower the unnecessary cost.

[Holliday et al., 2000] have investigated an epidemic update protocol that guarantees consistency and serializability in spite of a write-anywhere capability and conduct simulation experiments to evaluate this protocol. The authors argue that the traditional replica management approaches suffer from significant performance penalties. This is due to the requirement of a synchronous execution of each individual read and write operation before a transaction can commit. An alternative approach is a local execution of operations without synchronization with other sites. In their approach, changes are propagated throughout the network using an epidemic approach, where updates are piggy-backed on messages. This ensures that eventually all updates are propagated throughout the entire system. The authors advocate that the epidemic approach works well for single item updates or updates that commute; however, when used for multi-operation transactions, these techniques do not ensure serializability. To resolve these issues, the authors have developed a hybrid approach where a transaction executes locally, and uses epidemic communication to propagate all its updates to all replicas before actually committing. Transaction is only committed, once a site is ensured that updates have been incorporated at all copies throughout the system. They present experimental results supporting this approach as an alternative to eager update protocols for a distributed database environment where serializability is needed. The epidemic protocol relieves some of the limitations of the traditional approach by eliminating global deadlocks and by reducing delays caused by blocking. Additionally, the authors claim that the epidemic communication technique is more flexible than the reliable, synchronous communication required by the traditional approach, and justify this by presenting results of their performance evaluations. These results indicate that for moderate levels of replication, epidemic replication is an acceptable solution while significantly reducing the transmission cost.

### 1.3.5 Transaction Management Layer

This sub-layer deals with the managing transactions initiated by devices in mobile ad-hoc networks. For a transaction to succeed, a device must be able to commit its updates at the appropriate data manager that can be located in the wired network or on some of the device's peers in the current vicinity. Additionally, when data is modified at the primary side, all mobile devices should receive corresponding updates for their replicas.

[Oezsu and Valduriez, 1999] defines transaction as a basic unit of consistent and reliable computing, consisting of a sequence of database operations executed as an atomic action. This definition encompasses the four important properties of a transaction: atomicity, consistency, isolation, and durability (*i.e.*, ACID properties). *Atomicity* refers to the fact that a transaction is treated as a unit of operation. *Consistency* refers to a transaction being a correct transformation function from map-

ping one consistent state of a database onto another consistent state. *Isolation* requires that the data changes triggered by a transaction are hidden from others until the transaction commits. Lastly, *duration* of a transaction implies that an outcome of committed transaction is permanent and cannot be subsequently removed. Another important feature of a transaction is that it always terminates, by either committing the changes or by aborting all its updates.

The transaction problems in mobile environments arise due to the traditional concurrency control technique. The control technique often relies on locking, where a client wishing to modify data on the server database must first acquire a valid lock. For example, in the two-phase commit protocol (2PC) [Eswaran et al., 1976; Oezsu and Valduriez, 1999] each participant and coordinator enter a state, where they are waiting on a message from one another. The only other escape from the idle state is only triggered by an expired timer. Since mobile devices may become involuntarily disconnected, this technique raises serious problems. If lock is established on a mobile device, which becomes disconnected, the lock may be active for a long time, thus blocking the termination of a transaction. On the other hand, when a lock is established on a wired device by a mobile device, which since becomes disconnected, the data availability is reduced. These problems have spurred numerous solutions [Bukhres et al., 1997; Demers et al., 1994; Dunham et al., 1997; Lauzac and Chrysanthis, 1998; Pitoura, 1996]. These approaches are usually based on modeling a novel breed of mobile transactions by proposing different transaction processing techniques, such as [Dunham et al., 1997], and/or by relaxing the ACID properties as for example in [Walborn and Chrysanthis, 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 [Ellis and 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 [Kistler and Satyanarayanan, 1991; 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 [Holliday et al., 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.

### 1.3.6 Security and Privacy Plane

The issues related to security and privacy are very important in mobile ad-hoc networks. The three main reasons for this are the lack of any notion of security on the transmission medium, the lack of guaranteed integrity of data stored on mobile devices in the environment, and the real possibility of theft of a user's mobile device.

Despite the increased need for security and privacy 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 [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 be-

ing 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 [Yi et al., 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.

### 1.3.7 System Management Plane

The system management plane provides interfaces so that any layer of the stack in Figure 1.1 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.

## 1.4 Peer-to-Peer Data Management Model for Mobile Ad-Hoc Networks

This section presents the MoGATU model introduced by [Perich et al., 2002a,b, 2003, 2004a,b,c], which attempts to answer mobile data management challenges raised by traditional mobile computing environments and those challenges specific to mobile ad-hoc networks. The goal of the model is to allow mobile devices present in the environment to utilize efficiently their current resource-rich vicinity while pursuing their individual and collective tasks. The model makes three propositions:

**Postulate 1** *All devices in mobile ad-hoc networks are peers.*

The widespread adoption of short-range ad-hoc networking technologies allows mobile devices to interact with other devices in their current vicinity without the need of a back-end wired infrastructure. As a result, a mobile device can be both an information consumer, *i.e.*, a client in the traditional mobile model, or an information provider, *i.e.*, a server in the traditional mobile model. Consequently, there are no longer explicit clients and servers in this paradigm. Instead, they become peers that can both consume and provide different services and data.

**Postulate 2** *All devices in mobile ad-hoc networks are semi-autonomous, self-describing, highly interactive, and adaptive.*

The characteristics of mobile ad-hoc networks imply that a device's vicinity is highly volatile. Since all devices in the vicinity may be mobile, there is no guarantee about the duration of a connection among any pair of mobile devices. Consequently, mobile devices must be autonomous in order to operate correctly while their vicinity changes constantly. Additionally, as mobile devices move and as new data may arrive at any moment, there is no guarantee about the type of information

available at any given time and space. Mobile devices must be adaptive to this nature of the environment in that they must be able to change their functionality and needs based on what is currently available to them. Mobile devices must also be self-describing. They must be able to articulate their needs, which together with adaptivity will allow them to better utilize their vicinity. Finally, mobile devices must be highly interactive by offering data and services to their peers and by querying the information available on those peers.

**Postulate 3** *All devices in mobile ad-hoc networks require cross-layer interaction between their data management and communication layers.*

It is insufficient for mobile devices to employ mobile data management solutions that do not consider the underlying network characteristics. At the same time, it is simply not enough to attempt to solve the underlying networking problems, including device discovery and routing of traffic between devices, independently from the data management aspect. Such solutions would waste the limited bandwidth and other resources. They would also fail due to the inability to allow mobile devices to completely satisfy their individual and collective tasks. As argued in previous section, it is imperative that all mobile devices employ a model that considers both the networking and the data management aspects of the environments.

□

Figure 1.2 illustrates the corresponding representation of mobile devices from the MoGATU model's perspective. Applying definitions from [Yang and Garcia-Molina, 2001], the model can be classified as *chained* architecture with a *random* replication and local *incremental* policy.

The model is a chained architecture because each data source, or consumer, registers with a local Information Manager only. Remote Information Managers present on other devices in the system are unaffected. When a query is placed, first the local Information Manager attempts to answer it. If it is unable to answer the query, only then the Information Manager, forwards the query to some remote Manager to which it is currently connected, *i.e.*, to which it is *chained*.

The model employs a random replication policy because any mobile device can obtain and cache a specific data objects. There is no prior knowledge that can determine the location of all copies of a specific data object with respect to time and space.

The model also employs a local incremental update policy because there is no guarantee how long two devices may be able to communicate with each other. In order to overcome short session durations and network bandwidth limitations, Information Managers do not attempt to *load* all information their peers have available. Instead, an Information Manager only learns incrementally the capabilities of its peers as it queries them or through receiving remote advertisements.

Specifically, the MoGATU model addresses the data management challenges from Section 1.3.4 as follows:

- **Autonomy.** As described above, all devices are treated as independent entities acting autonomously from others.
- **Mobility.** The model does not place any restriction on the mobility patterns of devices.
- **Heterogeneity.** Mobile ad-hoc networks are highly heterogeneous in terms of devices, data resources, and networking technologies. The model addresses this issue by having each device implement an Information Manager. Each stored information and service that is able to generate additional information are further abstracted by Information Providers. Lastly, networking technologies are abstracted in terms of Communication Interfaces that allow devices to interact regardless of the underlying networks.

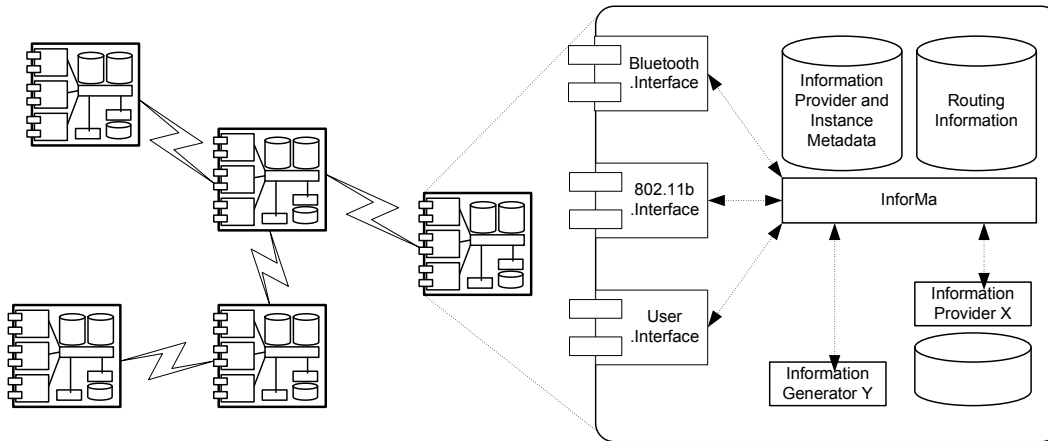


Figure 1.2: Device abstraction in the MoGATU model.

- Distribution.** Mobile devices may have multiple Information Providers, each holding a distributed subset of the global data repository. The model allows devices to advertise, solicit, exchange and modify such data with their peers.
- Lack of a global catalog and schema.** The model does not require a global catalog or schema. Instead, the model employs ontologies based on a semantically rich language – a set of common vocabularies. These ontologies enable devices to describe information provided by any Information Provider. These ontologies are also used to advertise, discover, and query such information among devices.
- No guarantee of reconnection.** To remedy the effects of reconnection, the model is a best-effort only and relies on pro-actively cached information. Additionally, a data-based routing algorithm is introduced, which allows *closer* devices to provide answers to queries placed by their peers whenever data is more important than its origin.
- Spatio-temporal variation of data and data source availability.** The model encourages every device to gather information pro-actively without a human interaction. A user profile is used for representing the necessary information in order to allow devices to act independently. The profile is also annotated in a semantically rich language and is used by devices for adapting their caching and querying behavior.

The model abstracts each peer device in terms of Information Providers, Information Consumers, and Information Managers. Additionally, the model defines abstract Communication Interfaces for supporting multiple networking technologies. This is illustrated in Figure 1.2.

Information Providers, described in Section 1.4.3.1, represent the available data sources. Every Information Provider holds a partial distributed set, a *fragment*, of heterogeneous data available in the whole mobile ad-hoc network. The data model, described in Section 1.4.1, is a set of ontologies with data instances expressed in a semantic language. Each Information Provider stores its data in the data's *base* form according to the ontology definition. Data involving one ontology is already expressed in its base form and stored in the format in which it was obtained. For data involving multiple ontologies, a Provider decomposes the data into their base forms and maintains a view linking to the base forms. Using this approach a view is represented as a list of pointers to the respective base fragments. It may be impossible to maintain global consistency among all Information Providers because the mobile ad-hoc network frequently remains partitioned. As a result, mobile nodes attempt to be vicinity-consistent only.

Information Consumers, described in Section 1.4.3.2, represent entities that query and update data available in the environment. Information Consumers can represent human users but also pro-active agents that actively pre-fetch context-sensitive information from other devices in the environment.

Lastly, an instance of an Information Manager, described in Section 1.4.4.1, must exist on every mobile device. Information Managers are responsible for network communication and for most of the data management functions. Each Information Manager is responsible for maintaining information about peers in its vicinity. This information includes the types of devices and information they provide. An Information Manager also maintains a data cache for storing information gathered from other mobile devices and for caching information generated by its local Providers.

Not illustrated in Figure 1.2 is the fact that each Information Manager also includes a user's profile reflecting some of user's beliefs, desires, and intentions (BDI). The BDI model has been explored in multi-agent interactions [Bratmann, 1987]. For profiles, it significantly extends [Cherniak et al., 2002], which explicitly enumerates data and its utility. In contrast, by using the BDI concept, profiles adapt to the environment by varying both data and their utility over time and present situations. Therefore, a profile enables pro-active device behavior because mobile devices can adapt their operation and functionality dynamically based on the current context and user's needs without waiting explicitly for a user's input. The Information Manager uses the profile for adapting its caching strategies and for initiating collaboration with peers in order to obtain desired information.

### 1.4.1 Data Representation Model

Every mobile device holds a subset of globally available *heterogeneous* data. Since the mobile ad-hoc networks are, by definition, open systems there are no restrictions or rules specifying the type and format of available data. In order to support heterogeneous devices but at the same time allow these devices to interact, it is important that these devices *speak* using a common language.

The efforts of the Semantic Web community attempt to address similar issues by defining a semantically rich language – the Web Ontology Language (OWL) [World Wide Web Consortium (W3C)]. This semantically rich language allows the specification of numerous types of data in terms of classes and their properties, and also defines relationship among the classes and the properties. It is advantageous to employ their proposed solution. In fact, as advocated in [Perich et al., 2004a], the use of ontologies in these environments is vital.

By adhering to an already existing language, the syntax and rules do not have to be re-invented by defining new formal language. Second, by utilizing a language used by the Semantic Web community, mobile devices will be able to use the resources available in their current vicinity as well as the vast resources available on the Internet. Therefore, the model assumes that information instances, profiles and other data objects are represented using the Web Ontology Language.

By using ontologies, the model, however, imposes a requirement that each device is able to parse OWL-annotated information. This is not to say that all devices will, or must, understand all ontologies. Rather, the model anticipates a scenario where each device has some knowledge over a set of ontologies. For new or unknown ontologies already present in the environment, a device can at least detect some metadata information by applying default OWL rules. This will allow a device to match queries with Information Providers' advertisements without any knowledge about the particular data. For example, this may allow a device that understands ontology *A* to use data annotated in an unknown ontology *B*, if *B* is a *subClassOf* *A*. For example, a device may not be able to deduce that Joy Luck is a Chinese restaurant, but it will at least know that Joy Luck is a restaurant.

Because each device is required to only parse OWL-annotated information, the introduction of ontologies in the system does not require much more processing resources than already available.

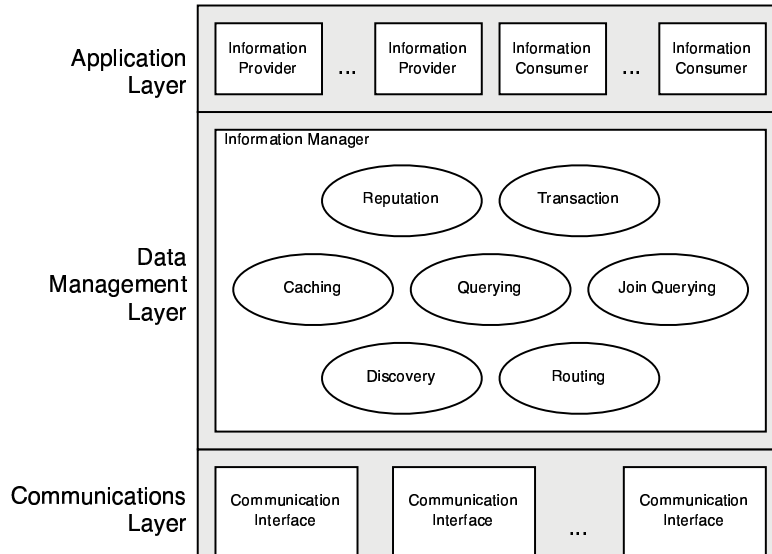


Figure 1.3: Layered architecture of the MoGATU architecture model.

## 1.4.2 MoGATU Architecture Model

The model can be represented using the layered approach illustrated in previous chapter in Figure 1.1. Communication Interfaces are responsible for the functionality of the communications layer. Information Providers and Information Consumers are specific instances defining the application logic at the application layer. Lastly, the Information Manager combines the tasks of data and transaction management layers and their discovery and location sub-layers. Figure 1.3 illustrates this possible re-ordering of the various components of the MoGATU model.

### 1.4.3 Application Layer

The application layer defines the specific logic employed by mobile devices. The logic specifies the interface for allowing users to operate over the devices. It also defines logic for devices to initiate actions and interact with other mobile devices. This logic can be abstracted in two types. One type represents Information Consumers, *i.e.*, applications that are searching for information, while the other type of logic represents Information Producers. Information Producers are those applications that can store or produce information requested by other applications, which can reside on the same or remote devices.

#### 1.4.3.1 Information Providers

Every device may hold one or more Information Providers. An Information Provider manages and provides an interface to a distributed subset of the global data repository. The data can be stored on the device or generated on-demand. The managed subset may be inconsistent with other *copies* located on other devices as there is no guarantee that the devices can interact, and the subset may even be empty. In MoGATU, any entity is an Information Provider whenever it is able to accept some query and generate a proper response. For example, an Information provider can represent a clock, a calendar or any other application on a mobile handheld device. Given the variety of Information Providers, the response of each Provider is based on the query, available stored data, and Provider specific mechanisms, including reference rules, for generating new data.

Each Information Provider describes its capabilities in terms of ontologies defined in a semantically rich language. The MoGATU model employs OWL. Moreover, the design is based on the OWL-S standard [The OWL Services Coalition, 2003], which attempts to comprehensively describe services for the World Wide Web. Using this approach, each Provider can describe itself by defining the service model it implements, the process model that provides the information, and the input, *i.e.*, query, restrictions, and requirements. Moreover, the language supports efficient discovery and matching approaches required for locating Information Providers, cached answers, or for answering queries [Chakraborty et al., 2001].

Upon start-up, each Information Provider registers itself with the local Information Manager by sending a registration message including the service model  $s$ , process models  $p$  and input restrictions  $I$ :

$$\boxed{\text{registration} = (s, p, I, t, a)} \quad (1.1)$$

Each Provider also defines a lifetime  $t$ , for specifying the time the Provider will be available, and whether it is willing to answer queries originating from remote devices, denoted as  $a$ . Each Provider, however, communicates with its local Information Manager only, which in turn routes messages between the Provider and other devices in the vicinity.

The Information Manager adds this Provider into its cache of local providers, and discards the entry once the *lifetime* expired and the Provider has not renewed its registration. Additionally, Information Manager may advertise the Provider to other devices in the vicinity if the Provider is willing to process queries for remote devices. The advertisement frequency is a tunable parameter for each Information Manager.

### 1.4.3.2 Information Consumers

Information Consumers represent entities that can query, consume, and update data. Information Consumers represent primarily human users asking their mobile devices for context-sensitive information but also represent autonomous software agents. Like Information Providers, Consumers register with local Information Managers by sending a registration message. The presence of Information Consumers is, however, not advertised to remote devices.

When a Consumer needs to obtain a specific data, the Consumer constructs an *explicit query*. It sends the query to its local Information Manager. The Manager routes the query to appropriate local Information Providers or other matching Providers located on remote peer devices for processing, and awaits a response.

Like data they operate over, queries are also defined using an OWL-based ontology. Specifically, the queries are written using OWL-S. A query is specified by a tuple consisting of a set of used ontologies ( $O$ ), selection list ( $\sigma$ ), filtering statement ( $\theta$ ), cardinality ( $\Sigma$ ), and temporal ( $\tau$ ) constrains:

$$\boxed{\text{query} = (O, \sigma, \theta, \Sigma, \tau)} \quad (1.2)$$

Each query defines the set of ontologies used for constructing the filtering clause and for final projection of the matching data instances. The set can include a specific ontology multiple times if the filtering clause consists of a join over multiple data streams represented in that ontology. The size of the ontology list, therefore, specifies the *degree* of the query. The degree represents the number of joins that must be performed for obtaining an answer. The filtering clause represents a combination of boolean conjunctive and disjunctive predicates. A device uses its cached data and context information including current geographical position and time of the day as inputs to these predicates. This allows a mobile device to place a *dynamic* query asking for the closest local gas station. It also allows a device to pose a *static* query, for example, asking for a Chinese restaurant located on the W 72nd Street. Along with string and numeric comparisons the filtering clause supports basic calculations, such as addition and multiplication. Additionally, the filtering

clause supports more advanced predicates based on the ontology specification, such as a distance computation between two geographical objects. The cardinality constraints of the query specify the minimum and maximum size of a required answer. Lastly, the temporal constraint specifies the relative deadline when the query should be completed. This is used by the device in order to query periodically its peers when time permits and the device has not yet cached a sufficient answer, given an implicit query.

#### 1.4.4 Data Management Layer

This section describes the most important layer for data management in mobile ad-hoc networks. It details the Information Manager, which is responsible for pro-active profile-driven discovering, processing, combining, and storing of data available in the environment. The Information Manager is also responsible for evaluating the integrity of peer devices and the accuracy of peer provided information, in order to provide the best results to its local Information Consumers.

##### 1.4.4.1 Information Manager

An Information Manager is responsible for majority of data management functions and partially for underlying network communications. From the data management perspective, Information Manager must be able to discover available sources, construct dynamic indexes and catalogs, support queries, and provide caching mechanisms for addressing the dynamic nature of the environments. From the networking perspective, Information Manager must also be able to discover and interact with remove devices, and route messages between them.

Each Information Manager maintains information about Providers and Consumers present on the same device as the Information Manager. This information includes the *lifetime* of each Provider, their service models, process models, and their query restrictions. Each Information Manager also maintains information about peers in its vicinity. This information includes the identity of devices – a unique identification number similar to an Internet Protocol address, and types of information they can provide, *i.e.*, Provider advertisements. Lastly, Information Manager maintains a data cache for storing information obtained from other mobile devices as well as the information provided by its local Providers, *i.e.*, answers to previous queries. Additionally, each Information Manager may include a user profile reflecting the user's preferences and needs. The Information Manager uses the profile to adapt its caching strategy and to initiate collaboration with peers in order to obtain missing required information.

##### 1.4.4.2 Complexity Levels

Since devices can range from sensors to laptops the framework does not require all devices to implement the same set of functionalities. Instead, the framework differentiates among five types of Information Managers based on their complexity levels.

In the simplest case (type 0), the Information Manager maintains at most one local Provider. It does not cache any remote information and it does not possess any reasoning or parsing mechanisms. This Information Manager only periodically *broadcasts* data sent to it by the Provider. This type of Information Manager is well suited for extremely resource limited devices, such as a store beacon whose only task is to advertise the presence of its store.

On the other hand, devices wishing to interact in more intelligent manner and those that possess more resources must implement an Information Manager that is able to maintain information about multiple local and *remote* Providers. These types of Information Manager must also be able to parse messages, route message to other peer devices and pro-actively query peers. Based on the varying collaboration level four additional types of Information Managers are possible:

1. Information Manager does not cache any remote advertisements or answers to queries,

2. Information Manager caches remote advertisements only for the *lifetime* specified in the message or until replaced by another entry,
3. Information Manager caches both advertisements and answers, and
4. Information Manager caches all advertisements and answers, and makes them available to other peers. This type of an Information Manager can effectively serve as a temporary partial catalog for all peers in the current vicinity.

In order to present all functionalities of the Information Managers, the following description assumes the most advanced type of an Information Manager, *i.e.*, type 4. The remaining part of this section presents the most important components of the Information Manager including:

- Data and Service Discovery Component
- Query Processing Component
- Join Query Processing Component
- Caching Component
- Transaction Component
- Reputation Component

#### 1.4.4.3 Data and Service Discovery Component

An important aspect of the framework is to discover local and remote Information Providers. The discovery allows each Information Manager to construct a temporary catalog representing current data and data sources in the vicinity. The MoGATU framework supports both push and pull based approaches, *i.e.*, each Information Manager can advertise its capabilities from solicit capabilities of other peers. The frequencies of advertisements and soliciting queries are tunable parameters. In order to restrict the number of messages in the environments, solicitation and advertisements are limited to *one-hop* neighbors only.

#### 1.4.4.4 Query Processing Component

While advertising and soliciting for Information Providers is an important functionality, the key objective of an Information Manager is to provide querying capabilities. The querying is initiated by an Information Consumer, which sends the Information Manager a query annotated in OWL, as defined above in Section 1.2.

The query includes a set of used ontologies ( $O$ ), selection list ( $\sigma$ ), filtering statement ( $\theta$ ), cardinality ( $\Sigma$ ), and temporal ( $\tau$ ) constrains. The set of ontologies also represents the service model that can be used to answer the query. A query can represent a selection over a certain data set or it can involve a join over multiple data streams. The later scenario is addressed in Section 1.4.4.5.

An Information Manager matches the query against entries in its cache. Each entry in the cache represents an unexpired answer to a previous query (otherwise it would be removed), or an advertisement for some local or remote Provider. The Information Manager parses the query and each entry according to OWL rules and relationships specified in the involved ontologies. The Information Manager compares service models  $s$ , *i.e.*, the set of required ontologies, and validates the query against inputs restrictions  $i$ , *i.e.*, the filtering and selection statements of the specific query. For cached answers, the Information Manager matches input values of the query, against those in the cached answer. The approach is equivalent to using traditional forward chaining methods, used by DATALOG / Prolog based query processing techniques [Gaasterland and Lobo, 1994; Grant et al., 1997]. The Information Manager first tries to find and return a cached answer. Otherwise, the Information Manager tries to find a local or remote Provider, in that order.

#### 1.4.4.5 Join Query Processing Component

Often, an Information Consumer can ask a query that requires a device to join *horizontally* data streams from multiple devices holding the same type of data. The device may also have to join *vertically* data streams from different devices as they become available. The term *data stream* is used to represent any source that is able to provide data given a specific query and also to represent the “streaming”-data sources defined by the sensor network community.

To allow devices to answer queries involving multiple sources, a collaboration protocol is defined based on the principles of Contract Nets [Avancha et al., 2003; Smith, 1988]. The Collaborative Query Processing protocol enables a mobile device to query its vicinity and locate peer data sources matching a given query. The protocol allows the Information Manager to obtain data matching any query, irrespective of whether the query is a *selection* or a *join*. It extends the traditional concept of nested loop joins and the simple selection query algorithm from above.

The Collaborative Query Processing protocol allows two or more Information Managers to cooperate by executing any combination of select-project-join queries. The protocol accomplishes the task by subdividing queries. Each sub-query can be assigned to a different device. The assignment is determined according to the available resources of devices present in the vicinity. For example, the Collaborative Query Processing protocol allows a tourist to use her handheld device to ask for the closest, cheapest laundromat that is open given her current location, time of the day, and a price range. The protocol also allows the tourist to ask for the closest laundromat adjacent to a Chinese restaurant – a query requiring a join over two data *streams*.

#### 1.4.4.6 Caching Component

Another key component of an Information Manager is caching. Each Information Manager stores query answers together with advertisements and registrations of local and remote Providers in a cache. In order to provide answers to, at least the expected, user explicit queries, *i.e.*, to overcome the spatio-temporal variation of data and data source availability, an Information Manager must utilize the cache in the most effective manner. MoGATU supports the traditional LRU and MRU replacement algorithms; however, as shown by [Perich et al., 2004a], these two approaches are highly ineffective. This is because an Information Manager must utilize the knowledge included in a user’s profile in order to improve cache *effectiveness*.

For caching, the Information Manager should use the profile in two ways: (i) to allocate space for specific data type and (ii) to assign utility value to each entry. In the first case, the Information Manager uses the profile to determine types of standing queries. The Information Manager uses these types to reserve portions of the cache for the related data types, *e.g.* traffic. MoGATU applies the first heuristic for defining two hybrid LRU+P and MRU+P algorithms and both heuristics for defining a semantic cache algorithm S+P [Perich et al., 2004a].

#### 1.4.4.7 Transaction Component

Maintaining data consistency between devices in distributed mobile environments has always been, and continues to be, a challenge. In order to operate correctly, devices involved in a transaction must ensure that their data repositories remain in a consistent state. While stationary nodes often embody powerful computers located in a fixed, wired infrastructure, this is not an option for mobile devices in wireless ad-hoc networks. Since most traditional transactions rely on infrastructure help, the use of such transactions is limited in these environments.

To address the problem, MoGATU also defines a novel transaction model – the Neighborhood-Consistent Transaction model (NC-Transaction) [Perich et al., 2003]. The focus of NC-Transaction is on maintaining consistency of transactions. This has generally been termed as the most important ACID property of transactions for mobile environments [Dunham et al., 1997]; however, it is not critical for read-only transactions [Perich et al., 2002b].

NC-Transaction provides a higher rate of successful transactions in comparison to models designed for traditional mobile computing environments. NC-Transaction maintains neighborhood consistency among devices in the vicinity. It does not ensure global consistency, a task often impossible since there is no guarantee that two devices will ever reconnect in mobile ad-hoc networks.

NC-Transaction accomplishes neighborhood consistency and high successful termination rate by employing active witnesses and an epidemic voting protocol. NC-Transaction defines witnesses as devices in vicinity that can *hear* both transacting devices and agree to monitor the status of a transaction. Each witness can cast a vote to commit or abort a transaction. A transacting device must collect a quorum of the votes, defined as a percentage of all witness votes, to decide on the final termination action for a transaction. By using a voting scheme and redundancy of witnesses, NC-Transaction ensures that transacting devices terminate in a consistent state. Additionally, information stored by each witness can be used to resolve conflicts between devices involved in a transaction.

#### 1.4.4.8 Trust Component for Evaluating Data and Source Integrity

In the preceding discussion it was assumed that all Information Providers and Information Managers are reliable. These components explicitly assume that answers provided are correct and do not verify the veracity of the information or their providers. This assumption is suitable for most mobile client-server environments; however, it is not suitable for peer-to-peer environments as they lack the intrinsic stability of *anchored* sources. In mobile peer-to-peer environments, some sources may provide faulty information due to malice or ignorance, which can lead to incorrect conclusions. Consequently, devices need a mechanism to evaluate the integrity of their peers and the accuracy of peer provided information.

To address this problem MoGATU introduces an additional feature of an Information Manager. This feature depends on distributed trust and beliefs in order to evaluate data and device integrity [Perich et al., 2004c]. In this belief-driven model, each device maintains and shares beliefs regarding the degree of trust it has for its peers. This trust is determined by previous experience and reputation made by other devices in the environment. Additionally, each device associates a value indicating its belief in the accuracy of the information the device holds. Each device, when querying its peers, uses the trust it has placed in the peers, in conjunction with the peers' accuracy belief of their information, to determine the reliability of the responses to its query.

### 1.4.5 Communications Layer

The lowest level of MoGATU deals with the networking aspect of the mobile ad-hoc networks. The communication layer is responsible for discovering devices and for reliable exchange of data among devices. This layer is implemented using Communication Interfaces, which abstract different ad-hoc networking technologies, such as Bluetooth or Ad-Hoc IEEE 802.11 standards.

#### 1.4.5.1 Communication Interface

To support multiple types of networking interfaces and to abstract these types from an Information Manager, each device implements at least one Communication Interface. A Communication Interface provides a common set of interfaces for discovering neighboring devices and for communicating with them.

Every Communication Interface registers its capabilities with its local Information Manager. Upon start-up, the Communication Interface sends a registration message including the network type  $n$  and process model  $p$  encoded in OWL:

$$\boxed{\text{registration} = (n, p)} \quad (1.3)$$

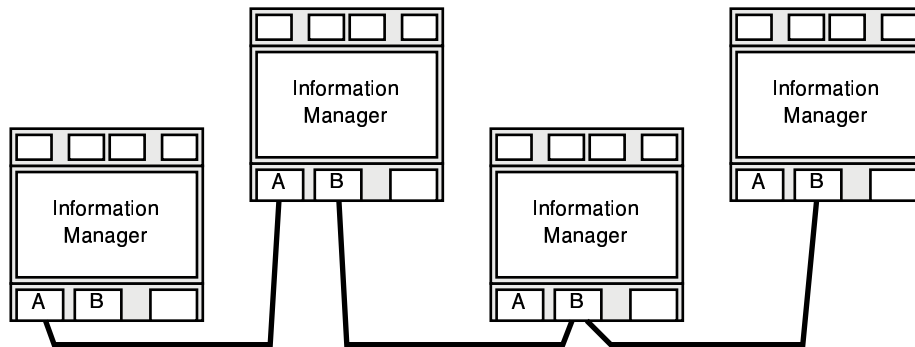


Figure 1.4: Sample routing across multiple networking technologies.

The network type is a specific service instance of the OWL-S ontology, while the process model allows the Information Manager to interact with Communication Interfaces in the same manner as with Information Providers. While a Communication Interface is responsible for sending and receiving data over the transmission medium, the Information Manager is still *network aware*. This is because it can infer the network constraints and requirements from the information contained in the registered capabilities. The advantage of using an abstract representation for the underlying networks is two fold:

First, the addition of a new networking technology does not require changes to the Information Manager component. An Information Manager is not burdened by different packet formats and message sizes for different networking technologies.

Moreover, the abstraction allows an Information Manager to route data across multiple network technologies at once. An Information Manager can accept data over one network technology and route it through an interface of another network technology. For example, this allows an Information Manager to talk to its peer using Bluetooth while the peer forwards the data to another peer using its Ad-Hoc IEEE 802.11 interface. One such scenario is illustrated in Figure 1.4.

This is because each Information Manager only maintains information about what Communication Interface it needs to use in order to interact with a specific peer. Additionally, since the underlying network is hidden, Information Managers use only the identity of the peer Information Manager as a destination of data instead of the network-specific address, which could otherwise be incompatible with other network standards. This is similar to the concept of Internet Protocol addresses allowing devices connected to the Internet to communicate over heterogeneous network technologies like Ethernet and ATM.

An Information Manager can thus abstract its current vicinity as a graph, where nodes represent other devices in the environment and edges represent a connection between two peers over any technology. The Information Manager can then apply any link state or dynamic vector routing algorithm for computing path to a desired destination, *e.g.*, AODV, DSDV, or DSR.

In *data-intensive* environments, an answer can often be provided by more than one device, *e.g.* cached by Information Managers, or available by local Providers. The querying source may not be interested in who it interacts with as long as it receives a correct answer. In order to improve the performance of the system, an Information Managers can intercept routed queries at the communication level.

MoGATU defines a hybrid mechanism combining discovery and routing for queries or data discovery among peer Information Managers in multi-hop networks. The algorithm uses a source-initiated approach similar to AODV and DSR. The algorithm works on a best-effort basis as it attempts to rebuild disconnected routes; however, it does not guarantee message delivery. More-

over, each Information Manager can intercept all messages it receives or routes in order to provide *shortcuts* for cached routes. Here, each Information Manager maintains a route entry for *one-hop* peers. The Information Manager also maintains a route entry for peers more than one hop away if those peers are used in on-going interactions or the Information Manager is caching advertisements for those peers.

---

## 1.5 Future Work

This section outlines the future work that is required for achieving the overall goal of data management and processing in mobile ad-hoc networks to allow individual devices to compute *what* information each device needs, *when* the device needs it, and *how* it can obtain the information. In particular, because of restrictions on query-answering power in mobile ad-hoc networks, one important direction of work will be on intelligently applying precomputed information.

For processing user queries, many devices use cached information, which can also be referred to as materialized views. In many current approaches, the cached data mainly represent stored answers to queries that a device issued in the past. Much is to be gained if one changes the current approaches to caching data on devices, by doing *purpose-driven* data caching and view materialization. Since previously cached information is kept around in order to enable processing of current queries, the first step is to determine which precomputed information would benefit precisely the expected current queries. In many scenarios, expected queries can be extracted from the current context, from past queries on the given device, or from user profiles stored on the device. Once the system has collected a workload of expected queries, purpose-driven data caching can be done *in advance*, in order to enable maximally efficient processing of queries. For instance, rather than caching full answers to some past queries, it makes sense to precompute and store on a device partial answers to multiple expected queries. A variety of approaches is possible, from purely ad-hoc to fully formal approaches [Chirkova et al., 2002; Afrati and Chirkova, 2005]. As mobile ad-hoc networks use relatively simplified ways of query processing, approaches based on multiquery optimization, such as [Roy et al., 2000], may be, however, able to guarantee better solution optimality than in standard standalone or distributed database scenarios. Additionally, to further improve the quality of stored data, precomputed for a given set of expected queries, it may be possible to use constraints that come from user profiles. Finally, it is also worth exploring restrictions on the size of cached data that need to be stored on a device in order to enable efficient processing of expected queries, such as [Chirkova and Li, 2003].

Once useful data are precomputed and cached on a device, one has to keep them consistent and up-to-date as the source data may be continuously updated in the network over time. To deal with changes in the source data over time, it is necessary to incorporate view-maintenance mechanisms [Roussopoulos, 1998] as well as lightweight approaches to concurrency control and invalidation [Gwertzman and Seltzer, 1996; Cao and Liu, 1998; Worrell, 1994].

Finally, as the location of a device may change over time, and thus the prevalent expected queries also change, there is a need to periodically change the *definitions* of the derived data that need to be stored on the devices, to accommodate new expected queries. Suppose a device already has access to “good-quality” locally cached data, which has been precomputed to address the needs of the device’s expected important queries. The next question is *how* to use the cached data to process the queries. As mobile ad-hoc networks tend to experience spatio-temporal data variation and have no guarantee of reconnection or collaboration in query processing, in many cases the best we can hope for is to get *approximate* answers to the queries. Among others, the reasons are that not all sources can be taken into account when answering a query and that stored data are not necessarily up-to-date. A promising direction here is to explore *purpose-driven* approximate query answering, where

a device chooses just some of the more useful sources for the purposes of the task related to the query. We can call this direction purpose-driven data integration, where, in particular, part of query processing is to use the advertised purposes of other data sources to collect only the information that is the most relevant for the query. In approximate query answering, we need to do formal analysis of the quality of answers to the queries using the currently available data. Another aspect of collaborative query answering is trust and security; one interesting direction here is to use, in query processing, trusted devices in the current context to verify the quality of information from potentially unreliable primary sources of the data [Perich et al., 2004c].

---

## 1.6 Chapter Summary

This chapter presented an overview of challenges arising in the area of mobile data management and surveys existing solutions, with the emphasis on data management in mobile ad-hoc networks. The chapter described origins of the mobile peer-to-peer computing model and related it to the traditional mobile data management models. The chapter then concentrated on the specific data management challenges due to three sources. One set of challenges is due to the nature of ad-hoc networking, including problems such as device discovery, message routing and restricted bandwidth. Next set of challenges relates to an information discovery in dynamic networks. The last set of challenges represents traditional data management issues, such as transactional support or consistency among data objects. While describing the problems, the chapter surveys proposed solutions to each problem category.

An important theme of this chapter is the fact that despite wireless ad-hoc networking technologies and peer-to-peer based data management paradigms attempting to solve similar problems, there is a very limited effort on cross-layer interaction, which is essential for mobile ad-hoc networks. The gap between the research on networking, data management, and context-awareness in pervasive computing environments is the fundamental problem of allowing a device to compute *what* information the device needs, *when* the device needs it, and *how* it can obtain the information. As an effort to address this issue, this chapter presented the MoGATU model [Perich et al., 2002a,b, 2003, 2004a,b,c] – a novel peer-to-peer data management model for mobile ad-hoc networks. The chapter outlined the underlying conceptual model and basic assumptions of MoGATU, which treats all devices in the environment as semi-autonomous peers guided in their interactions by profiles and local context. The chapter also presented the MoGATU abstraction of each device in terms of Information Providers, Information Consumers, and Information Managers, and details their respective concepts and functionality.

Finally, this chapter offered future work, which is required in order to bring the data management and processing in mobile ad-hoc networks close to their goal to allow individual devices to compute *what* information each device needs, *when* the device needs it, and *how* it can obtain the information.

---

## References

- S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: data management for asymmetric communication environments. In *Proceedings of ACM SIGMOD*, pages 199–210, 1995.
- F. Afrati and R. Chirkova. Selecting and using views to compute aggregate queries. In *Proceedings of ICDT*, 2005.
- K. Arnold, B. Osullivan, R. W. Scheifler, J. Waldo, A. Wollrath, B. O’Sullivan, and R. Scheifler. *The Jini Specification (The Jini Technology)*. Addison-Wesley, Reading, MA, June 1999. ISBN 0-201-61634-3.

- S. Avancha, P. D'souza, F. Perich, A. Joshi, and Y. Yesha. P2P M-Commerce in Pervasive Environments. In *ACM SIGecom Exchanges*, 2003.
- S. Avancha, A. Joshi, and T. Finin. Enhancing the Bluetooth Service Discovery Protocol. Technical report, CSEE, UMBC, August 2001. TR-CS-01-08.
- S. Avancha, A. Joshi, and T. Finin. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, 35(6):96–99, June 2002.
- P. Bahl and V. 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.
- H. Bharadvaj, A. Joshi, and S. Auephanwiriyakul. An Active Transcoding Proxy to Support Mobile Web Access. In *Proc. of the IEEE Symposium on Reliable Distributed Systems*, October 1998.
- Bluetooth SIG. Specification. <http://bluetooth.com/>.
- C. Bobineau, L. Bouganim, P. Pucheral, and P. Valduriez. PicoDBMS: Scaling down Database Techniques for the Smartcard. In *Proc. of the 26th International Conference on Very Large Databases*, 2000.
- M. Bratmann. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.
- D. Brickley and R. Guha. Resource Description Framework (RDF) Schema Specification 1.0 - W3C Recommendation. <http://www.w3.org/TR/2000/CR-rdfschema-20000327>, 2000.
- C. Brooks, M. S. Mazer, S. Meeks, and J. Miller. Application-Specific Proxy Servers as HTTP Stream Transducers. In *4th International World Wide Web Conference*, pages 539–548, Boston, Massachusetts, December 1995.
- O. Bukhres, S. Morton, P. Zhang, E. Vanderdijs, C. Crawley, J. Platt, and M. Mossman. A Proposed Mobile Architecture for Distributed Database Environment. Technical report, Indiana University, Purdue University, 1997.
- P. Cao and C. Liu. Maintaining strong cache consistency in the world wide web. *IEEE Transactions on Computers*, 47(4):445–457, 1998.
- D. Chakraborty, A. Joshi, T. Finin, and Y. 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.
- D. Chakraborty, F. Perich, S. Avancha, and A. Joshi. DReggie: A smart Service Discovery Technique for E-Commerce Applications. In *Workshop at 20th Symposium on Reliable Distributed Systems*, October 2001.
- D. Chakraborty, F. Perich, S. Avancha, and A. Joshi. DReggie: Semantic Service Discovery for M-Commerce Applications. In *Workshop on Reliable and Secure Applications in Mobile Environment, SRDS*, October 2001.
- D. Chakraborty, F. Perich, A. Joshi, T. Finin, and Y. Yesha. A Reactive Service Composition Architecture for Pervasive Computing Environments. In *7th Personal Wireless Communications Conference (PWC)*, pages 53–62, Singapore, October 2002b.
- H. Chen, T. Finin, and A. Joshi. Semantic Web in a Pervasive Context-Aware Architecture. *Artificial Intelligence in Mobile System 2003*, pages 33–40, October 2003.

- M. Cherniak, E. Galvez, D. Brooks, M. Franklin, and S. Zdonik. Profile Driven Data Management. In *28th International Conference on Very Large Databases*, August 2002.
- R. Chirkova, A. Halevy, and D. Suciu. A formal perspective on the view selection problem. *VLDB Journal*, 11(3):216–237, 2002.
- R. Chirkova and C. Li. Materializing views with minimal size to answer queries. In *Proceedings of PODS*, 2003.
- Cormen, Leiserson, Rivest, and Stein. *Introduction to Algorithms*. Prentice-Hall, 2001.
- S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz. An Architecture for a Secure Service Discovery Service. In *Fifth Annual International Conference on Mobile Computing and Networks (MobiCom '99)*, pages 24 – 35, Seattle, WA, 1999.
- DARPA. DARPA Agent Markup Language + Ontology Inference Layer (DAML+OIL). <http://www.daml.org>.
- A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and B. B. Welch. The bayou architecture: Support for data sharing among mobile users. In *IEEE Workshop on Mobile Computing Systems & Applications*, 1994.
- A. K. Dey and G. D. Abowd, editors. *Towards a Better Understanding of Context and Context-Awareness*, The Hague, The Netherlands, April 2000. GVU, Georgia Institute of Technology.
- M. Dunham and A. Helal. Mobile Computing and Databases: Anything New? *SIGMOD Record*, pages 5–9, 1995.
- M. Dunham, A. Helal, and S. Balakrishnan. A Mobile Transaction Model That Captures Both the Data and Movement Behavior. *Mobile Networks and Applications*, pages 149–162, 1997.
- C. S. Ellis and R. A. Floyd. The Roe File System. In *3rd Symposium on Reliability in Distributed Software and Database Systems*, pages 175 – 181, Clearwater Beach, Florida, 1983. IEEE.
- K. P. Eswaran, J. Gray, R. A. Lorie, and I. L. Traiger. The Notion of Consistency and Predicate Locks in a Database System. *Communications of the ACM*, 19(11):624–633, December 1976.
- M. Franklin. Challenges in ubiquitous data management. In *Informatics*, 2001.
- T. Gaasterland and J. Lobo. Qualified Answers That Reflect User Needs and Preferences. In *VLDB*, 1994.
- D. Goodman, J. Borrás, N. Mandayam, and R. Yates. INFOSTATIONS : A New System Model for Data and Messaging Services. In *Proc. of IEEE VTC'97*, volume 2, pages 969–973, 1997.
- J. Grant, J. Gryz, J. Minker, and L. Raschid. Semantic Query Optimization for Object Databases. In *ICDE*, 1997.
- E. Guttman, C. Perkins, J. Veizades, and M. Day. RFC 2068: Service Location Protocol, version 2, 1999. <ftp://ftp.isi.edu/in-notes/rfc2068.txt>.
- R. G. Guy, P. L. Reiher, D. Ratner, M. Gunter, W. Ma, and G. J. Popek. Rumor: Mobile data access through optimistic peer-to-peer replication. In *ER Workshops*, pages 254–265, 1998.
- J. Gwertzman and M. I. Seltzer. World wide web cache consistency. In *USENIX Annual Technical Conference*, pages 141–152, 1996.

- B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer-Verlag, 4th edition, May 1997.
- J. Holliday, D. Agrawal, and A. E. Abbadi. Database replication using epidemic communication. In *European Conference on Parallel Processing*, pages 427–434, 2000.
- T. Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in data management. *Communications of the ACM*, 37(10):18–28, 1994.
- T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, pages 352–372, May/June 1997.
- J. Ioannidis, D. Duchamp, and G. M. Jr. IP-based protocols for mobile internetworking. In *ACM SIGCOMM Symposium on Communication, Architecture and Protocols*, pages 235–243, Zürich, Suisse, 1991.
- D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*. 1996.
- A. Joshi. On Proxy Agents, Mobility and Web Access. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2000.
- A. Joshi, R. Weerasinghe, S. P. McDermott, B. K. Tan, G. Bernhardt, and S. Weerawarana. Mowser: Mobile Platforms and Web Browsers. *Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS)*, 8(1), 1996.
- P. Keleher and U. Cetintemel. Consistency Management in Deno. *ACM MONET*, 1999.
- S. Kent and R. Atkinson. IP Encapsulating Security Payload. World Wide Web, <http://www.ietf.org/rfc/rfc2406.txt>, November 1998.
- J. J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. In *Thirteenth ACM Symposium on Operating Systems Principles*, pages 213–225, Asilomar Conference Center, Pacific Grove, U.S., 1991. ACM Press.
- G. Klyne, F. Reynolds, and C. Woodrow. Composite Capabilities/Preference Profiles (CC/PP): Structure and Vocabularies. World Wide Web, <http://www.w3.org/TR/CCPP-struct-vocab/>, March 2001.
- H. Kottkamp and O. Zukunft. Location-Aware Query Processing in Mobile Database Systems. In *Selected Areas in Cryptography*, pages 416–423, 1998.
- O. Lassila and R. Swick. Resource Description Framework. <http://www.w3.org/TR/1999/REC/rdf-syntax-19990222>, 1999.
- S. Lauzac and P. Chrysanthis. Utilizing Versions of Views within a Mobile Environment. In *Proceedings of the 9th International Conference on Computing and Information*, 1998.
- Z. M. Mao, E. A. Brewer, and R. H. Katz. Fault-tolerant, Scalable, Wide-Area Internet Service Composition. Technical report, CS Division, EECS Department, University of California, Berkeley, January 2001.
- S. Muench and M. Scardina. XSLT Requirements. World Wide Web, <http://www.w3.org/TR/xslt20req>, February 2001.
- M. Oezsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Inc., New Jersey, 2nd edition edition, 1999.

- F. Perich, S. Avancha, D. Chakraborty, A. Joshi, and Y. Yesha. Profile driven data management for pervasive environments. submitted to a conference, 2002a.
- F. Perich, S. Avancha, D. Chakraborty, A. Joshi, and Y. Yesha. Profile Driven Data Management for Pervasive Environments. In *13th International Conference on Database and Expert Systems Applications (DEXA 2002)*, Aix en Provence, France, September 2002b.
- F. Perich, A. Joshi, T. Finin, and Y. Yesha. On Data Management in Pervasive Computing Environments. *IEEE Transactions on Knowledge and Data Engineering*, 2004a. accepted for publication.
- F. Perich, A. Joshi, Y. Yesha, and T. Finin. Neighborhood-Consistent Transaction Management for Pervasive Computing Environments. In *14th International Conference on Database and Expert Systems Applications (DEXA 2003)*, Prague, Czech Republic, September 2003.
- F. Perich, A. Joshi, Y. Yesha, and T. Finin. Collaborative Joins in a Pervasive Computing Environment. *VLDB Journal*, December 2004b.
- F. Perich, J. L. Undercoffer, L. Kagal, A. Joshi, T. Finin, and Y. Yesha. In Reputation We Believe: Query Processing in Mobile Ad-Hoc Networks. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, August 2004c.
- C. Perkins. RFC 3220: IP Mobility Support for IPv4, 2002. <http://www.ietf.org/rfc/rfc3220.txt>.
- C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.
- C. Perkins and E. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.
- E. Pitoura. A Replication Schema to Support Weak Connectivity in Mobile Information Systems. In *Database and Expert Systems Applications*, pages 510–520, 1996.
- C. Pulella, L. Xu, D. Chakraborty, and A. Joshi. A Component based Architecture for Mobile Information Access. In *Workshop in conjunction with International Conference on Parallel Processing*, pages 65–72, August 2000.
- J. Rekish. 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>.
- N. Roussopoulos. Materialized views and data warehouses. *SIGMOD Record*, 27(1), 1998.
- P. Roy, S. Seshadri, S. Sudarshan, and S. Bhoje. Efficient and extensible algorithms for multi query optimization. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 249–260. ACM, 2000. ISBN 1-58113-218-2.
- R. Sessions. *COM and DCOM: Microsoft's Vision for Distributed Objects*. John Wiley & Sons, New York, NY, October 1997.
- R. Smith. *Readings in Distributed Artificial Intelligence*, chapter The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. 1988.
- Sun Microsystems. Jini architecture specification. <http://www.sun.com/jini/>.

- C. Tait, H. Lei, S. Acharya, and H. Chang. Intelligent File Hoarding for Mobile Computers. In *Proc. of the First ACM International Conference on Mobile Computing and Networking - MobiCom'95*, 1995.
- The OWL Services Coalition. OWL-S: Semantic Markup For Web Services. <http://www.daml.org/services/>, 11 2003.
- J. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, and A. Joshi. A Secure Infrastructure for Service Discovery and Access in Pervasive Computing. *ACM MONET: Special Issue on Security in Mobile Computing Environments*, 8(2):113–125, 2003.
- UPNP Forum. Understanding Universal Plug and Play: A White Paper. [http://upnp.org/download/UPNP\\_UnderstandingUPNP.doc](http://upnp.org/download/UPNP_UnderstandingUPNP.doc).
- J. Veizades, E. Guttman, C. E. Perkins, and S. Kaplan. RFC 2165: Service location protocol, June 1997.
- G. Walborn and P. Chrysanthis. PRO-MOTION : Management of mobile transactions. In *Selected Areas in Cryptography*, pages 101–108, 1997.
- J. R. Walker. Unsafe at any key size; An analysis of the WEP encapsulation. IEEE Document 802.11-00/362, October 2000.
- World Wide Web Consortium (W3C). Ontology web language (owl). <http://www.w3.org/>.
- K. J. Worrell. Invalidation in large scale network objects caches. Master's thesis, 1994. URL [citeseer.ist.psu.edu/worrell194invalidation.html](http://citeseer.ist.psu.edu/worrell194invalidation.html).
- B. Yang and H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *27th VLDB Conference*, 2001.
- S. Yi, P. Naldurg, and R. 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.
- 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.
- A. B. Zaslavsky and Z. Tari. Mobile computing: Overview and current status. *Australian Computer Journal*, 30(2):42–52, 1998.
- B. Zenel. *A Proxy Based Filtering Mechanism for The Mobile Environment*. PhD thesis, Department of Computer Science, Columbia University, New York, NY, December 1995.