Context-Aware Middleware Abstractions for Ad Hoc Mobile Computing

Christine Julien
Department of Computer Science and Engineering
Washington University
Saint Louis, MO 63130
julien@cse.wustl.edu

Abstract

Some of the most dynamic systems being built today consist of physically mobile hosts and logically mobile agents. Such systems exhibit frequent configuration changes and a great deal of resource variability. Applications executing under these circumstances need to react continuously and rapidly to changes in operating conditions and must adapt their behavior accordingly. Applications with these capabilities are referred to as context-aware. Much of the current work on context-aware computing relies on information directly available to an application via context sensors on its local host, e.g., user profile, host location, time of day, resource availability, and quality of service measurements. The goal of this proposal is to develop a new perspective on context-awareness, in which the context includes, in principle, any information available in the ad hoc network but is restricted, in practice, to specific projections of the overall context. This work aims to design and implement a middleware model that brings this notion of context to the application programmer. The unique features of this middleware model will be: (a) the declarative abstract specification of task-specific views of the global context; (b) the ability to redefine these views dynamically; (c) the continuous maintenance of the desired contextual information despite the fact that interactions among hosts in the underlying network are transient and disconnections are frequent; and (d) flexible support for a range of context-aware interactions. This proposal outlines the specific goals of the research, the issues expected to be encountered, and the intended ultimate results of the research.

1 Introduction

The ubiquity of computing devices in our society opens the user's computing environment to a dynamic world where the network topology, or the communication connections between the hosts in the network, constantly change. Software must continuously adapt to this changing context. This proposal focuses specifically on the needs of applications executing in ad hoc mobile networks. Ad hoc networks form opportunistically and change rapidly in response to the movement of the mobile hosts.

This highly dynamic physical structure supports an even more fluid logical structure. Application level components, henceforth referred to as agents, possess the ability to move among the mobile hosts. This agent mobility removes the static binding between the application agents and the hosts where they are executing. Agent mobility also allows distributed applications to achieve a higher degree of flexibility and customizability, while improving bandwidth utilization.

Given this unique environment, software development becomes more complex because the software must function in a setting that is increasingly open and dynamic. Applications will require behavior that is opportunistic, highly adaptive, and very much dependent on the availability of various resources which may also be transient in nature. While it is not possible to eliminate the intrinsic complexity of software that operates under such demanding circumstances, this research aims to reduce the complexity of the application development task by shifting much of the burden to the support infrastructure. A new and high level of abstraction will provide the developer increased

programming power. Effective communication of the application's needs to the system support infrastructure will allow more efficient use of the limited resources inherent in the ad hoc network environment. The research proposed here focuses on the use of *context-aware computing* to achieve rapid development of adaptive ad hoc mobile applications.

Context-aware computing refers to a software system's explicit ability to detect and adapt to changes in its environment, e.g., a drop in the quality of service on a video transmission, a low battery level, or the sudden availability of much needed access to the Internet. Most current facilities supporting context-awareness sense simple facets of the environment (e.g., location or temperature) that can be measured locally by sensors on the host. When the needs of the application must reach beyond the basics (e.g., to use services available at a remote location), the programmer needs to contend with more complex processes that include discovery and communication. While these extra costs may be acceptable in wired networks where connections persist over extended periods of time, in ad hoc networks the complexity of managing frequent disconnections can significantly increase the programming effort. Yet mobile systems do need access to a broad range of resources, maybe even more so than distributed applications. Of great interest is the ease with which resources can be acquired and retained in the presence of mobility. In ideal circumstances, a programmer simply declares a resource as being within the scope of a program. We actually do so when we declare program variables or when we provide directives to a compiler to include certain packages. This works because of the relatively static nature of the setting in which the program ends up being used. This research extends this notion of declaration to a much broader set of resources and provides the mechanisms needed to maintain access to the specified resources despite rapid changes in the environment caused by the mobility of hosts, migration of software components, and changes in connectivity. For instance, an application on a palm top should be able to declare its need for printer access, and, as the owner moves, a printer should always appear available, as long as some printer is within wireless communication range. Of course, building such an application with today's technology is feasible, but coding it cannot be reduced to the simple act of providing a declaration in the program. In fact, programming it with current technology requires the developer to constantly deal with explicitly discovering wireless neighbors and creating and maintaining communication links with discovered neighbors. This proposal contends that applications requiring such contextual information can be realized by extending the notion of context-aware computing and by developing a software infrastructure that continuously and transparently secures the resources declared by the application.

The remainder of this proposal is structured as follows. Section 2 introduces a new perspective of context-aware computing. Section 3 discusses work in some of the areas related to different components of this proposed research. Section 4 outlines some anticipated research issues, and Section 5 provides a research plan. Conclusions appear in Section 6.

2 A New Perspective of Context-Aware Computing

This proposal is driven by the desire to simplify the development of mobile applications for ad hoc networks. This section discusses the overall solution strategy in more detail and explores some of the potential implications of this research.

2.1 Computational Model

This work assumes a computing model in which hosts can move in physical space, and the applications they support are structured as a community of software agents that can migrate from one host to another whenever connectivity is available. Thus an agent is the unit of modularity, execution, and mobility, while a host is a container characterized by, among other things, its location in physical space. Communication among components and agent migration can occur whenever the hosts involved can physically communicate with each other, i.e., they are connected. Connections may be wired or wireless as hosts may be either mobile or stationary. Since the notion of context is always relative, the term *reference agent* will denote the agent whose context is being considered, and *reference host* will refer to the host on which that agent is executing. An agent's location is always a host, while a host's location is always a point in some physical or logical space.

An ad hoc network is defined as a closed set of reachable hosts. In principle, the context associated with a given agent consists of all the information available in the ad hoc network. Of course, such broad access to information is generally costly to implement. In addition, various parts of the same application may need different resources

at different times during a program's execution. For this reason, this proposed research focuses on structuring the context in terms of fine-grained units called views. A *view* is a projection of the maximal context, packaged with an interpretation that defines the rules of engagement between the agent and the data available in the view. An agent defines one or more views (which can be redefined over time) and can operate on each view in a manner compatible with the view definition. The agent may operate on the actual contents of the view directly or indirectly, depending on the abstract interpretation associated with the view. This context interpretation is generally built-in and exposes the view's contents to the agent by means of some concrete representation. For instance, the view contents may be restricted to a set of objects located on hosts within a certain distance, but the reference agent may see them as a list of points in some virtual space. Because tuple spaces based on the Linda model of coordination have enjoyed great success in coordination models tailored to the ad hoc environment, this research will use tuple spaces for coordination and leave open the issue of providing more generalized context-aware data structures in the future.

2.2 Declarative Specification of Views

The concept of view is agent-centric because every view is defined relative to a reference agent and with respect to its individual needs for resources from and knowledge about its environment. An agent sees the world through a set of these views. The set may be altered at will by the agent defining, redefining, and deleting views as processing requirements demand. The software engineering gains will be determined to a great extent by the level of flexibility and simplicity the abstractions offer the application programmer. This research centers around declarative specifications which allow an application agent to describe its contextual needs to the underlying context maintenance system by specifying a projection of the maximal context. A rich set of criteria can be employed towards this purpose, and full flexibility is achievable only if every element of the underlying computational model can be constrained by this specification. For instance, one ought to be able to describe the view contents in terms of phrases such as:

> All empty printer queues (reference to objects) posted by color laser printers (reference to agents) within 50 meters (implicit reference to hosts) of my current location (property of the reference host).

In general, constraints on the attributes of the desired resources (data or objects) and the agents that own them would be an effective way to restrict the view's contents. The unique properties of ad hoc networks, however, force the combination of these constraints with ones on the attributes of the hosts on which the agents reside and with properties of the ad hoc network in the immediate vicinity. Security and network considerations will likely emerge as important research issues in any effort to design a language for view specifications. At the network level, for instance, it may be desirable to limit context to a connected subnet of the ad hoc network forming a region around the reference host. The network topology, geometry, physical distribution in space, and security enforcement procedures may play a role in determining the shape of the region of interest. These considerations are new to context-awareness and arise from the focus on ad hoc mobility.

2.3 Transparent Context Maintenance

The adoption of a declarative context specification is motivated by the expectation that transparent context maintenance will shift much of the programming burden to the underlying middleware. An emergency response application, for example, may be designed to help enforce a policy that requires all members of an earthquake damage assessment team to remain in communication at all times and not to stray farther than some particular distance. To accomplish this, we can place a monitor agent on each inspector's PDA. The context built for this agent is a set of profiles that include the physical location of the respective team member as obtained from a GPS device. As individuals join or leave the team, the system automatically updates the set of profiles without the reference agent's involvement. As team members move, their new locations are immediately available. From a programming perspective, the view functions like a local data structure automatically updated to reflect the overall context.

Several other benefits can be derived from employing a transparent context management strategy. First, a semantically precise specification of views and the implied formal guarantees provided by a correct implementation should ease program analysis, even when conducted informally. Second, the programmer has explicit control over

the cost associated with context management. The programmer controls the scope of the view (a large or small neighborhood), the size of the view (the range of entities included), and the relative cost of executing a particular operation on that view (by defining the level of consistency, e.g., best-effort or transactional semantics). Finally, there are also opportunities for optimization. By assigning the middleware the responsibility of managing all views and by having potential access to a complete specification of the global needs of all agents residing on the same host, one can identify common resources or find ways to avoid unnecessary conflicts. Section 4 explores these research issues in more detail. First, however, we take a closer look at some other relevant research.

3 Related Work

This work, while specifically investigating models and middleware for context-aware computing in ad hoc networks, touches many areas of computer science, unifying them in some ways and extending them in others. This section describes some current work in associated areas and how it relates to the research proposed.

3.1 Context-Aware Computing

Context-aware computing first came to the forefront in the early 1990's with the introduction of small, mobile computing devices. Initial investigations at Olivetti Research and Xerox PARC laid the foundation for the development of more recent context-aware software. Olivetti's ActiveBadge [26] uses infrared communication between badges worn by users and sensors placed in a building to monitor movement of the users in order to forward telephone calls to the user's location. PARC's PARCTab system [54], released in 1993, also uses infrared communication between users' palm top devices and desktop computers. It uses location information to allow applications to adapt to the user's environment. Applications developed for PARCTab perform activities ranging from simply presenting information to the user about his current location to attaching a file directory to a room for use as a blackboard by users in the room.

Current context-aware applications serve as tour guides by presenting information about the user's current environment. Cyberguide from Georgia Tech [1] and GUIDE from the University of Lancaster [12] are examples of two such systems. Fieldwork tools [44] automatically attach contextual information, e.g., location and time, to notes taken by a researcher in the field. Memory aids [41, 18] record notes about the current context that might later be useful to the user.

More recently, frameworks built to support the development of context-aware applications began to be developed. Among the best known systems is Georgia Tech's Context Toolkit [19], which uses an object-oriented approach to separate the sensing, gathering, and interpretation of the contextual information for each user. The Context Fabric [28], on the other hand, provides a service infrastructure for mediating contextual interactions. These systems take a first step towards providing a middleware layer between context sensing devices and applications that must adapt their behavior to this context information.

3.2 Middleware

Middleware supports the application development task by enhancing the level of abstraction associated with the programming effort and by adding mechanisms and services which are more specialized than those provided by the operating system. Distributed object systems, agent systems, and coordination systems each address different abstractions useful for dynamic mobile applications.

Distributed Object Systems. In systems such as the Object Management Group's CORBA (Common Object Request Broker Architecture) [21], Microsoft's COM (Common Object Model) [21], and Sun's Jini [20], objects serve as the main abstraction of the distribution in the network [49]. Key features include a lookup mechanism to identify the objects associated with particular services and common interfaces to enable interactions and to hide the distribution of components. For example, in Jini, service objects register with a centralized lookup-service which plays the role of matchmaker between clients and services. After a client finds a service, all interactions are performed in a location-transparent manner and without the aid of the lookup-service. Typically, object-based systems assume that a connection between a client and a service object is long-lasting, and therefore these systems do not address the possibility of disconnection. The DENO (Decentralized Network Objects) [30] system begins

to address this issue in the context of mobile and unreliable networks, adding replication of the objects to increase efficiency, availability, and fault tolerance.

Mobile Agent Systems. Mobile code and mobile agents can further enhance the programming model, increasing program performance by optimizing the placement of executing code during the application lifetime. One of the primary points of comparison for mobile agent systems is their support for either strong or weak mobility [22]. In weak mobility, a mobile agent migrates, carrying only its code and data state, and abandoning its execution state including the program counter and stack variables. This model is by far the most common as it requires little additional support from the runtime environment. Java-based systems such as Aglets [51] and mobile code systems such as μCode [40] support weak mobility. In strong mobility, the execution state is included when the agent migrates. The TCL scripting language, supported by D'Agents [24] offers strong mobility, and recent work has been done to enhance the Java VM [48] or modify Java byte code [50] to allow strong mobility in the Java language.

Other work aims to combine the mobile agent paradigm with distributed object systems. The ORB/OS Task Force [35] and SOMA (Secure and Open Mobile Objects) [46] address scalability and interoperability by exploiting CORBA interfaces and adding services to support mobile objects. Because the objects (or agents) in these systems adhere to the CORBA standards, they can interact with other agents only according to those specifications. That is, to locate one another, they must coordinate with the centralized server. Other extensions allow objects to maintain references when clients move [43, 3], but, in most cases, this is supported by a straightforward proxy and does not generalize to the mobile ad hoc environment. The FarGo framework [27] provides for relocation of objects at runtime, to follow user-specified semantics. The system, however, provides limited support to locate objects and no support to automatically rebind to a new object after a reference is dropped.

Coordination Systems. Coordination abstracts the behavior of the mobile units and focuses on high level communication protocols. The Linda tuple space coordination model was introduced and popularized in the mid 1980's to support parallel computation [23]. More recently, the distributed computing community has realized the power of this model and has produced several client-server tuple space implementations including IBM's TSpaces [29] and Sun's JavaSpaces [47]. Because these systems rely on a centralized tuple space, they do not adapt well to mobile systems where support for transient connections, decreased access to resources, and dynamic changes in context are needed. The MARS [7] and TuCSoN [36] systems exploit programmable Linda tuple spaces for coordination of logically mobile agents, essentially providing a tuple center on each host to support interaction among co-located agents. The LIME [33] middleware introduces the notion of reactive, transiently shared tuple spaces, a model which does not rely on any central server and supports both physical and logical mobility. In LIME, each agent is permanently bound to a tuple space, and when two agents can communicate, their tuple spaces are logically merged. PEERWARE [16] similarly exploits the notion of transiently shared data structures in peer to peer networks, focusing on data organized as a forest of trees and providing an event distribution mechanism over the trees. This event distribution mechanism extends the JEDI [15] distributed event model. Both PEERWARE and LIME have a symmetric sharing model which is not always appropriate, especially in large ad hoc networks.

3.3 Security

In such open environments as ad hoc networks, security becomes of heightened importance. When considering coordination among mobile agents, security concerns can be grouped into three categories: protecting hosts from malicious agents in the system, protecting agents from malicious hosts, and protecting data. The D'Agents system [24] uses public-key cryptography to authenticate incoming agents to increase the security of hosts. In this way, hosts can prevent malicious agents from operating on their systems. The more difficult problem of protecting agents from malicious hosts led to the notion of agents computing with encrypted functions [45]. These functions allow agents to conditionally decrypt code and data from hosts based on environmental conditions.

Mechanisms for protecting data can be divided into two categories: ensuring data integrity and controlling access to data. Because of the rise in distributed computing middleware, much research has focused on encrypting communication within a shared data space. SAMCat [34], a distributed file management system based on XML, uses SSL encryption and authentication to securely transmit tuples into and out of a globally persistent data space. Yalta [6] attempts to achieve a similar goal by allowing components to coordinate through a shared tuple space. Communication to and from the tuple space is encrypted, and users attempting to access the data must first be authenticated. In these two cases, the data in the tuple space is unencrypted, and because authentication

occurs on a system-wide scale, providing secure channels directly between components becomes more difficult. In the arena of access control provision, models and systems abound. The Mobile Ambients [8] model and the Seal framework [53] provide administrative domains that logically divide an execution environment into multiple nested levels. KLAIM (a Kernel Language for Agents Interaction and Mobility) [17] addresses the protection of data in agent systems through the use of a specialized type system. Finally, SecOS [52], based on the Linda coordination model, provides fine-grained access control by allowing agents to lock individual fields of tuples. Due to issues involving key secrecy, this system has trouble scaling from a single machine to a true distributed environment.

3.4 Ad Hoc Routing

As the mobile environment has evolved, so have the general algorithms to support mobile applications. Because of location changes, power restrictions, transient connections, and other facets of the mobile environment, traditional distributed algorithms cannot be directly adapted.

Creating paths between nodes in an ad hoc network is neither a new problem nor an easy one. Routing protocols for traditional wired networks do not function well in the ad hoc environment because of the special conditions encountered in this new type of network. Hosts in ad hoc networks are constantly moving, and hosts that are encountered once are likely never to be encountered again. Ad hoc routing protocols can generally be divided into two categories. Table-driven protocols, such as Destination Sequenced Distance Vector (DSDV) routing [38] and Clusterhead Gateway Switch Routing [13] mimic traditional routing protocols because they maintain consistent up-to-date information for routes to all other nodes in the network [42]. This class of algorithms is based on modifications to the classical Bellman-Ford Routing algorithm [10]. Maintaining routes for every other node in the network can become quite costly. Performance comparisons [5] have shown that, while the overhead of DSDV is predictable, the protocol can be unreliable. Additionally, the overhead can be lessened by utilizing routing protocols from the second class, source initiated on-demand routing protocols. This type of routing creates routes only when requested by a particular source and maintains them only until they are no longer needed. Ad-Hoc On-Demand Distance Vector (AODV) routing [39] builds on the DSDV algorithm but minimizes routing overhead by creating routes on demand. Dynamic Source Routing (DSR) [4] requires that nodes maintain routes for source nodes of which they are aware in the system. Finally, the Temporally Ordered Routing Algorithm (TORA) [37] uses the concept of link reversal to present a loop-free and adaptive protocol. It is source initiated, provides multiple routes, and has the ability to localize control messages to a small set of nodes near the occurrence of a topological change. Another type of routing that may lend itself well to ad hoc environments is Distributed Quality of Service Routing [9]. In this scheme, routes are chosen from the source to the destination based on network resources available along the path.

Communication with a subset of the nodes in a network is accomplished using multicast routing protocols. These protocols generally build a multicast tree or mesh over the network, and messages are later routed over this structure. Multicasting in ad hoc networks has received much attention as of late. Early approaches used the shared tree paradigm commonly seen in wired networks. Shared tree protocols have been adapted for the wireless environment to account for mobility in these systems [14, 25]. More recent work in ad hoc multicasting has realized that maintaining a multicast tree in the face of a highly mobile environment can drastically increase the network overhead. These research directions have led to the development of shared mesh approaches in which the protocol builds a multicast mesh instead of a tree [2, 31]. Both the multicast tree and mesh protocols use a shared data structure approach. That is, they assume that for a given multicast group, there may be multiple senders. These senders share the tree built for the group to route their messages.

3.5 Consistency

In ad hoc mobile settings, communication costs and unannounced disconnections affect the guarantees that can be provided to applications. The Coda File System [32] exploits the weak connectivity inherent in nomadic mobile systems to adapt the behavior of a distributed file system to the changing connectivity. The attention to consistency tradeoffs also appears outside of mobile systems within the TACT (Tunable Availability and Consistency Tradeoffs) toolkit [55]. TACT allows Internet services to specify their quality of service requirements in the form of availability and consistency tradeoffs. Applying a similar notion of application defined consistency in the ad hoc setting will allow applications which do not require strong guarantees to take advantage of data immediately available.

The MOST [11] project identifies the design requirements for systems supporting consistency in mobile applications. An extension of this work, the group execution service G-QEX, provides a set of services which gives programmers the ability to specify consistency parameters over properties such as the timing of message delivery, message ordering, reliability, and overall cost. Additionally, group membership algorithms can be adapted to provide looser guarantees with lower network overhead.

4 Research Issues

Because ad hoc mobility presents fundamental challenges to developing next generation applications, this work seeks to develop a new model of context-awareness able to accommodate the complexities of mobile computing, to build middleware that embodies this model, and to evaluate both on interesting application test beds. This section offers a discussion of major research issues likely to affect the effort to achieve these goals. Even though the results of this research will likely apply to a wide range of settings, this proposal focuses on its applicability to ad hoc mobility.

In many distributed systems, data access serves as the primary form of interaction among components. In mobile computing, several systems have used shared tuple spaces as a coordination medium. Many of these systems were discussed in the previous section. MARS is concerned with interactions among mobile agents and employs a single tuple space per host to facilitate coordination among co-located mobile agents. LIME relies on transient sharing of tuple spaces among agents on the same host and among hosts within communication range. This enables LIME to provide support for both physical and logical mobility. Other systems have explored different data structures. PEERWARE, for instance, stores documents in trees and adjusts the tree structure to account for mobility. All these systems assume a symmetric model of sharing; when a group of components is formed, they all share the same data, and they perceive it in the same manner. By contrast, the model introduced in Section 2 will allow each individual agent to define its own perspective of the data available in the world in terms of one or more views. This asymmetry, a distinguishing feature of this new model, allows each agent to assume responsibility for and control over the size and scope of the data it accesses. For example, an agent associated with a player participating in a commando war game exercise might define a view that includes the locations and activities of all the other team members within a certain distance, and this view's contents may be continuously adjusted as the game progresses. In general, the agent's view will contain a representation of a subset of the data available in the ad hoc network. The choice of representation is a defining feature of each specific instantiation of the general model. As indicated previously, the research proposed here focuses on using a tuple space as a basic representation. The choice of data included in the view, i.e., its contents, is determined by the declarative view specification. The latter is given by stating constraints on network properties (e.g., number of hops, Euclidean distance, bandwidth, etc.) so as to define a connected subnet immediately surrounding the reference host. This kind of locality is expected to help control the context maintenance costs while meeting the needs of most mobile applications. Within this contextual setting, further restrictions can be imposed on the properties of the physically mobile hosts (e.g., power availability, devices supported, etc.) in the subnet and of mobile agents supported by the admissible hosts. Finally, data associated with the remaining eligible agents can be filtered to produce the actual contents for a particular view. As hosts and agents move and properties of the network components change over time, the contents of the view must be transparently updated for the reference agent.

The dynamic nature of this view definition is illustrated in Figure 1, where the depicted view of agent `a1` changes as the distance between hosts `h1` and `h3` decreases. Agent `a1` is grayed to indicate that it is the agent specifying the view. Hosts, agents, and data items that contribute to the view are shown with darkened borders. In part (a) of the figure, due to some aspect of `a1`'s specification, only hosts `h1` and `h2` qualify to contribute agents to the view. Because of the restrictions on agent and data properties, only certain data items on certain agents on these hosts appear in the view. The balloon pointing to `a1` shows a table of the hosts, agents, and data items that should contribute to `a1`'s view. As part (b) shows, when host `h3` moves closer to `h1`, it satisfies the view's constraints. Again, only certain data items on certain agents appear in the view. Exactly which hosts, agents, and data items contribute is determined by the application-provided view specification, which is not specified for this particular example.

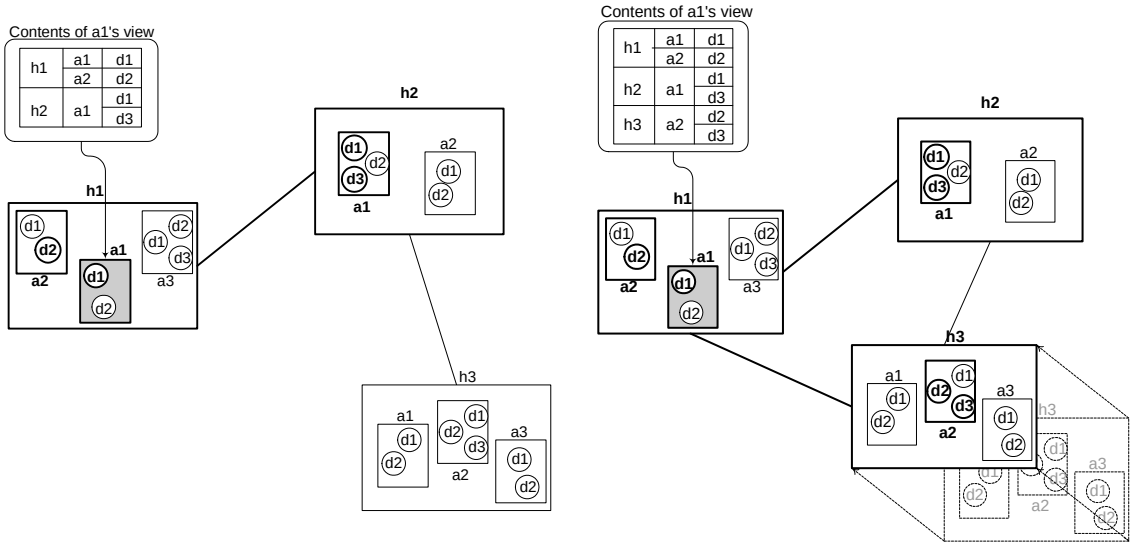

Figure 1. View dynamics. Data items visible to reference agent `a1` located on host `h1` before and after `h3` moves into `h1`'s range. Hosts, agents, and data items with darkened borders contribute to the view, while ones with lighter borders do not satisfy the specification.

4.1 Middleware

Issues. As suggested previously, this research focuses on speeding up the development of dependable applications in wireless ad hoc networks by shifting the complex context maintenance to an underlying middleware. This reliance on middleware facilitates rapid dissemination and adoption. In adopting a middleware model to this environment, several concerns immediately arise. They include issues dealing with data representation, access control, context maintenance in the presence of mobility of hosts and agents, and the mechanics of managing responses to the context changes triggered both by mobility and by other agents' actions, which may affect the data present in the agent's view.

Preliminary Ideas. Figure 2 attempts to identify the various areas of concern associated with the proposed research on middleware for context-aware computing. The grayed boxes denote areas that this research will directly address. The partially hashed boxes represent later areas of work that will build on this research. As part of the evaluation process of this work, these areas will be quickly prototyped to verify the correctness and usefulness of the underlying system. While the architecture of the middleware is likely to share the general layered approach depicted in the figure, the relationship among these layers is complex, and, due to performance and other considerations, the resulting design is likely to have a very different structure.

At one end of this figure lies the application code (a reference agent), which defines its requirements for contextual information and uses such information in a manner that is mostly oblivious to the realities of ad hoc networking. At the other end lies an ad hoc wireless network that is in a continuous state of flux due to the movement of hosts, the natural evolution of the application components, and the possible migration of code. In between lie components that allow the middleware to discover other wireless hosts in the neighborhood, to monitor context sensors both locally and on one-hop neighbors, to construct and maintain the context, and, finally, to allow the application to declare its needed context. This proposed research will further develop this model and use the final model to evaluate the concerns outlined above.

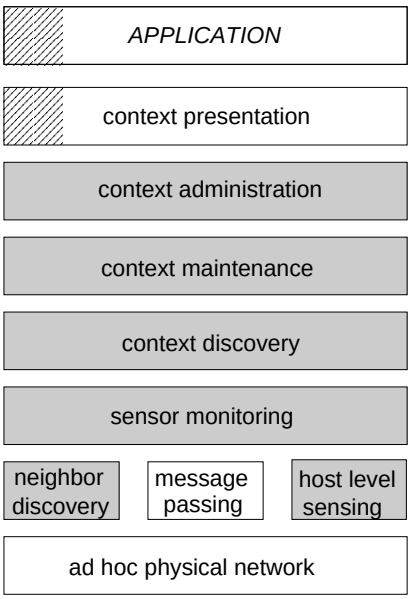

Figure 2. Middleware layers

4.2 Data Representation and Access

Issues. The manner in which each agent perceives and accesses data has ramifications involving the ease of programming and the efficiency of the operations over the data in the view. In addition, the chosen data representation should completely decouple the contents of the view from the representation of the contents that the reference agent sees. A proper decoupling will allow extensions of this model that operate over varied data representations. This should also allow increased decoupling among other system components and enhance the attainable level of interoperability.

Preliminary Ideas. Because this research proposes using tuple spaces for coordination among components, initial data access operations will most likely resemble traditional Linda tuple space operations. These operations allow coordinating components to place tuples in the tuple space, read tuples from the tuple space, and to remove tuples from the tuple space. Slight modifications of these operations and their semantics may provide increased flexibility for the application agents. Using such a basic data structure to represent the contents of the view should ease extensions to additional data structures. While exploration of these data structures is outside the scope of this proposed work, it is imperative that the design and implementation of the model account for this future extension.

4.3 Active Views

Issues. In addition to the core data access operations, an agent might also desire more powerful and flexible mechanisms for data interaction. For example, an agent may want to react to the appearance or disappearance of a piece of data. This style of interaction has proven useful in other coordination systems (e.g., MARS, LIME, and TuCSoN) designed for mobile systems. In general, any mechanisms that enable the agent to respond to changes in the set of available data enhance the application's capacity to detect and adapt to context changes.

Preliminary Ideas. Behaviors that this research proposes exploring include reactions to data, transparent migration of data from one agent to another, data replication, and data caching. This research will explore attaching such behaviors to individual views, thereby making the views *active*. A main concern in designing active views is taking care to integrate them into the model cleanly and naturally. The addition of behaviors to the model should also be done in a manner that is extensible, i.e., it should allow for the addition of new and unforeseen behaviors throughout the lifetime of the model and its implementation.

4.4 Consistency

Issues. The combination of distributed computing and a setting in which both hosts and agents are constantly moving renders any attempts to provide strong guarantees very costly. Even the simple case when the reference agent attempts to read a single piece of data can become complicated if the data happens always to be in the view, but the agent owning the data keeps moving around. In an implementation in which the read operation checks the agents on each host in turn, pinning down the agent with the data becomes costly, if not totally infeasible. These difficulties increase when attempting to provide a globally consistent view at all times, i.e., the equivalent of maintaining a consistent snapshot of a restricted region of the network. Yet there are applications in which, at least for limited periods of time, strong guarantees are required (e.g., security check points, wireless financial transactions, etc.).

Preliminary Ideas. Given the above concerns, it seems necessary to allow the reference agent to specify a range of guarantees that lie other than at the two extremes (best effort versus transactional semantics. This research hopes to discover ways of combining light weight constructs with formally stated guarantees. It is also important to allow different levels of consistency to coexist and to have the strength of the operations be controlled dynamically, e.g., high levels of consistency when resources are available and tolerable levels of inconsistency in a resource poor setting. Since a view is defined once and used over an extended period of time, it may make sense to associate the consistency level not with the operations employed but with the view itself. This direction may also open up new possibilities for more modular design.

4.5 Access Control

Issues. Authentication, encryption, and access control are security mechanisms typically employed in distributed computing. However, ad hoc mobility presents us with new challenges. First, there is the absence of a trusted server on which to rely for authentication purposes. Second, both hosts and agents can represent potential threats—agents can attack hosts and hosts can corrupt agents. Solutions to these two problems lie outside this proposed research. This research does plan to address access control mechanisms at the level of granularity of a single data item. Ideally, the access control mechanisms should allow an agent to grant access on a per-agent, per-data, per-operation basis. For example, if an agent requests to read a particular record, the agent owning that record may (implicitly) grant access if the requesting agent provides the correct password. However, if the same agent with the same password attempts to delete the same record, the owner may not allow it.

Preliminary Ideas. This research plans to examine the essential features of general access control policies designed to respond to the specific needs of agent coordination in the presence of logical and physical mobility. This will lead to the development of an access control construct to be integrated with the proposed coordination model. A possible solution to achieve the fine-grained access control described above is to have the reference agent associate a set of credentials with each view and an access control function for all its data. When the reference agent performs an operation, the system can check the credentials against the permissions being granted by the owner.

4.6 Network Scope Restriction

Issues. The most fundamental aspect to the system context is which hosts are reachable in the network. In a fixed network such as the Internet, all nodes are permanently available (ignoring faults), but in a mobile network, connections are dropped and added with host movement. At any instant in time, a maximal system context can be conceptualized as all transitively connected hosts and the links that connect them. A critical decision in limiting the view is the specification of a subnet around the reference host that contains only a portion of the maximal context. An example of such a neighborhood might be all hosts within five miles or within three network hops. While properties of this type can be easily expressed and visualized, computing them in a real network and presenting this computed context to the application programmer may be either unnecessary or impossible. One complication that arises immediately is in the complexity and network overhead required to calculate a context based only on the physical distance from the reference agent. Consider a C-shaped network as shown in Figure 3, where the distance between the two endpoints is less than the threshold defined by the agent, but greater than

the wireless communication range between the agents. Because of the transitive connectivity, communication is available, however, there is no limit on the number of hops between these nodes.

The volatility of ad hoc networks, the complexities of managing sudden, or unannounced disconnections, the unpredictability of reconnection, and host mobility make any system development task a major challenge. In addition, the heterogeneous ad hoc network involves coordinating devices that are all different and all provide different services to the applications running on them. The neighborhood computation and maintenance must account, and perhaps take advantage of these unique conditions.

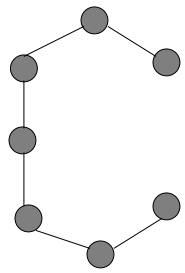

Figure 3. A C-Shaped network.

Preliminary Ideas. One possible direction to specify and calculate this restriction of the network would allow the application to specify an abstract metric over properties of the network that calculates a logical distance from the reference host to any other host in the network. By bounding allowable distances, the application can restrict the set of hosts belonging to the neighborhood. To address the problem explained via the C-shaped network above, this metric will have to be required to be strictly increasing as the number of hops from the reference host increases. Tentative results show that treating the physical network as a logical graph and defining the metric over the graph may allow computation of sophisticated metrics. Because the reference agent determines which properties of the network contribute to its neighborhood definition, allowances must be made that consider the heterogeneity of the network as described above.

One can possibly deal with the unique ad hoc network challenges described above by expecting less (see the prior discussion on consistency) or by increasing the apparent level of stability. One way to accomplish the latter may be to draw a distinction between the instantaneous physical network configuration and a logical configuration recognized by the application environment (call it a group) before even attempting to evaluate a context specification. In such a case, the members of the network delivered to the context maintenance algorithm are not necessarily *all* of the hosts in the network; instead, the algorithm is presented with some subset of the network hosts, based on some underlying group membership protocol. A previously explored idea adopts a group merging and partitioning policy based on relative distances and velocities among mobile hosts. For instance, a connection may be considered present only if the distance between two hosts is below a certain threshold, called the *safe distance*. The safe distance may be selected to allow any communications in progress to be completed before it is possible for the hosts to move out of each other's range, given appropriate assumptions or measurements of relative speeds, communication delays, and processing times. Conservative policies, such as safe distance, may help mask unannounced disconnections and will receive considerable attention as part of this proposed research. The use of location and velocity in low level protocols opens new ways of designing protocols for ad hoc networks.

5 Research Plan

The overall goal of this proposal is to deliver a novel model and middleware to facilitate coordination in large-scale ad hoc mobile environments. This first involves an exploration of the technical issues involved in the undertaking, which were discussed in detail in Section 4. The research will proceed with the careful formulation of the model, including formal semantics of view definitions and operations. A complete design will follow that will highlight the needs of the model from the underlying algorithms, playing into the development and implementation of these algorithms, specifically the ability to restrict the network neighborhood. A subsequent implementation will allow the development of applications on top of the middleware, the performance of which will feed back into the design and implementation of the model and middleware. In parallel with this implementation phase, extensions will be developed, including the aforementioned ability to attach behaviors to views and to weaken the semantics of the operations. This work may eventually lead to a formalization of context-awareness which will allow one to reason about the correctness of the model presented here.

5.1 Criteria for Success

First, this work will introduce a novel model of coordination for ad hoc networks. This model deviates from its predecessors by offering asymmetric interactions that allow more flexible interactions that directly address the needs of networks with large numbers of heterogeneous agents. Success will ultimately be measured by the ability

of this model and the resulting middleware to ease program development in the mobile ad hoc environment. A successful system will allow developers unfamiliar with the complexities of ad hoc networks but instead familiar with the coordination language introduced here to develop applications tailored to their needs. Moreover, the use of this middleware will allow this development process to occur more rapidly and with fewer errors. Additional success will be drawn from the expressiveness of the resulting model, measured by its ability to lend itself to a variety of application domains for a variety of tasks. Finally, the development of a formal model of context-awareness will allow reasoning about the correctness of this and other systems and will aid in the development of novel constructs tailored to the context-aware environment.

6 Conclusions

This proposed research is based on the premise that context-aware computing has the potential to reduce context management in mobile settings to a simple declarative specification. This research will entail redefining the notion of context by extending its scope in a manner that is amenable to the proposed declarative specification style and that provides a high level of flexibility, offers a facility for including general facets of the environment, and accommodates varying levels of consistency. One of the end products of this research will be a middleware for ad hoc mobile computing that embodies this new perspective of context-awareness and satisfies the requirements laid out previously in a cost-effective manner. In summary, the research strategy entails a careful crafting of high-level coordination constructs for context-aware computing and new mobile algorithms needed to implement them.

References

[1] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks*, 3:421–433, 1997.

[2] S. Bae, S.-J. Lee, W. Su, and M. Gerla. The design, implementation, and performance evaluation of the On-Demand Multicast Routing Protocol in multihop wireless networks. *IEEE Network, Special Issue on Multicasting Empowering the Next Generation Internet*, 14(1):70–77, January/February 2000.

[3] A. Bakre and B. R. Badrinath. Reworking the RPC paradigm for mobile clients. *ACM-Baltzer Journal on Mobile Networks and Applications (Special Issue on Mobile Computing - System Services)*, 1:371–385, 1996.

[4] J. Broch, D. B. Johnson, and D. A. Maltz. The dynamic source routing protocol for mobile ad hoc networks. Internet Draft, March 1998. IETF Mobile Ad Hoc Networking Working Group.

[5] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the ACM/IEEE MobiCom*, pages 85–97, October 1998.

[6] G. Byrd, F. Gong, C. Sargor, and T. Smith. Yalta: A secure collaborative space for dynamic coalitions. In *IEEE 2^{nd} SMC Information Assurance Workshop*, 2001.

[7] G. Cabri, L. Leonardi, and F. Zambonelli. Engineering mobile agent applications via context- dependent coordination. *IEEE Transactions on Software Engineering*, 28(11):1040–1056, 2002.

[8] L. Cardelli and A. Gordon. Mobile ambients. *Theoretical Computer Science, Special Issue on Coordination*, 240(1):177–213, 2000.

[9] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad-hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):2580–2592, August 1999.

[10] C. Cheng, R. Riley, and S. Kumar. A loop-free extended Bellman-Ford routing protocol without bouncing effect. In *Proceedings of the ACM SIGCOMM*, pages 224–236, 1989.

[11] K. Cheverst, N. Davies, and A. Friday. Services to support consistency in mobile collaborative applications. In *The 3^{rd} International Workshop on Services in Distributed Network Environments*, 1996.

[12] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Experiences of developing and deploying a context-aware tourist guide: The GUIDE project. In *Proceedings of MobiCom*, pages 20–31. ACM Press, 2000.

[13] C. Chiang and M. Gerla. Routing and multicast in multihop, mobile wireless networks. In *Proceedings of IEEE International Conference on Universal Personal Communications*, pages 546–551, October 1997.

[14] C. Chiang, M. Gerla, and L. Zhang. Adaptive shared tree multicast in mobile wireless networks. In *Proceedings of GLOBECOM '98*, pages 1817–1822, November 1998.

[15] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850, 2001.

[16] G. Cugola and G. P. Picco. PeerWare: Core middleware support for Peer to Peer and mobile systems. Technical report, Politecnico di Milano, 2001.

[17] R. DeNicola, G. Ferrari, and R. Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.
[18] A. K. Dey and G. D. Abowd. Cybreminder: A context-aware system for supporting reminders. In *Proceedings of the $w^n d$ International Symposium on Handheld and Ubiquitous Computing*, pages 172–186, 2000.
[19] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human Computer Interaction*, 16(2–4):97–166, 2001.
[20] K. Edwards. *Core JINI*. Prentice Hall, 1999.
[21] W. Emmerich. *Engineering Distributed Objects*. John Wiley and Sons, Ltd., 2000.
[22] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, May 1998.
[23] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.
[24] R. Gray, D. Kotz, G. Cybenko, and D. Rus. D'Agents: Security in a multiple-language, mobile-agent system. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 154–187. Springer-Verlag, 1998.
[25] S. Gupta and P. Srimani. An adaptive protocol for reliable multicast in mobile multi-hop radio networks. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 111–122, 1999.
[26] A. Harter and A. Hopper. A distributed location system for the active office. *IEEE Networks*, 8(1):62–70, 1994.
[27] O. Holder, I. Ben-Shaul, and H. Gazit. System support for dynamic layout of distributed applications. In *Proceedings of the 19^{th} International Conference on Distributed Conputing*, pages 403–411, 1999.
[28] J. Hong and J. Landay. An infrastructure approach to context-aware computing. *Human Computer Interaction*, 16(2–4), 2001.
[29] IBM. T Spaces. http://www.almaden.ibm.com/cs/TSpaces/, 2001.
[30] P. Keleher and U. Cetintemel. Consistency management in Deno. *Mobile Networks and Applications*, 5(4):299–309, December 2000.
[31] E. Madruga and J. Garcia-Luna-Aceves. Scalable multicasting: The core assisted mesh protocol. *ACM/Baltzer Mobile Networks and Applications, Special Issue on Management of Mobility*, 1999.
[32] L. B. Mummert, M. R. Ebling, and M. Satyanarayanan. Exploiting weak connectivity for mobile file access. In *Proceedings of the 15^{th} ACM Symposium on Operating System Principles*, December 1995.
[33] A. L. Murphy, G. P. Picco, and G.-C. Roman. LIME: A middleware for physical and logical mobility. In *Proceedings of the 21^{st} International Conference on Distributed Computing Systems*, pages 524–533, 2001.
[34] National Center for Supercomputing Applications, Integrated Decision Technologies Group. SAMCat: A securable active metadata catalogue. White Paper, 2002.
[35] OMG. Orb and object services task force. http://www.omg.org/homepages/orbos/, 2001.
[36] A. Omicini and F. Zambonelli. TuCSoN: A coordination model for mobile information agents. In *Proceedings of the 1^{st} International Workshop on Innovative Internet Information Systems*, pages 177–187, 1998.
[37] V. Park. and M. S. Corson. Temporally-ordered routing algorithm (TORA) version 1: functional specification. Internet Draft, August 1998. IETF Mobile Ad Hoc Networking Working Group.
[38] 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, October 1994.
[39] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.
[40] G. P. Picco. μCode: A lightweight and flexible mobile code toolkit. In *Proceedings of the 2^{nd} International Workshop on Mobile Agents*, volume 1477 of *Lecture Notes on Computer Science*, pages 160–171. Springer-Verlag, September 1998.
[41] B. Rhodes. Margin notes: Building a contextually aware associative memory. In *Proceedings of the International Conference on Intelligent User Interfaces*, 2001.
[42] E. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.
[43] R. Ruggaber and J. Seitz. A transparent network handover for nomadic CORBA users. In *Proceedings of the The 21st International Conference on Distributed Computing Systems (ICDCS-21)*, Mesa, AZ, USA, April 2001.
[44] N. Ryan, J. Pascoe, and D. Morse. Fieldnote: A handhelod information system for the field. In *1^{st} International Workshop on TeloGeoProcessing*, 1999.
[45] T. Sander and C. Tschudin. Protecting mobile agents against malicious hosts. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 44–60. Springer-Verlag, 1998.
[46] Soma. http://www-lia.deis.unibo.it/Software/SOMA, 2001.
[47] Sun. Javaspaces. http://www.sun.com/jini/specs/jini1.1html/js-title.html, 2001.

[48] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, and G. A. Hill. Strong mobility and fine-grained resource control in NOMADS. In *Proceedings of the Second International Symposium on Agent Systems and Applications and the Fourth International Conference on Mobile Agents*, 2000.

[49] A. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[50] E. Truyen, B. Robben, B. Vanhaute, T. Coninx, and W. Joosen. Portable support for transparent thread migration in Java. In *Proceedings of the Second International Symposium on Agent Systems and Applications and the Fourth International Conference on Mobile Agents*, 2000.

[51] B. Venners. Under the hood: The architecture of aglets. *JavaWorld*, 2(4), 1997.

[52] J. Vitek, C. Bryce, and M. Oriol. Coordinating agents with secure spaces. In *Proceedings of Coordination '99*, 1999.

[53] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. *Internet Programming Languages*, 1999.

[54] R. Want et al. An overview of the PARCTab ubiquitous computing environment. *IEEE Personal Communications*, 2(6):28–33, 1995.

[55] H. Yu and A. Vahdat. Building replicated internet services using TACT: A toolkit for tunable availability and consistency tradeoffs. In *Workshop on Advanced Issues of E-Commerce and Web-based Information Systems*, pages 75–84, 2000.