Intelligent Agents for Mobile and Embedded Devices

Tim Finin, Anupam Joshi, Lalana Kagal, Olga Ratsimor, Sasikanth Avancha,
Vlad Korolev, Harry Chen, Filip Perich, and R. Scott Cost

University of Maryland Baltimore County, Baltimore MD 21250 USA

Abstract. The *pervasive computing environments* of the near future will involve the interactions, coordination and cooperation of numerous, casually accessible, and often invisible computing devices. These devices, whether carried on our person or embedded in our homes, businesses and classrooms, will connect via wireless and wired links to one another and to the global networking infrastructure. The result will be a networking milieu with a new level of openness. The localized and dynamic nature of their interactions raises many new issues that draw on and challenge the disciplines of agents, distributed systems, and security. This paper describes recent work by the UMBC Ebiquity research group which addresses some of these issues.

This research was supported in part by the NIST Advanced Technology Program, DARPA contract F30602-97-1-0215, and NSF grants CCR0070802 and IIS9875433.

1 Introduction

In the past few years, the research community has seen plenty of hype associated with wirelesss / pervasive / mobile / ubiquitous computing. Mobile Commerce (M-Commerce) in particular was declared as the “killer app” driving the wireless revolution. In what is undoubtedly testimony to the speed at which internet time moves, the last year has also seen pundits declaring, with equal certainty, that M-Commerce is either a non-starter or that it is dead. In large part, the blame for both the initial hype and the more recent disappointments must rest with the rather narrow vision of m-commerce that some segments of the industry were promoting. In this vision, cell phones (or wirelessly connected PDAs) became mobile storefronts for e-tailers – essentially an incremental change in the present e-tailing idea. We are all familiar with the ads of people buying products and services from their cell phones from the beach. The drawbacks of this idea are not hard to identify, as an increasing number of recent critical commentaries show. This approach essentially treats palmtop devices as consumers (or clients) of goods or information. The information or goods come from servers on the wired side.

This approach (typically based on a client–proxy–server model) has been developed in academia over the last five or six years in contexts such as web access from mobile platforms (for instance [26, 30, 6, 33, 37, 3, 25]) or transaction support for database access [13]. Variants of this approach are now emerging from several commercial companies in the form of systems that allow phone based "microbrowsers" or PDAs to access domain specific internet content (headline news, stocks, sports etc.) from designated portals. The WAP consortium (http://www.wap.com/) is leading efforts among telecommunication companies and service providers to evolve a standard mechanism and markup language to support this. There have also been efforts, mostly from startups and e-tailers, to allow people to buy goods using the phone microbrowsers or PDAs. In some sense, one can think of this as a *supermarket approach*, where a few identified service providers exist and the traffic in services is one–way.

Viewed in a broader perspective, M-Commerce in particular, and M-Services in general, have yet to be fully articulated or explored, especially in the context of emerging mobile ad-hoc networks. Staying with the M-Commerce idea, consider a move away from the prevailing mobile store front vision. In the future, instead of just interacting with the "buy–it–yourself" web storefronts, consumers will be able to interact with service providers in a personalized way via automated service-oriented eMarket models (e.g. [22]). The selling of tangible goods will simply be one such service. Other services would include data, information, software components or even processing time/storage on networked machines. There will be no explicit clients and servers – but peers which can be both consumers and providers of different services. M-Commerce will thus be transformed into what we refer to as *Me-Commerce*, or more generally, *Me-Services*.

We have been working on a number of projects utilizing this alternative *bazaar approach* that involves the cooperation of autonomous ("active"), self-describing ("articulate"), highly interactive ("social"), and adaptive ("intelligent") components which are located in "vicinity" of one another. Such hardware and software components will automatically become aware of each other, establish basic (wireless) communication, exchange information about their basic capabilities (e.g. services they can offer) and requirements (e.g. payments they need), discover and exchange APIs, and learn to cooperate effectively to accomplish their individual and collective goals.

This idea of "ad-hoc" teams of entities that are dynamically formed to pursue individual and collective goals can be used to create the software infrastructure needed by the next generation of mobile applications. These will use the emerging third and fourth generation broadband wireless systems, as well as short range narrowband systems such as Bluetooth. Heretofore, the software component of mobile computing has lagged behind its hardware (communication, computing and networking) aspects. Much of the research in the software area is often limited to allowing applications built for the wired world (web, databases etc) to run in the wireless domain using proxy based approaches. Our work has sought to move beyond this and provide a framework which uses the power of mobility and ad-hoc wireless connectivity to enable novel applications. The results will point us toward a physical environment in which devices all around us and even on our body are in constant communication and organize themselves to cooperate to do our bidding.

1.1 A Fanciful Scenario

It is 5:30 in the evening, and Jane's panel meeting at NSF in Arlington is just ending. As she steps out of the building to walk the two blocks to her hotel, she decides that after resting for a bit, she'll get ready and have dinner in one of the nearby restaurants. She tells her palmtop of this decision, and asks it to find nearby restaurants that serve cuisines that she would like to try but can't find in her home town. The palmtop goes into discovery mode and connects to a nearby broker provided by the Arlington Restauranters Association. It sends it Jane's cuisine preferences and price ranges, and asks for a recommendation for a restaurant where she can eat in about 45 minutes from now. The broker has some static information about its local restaurants such as location and menus. Managers in the restaurants are also feeding it dynamic information such as wait times or any discounts/specials that they may have, based on their current situations. Based on both the static and dynamic information, the broker comes up with a list of possibilities. Meanwhile, as Jane is walking back, her PDA asks the PDAs of others in the area for their opinions of good local restaurants and stores them. When Jane is ready to head out, the Palm PDA pulls the recommendations from the broker and presents them to Jane. Her choices include Vietnamese, Malaysian, Mongolian and Cambodian restaurants, since these cuisines are similar to the Chinese foods she likes. She selects the Vietnamese restaurant, since it is offering a 20% off coupon and had a couple of good mentions in the information her PDA had gathered from other people. Her PDA communicates this choice to the broker, asks that a reservation be made. It then discovers the local map server that the city of Arlington runs, and gets directions from Jane's hotel to her restaurant.

Other scenarios can be drawn in and around meeting rooms, shopping malls, airport terminals, train stations, and highway exits. A commuter train passenger getting off her destination station may need to send an urgent fax but her PDA lacks dial-up capability. Or perhaps, during the trip, she may have received an email attachment that contains a new audio or graphics format that requires a player she does not have on her PDA. As the train starts to slow down entering the station, the passenger starts up an agent that can seek and retrieve the fax or the player software services as she walks out of the station. Another possible scenario that could arise in airport terminals and gateways is one in which arriving and departing passengers may seek each other to exchange leftover currency directly short of bank's buy/sell exchange rates. Similarly, crossing tourists may exchange or sell goodies such as e-coupons, unused museum tickets, and personal tips on their way out of a country they have visited.

1.2 Background

The basic assumption behind our work is that at the level of computing and networking hardware, we will see dramatic changes in the next few years. Computing will become pervasive – a large number of devices (e.g. phones, PDAs, household appliances) will become computationally enabled, and micro/nano sensors (the so called smart dust) will be widely embedded in most engineered artifacts. All of these devices will be (wirelessly) networked. More specifically, we assume the emergence of (i) palmtop and wearable/embeddable computers, (ii) Bluetooth like systems which will provide short range, moderate bandwidth connections at extremely low costs, and (iii) widely deployed, easily accessible wireless LANs and satellite WANs. We assume that these will be a mixture of traditional low bandwidth systems and the next generation high speed ones. These developments will lead to wireless networks that will scale all the way from ad hoc body area networks to satellite WANs, and link together supercomputers, "palmstations" and embedded sensors and controllers. There is ongoing research in industry and academia in creating the hardware and low level networking protocols that are needed to realize this vision. Some recent efforts have also started creating smart spaces and integrated sensor networks and addressing the data management challenges that will need to be solved.

The scenario we have described earlier serves to illustrate the technical challenges that we and others are trying to address. In particular, as computing becomes pervasive, an increasing number of entities will be both sources and consumers of data and information. This is a significant change from the present, where mobile devices essentially remain consumers of information, and the research challenge has been to get them the right information in a format suited to the bandwidth and resource constraints of the device [26]. In the future we envision, besides the traditional Mobile Support Station based wireless access, many of the devices will communicate using Mobile Ad-hoc Networks (MANET) [21] formed by Bluetooth type devices. Further, most devices will have (and use) multiple wireless network interfaces (Bluetooth, LAN, cellular etc.) [34]. Besides obtaining information from "canonical and centralized" sources (such as a yellow pages server for restaurants), a device may obtain information (recommendation about restaurants) or service offers (a discount coupon for dinner) from other devices around it – its present *dynamic community*. The scenario demands that the system have some sense of vicinity. The entities in the system must be able to describe themselves as well as discover others and figure how (and whether) to interoperate with them, both at syntactic and semantic levels. Finally, the entities should be able to communicate abstract ideas, (e.g. goals such as what information is it looking for) and be able to negotiate with others for services (I want to know what the traffic conditions are at I 95 Springfield Interchange, in return I can provide traffic conditions in downtown DC.). Of course, the devices participating will be resource constrained and heterogeneous, which poses further problems.

Note that the challenges here go over and beyond those found in heterogeneous and distributed data management, and are in ways more subtle than simply handling reduced bandwidth or disconnection, which of course remain important issues. To express this in traditional data management parlance [12], we could say that unlike heterogeneous data access, where “schemas” and “catalogs” at least are known in advance, we are talking of a situation where both are highly variable and not known up front. Thus a “query” will return results dependent on where it originates, what location it refers to, and who is around at that time.

1.3 The Remainder of this Paper

In the remainder of this paper we describe on three recent efforts we have made to explore some of the issues mentioned above. We first present our work in developing the Centaurus system, a relatively low-level framework on which to build intelligent services in a mobile environment. We then describe research aimed at using a distributed trust model to provide security and access control in such environments. Finally, we describe an example application, agents2Go, which uses location awareness to provide a simple restaurant recommendation service.

2 Centaurus

The system described in this section provides an infrastructure and communication protocol for providing ‘smart’ services to these mobile devices. This flexible framework allows any medium to be used for communication between the system and the portable device, including infra-red, and Bluetooth. Using XML for information passing, gives the system a uniform and easily adaptable interface. We explain our trade-offs in implementation and through experiments we show that the design is feasible and that it indeed provides a flexible structure for providing services. Centaurus provides a uniform infrastructure for heterogeneous services, both hardware and software services, to be made available to the users everywhere where they are needed.

Our goal is to provide an infrastructure and communication protocol for wireless services, that minimizes the load on the portable device. While within a confined space, the Client can access the services provided by the nearest Centaurus System (CS) via some short-range communication. The CS is responsible for maintaining a list of services available, and executing them on behalf of any Client that requests them. This minimizes the resource consumption on the Client and also avoids having the services installed on each Client that wishes to use them, which is a blessing for most resource-poor mobile clients.

We also expect all Services to communicate via XML or XML-based languages such as RDF and DAML which are suitable for defining ontologies and describing properties and interfaces of Services. As this is already being widely used, we think that it will help in integrating Centaurus with already existing systems. The information flowing in the system is strictly in the form of CCML (Centaurus Capability Markup Language) which is built on top of XML.

Additionally, we address the security concerns of *SmartSpace* by incorporating a simplified Public Key Infrastructure that provides for authentication, non-repudiation, anti-playback, and access control. We employ smartcards and other media as secure containers for digital certificates to allow clients to securely act as portals to services through heterogeneous networks.

To verify the feasibility of our infrastructure, we have used InfraRed (IR), Bluetooth and 802.11b wireless technologies to enable communication between Clients and Service Managers

In the last few years, a number of technologies have emerged that deal with 'Smart' Homes and Offices. Among them are the Berkeley Ninja Project [10], the Portolano project [15] from the University of Washington, Stanford's Interactive Workspaces Project [19], and Berkeley's Document-based Framework for Internet Application Control [23]. In the remainder of this section we will present the design and modeling issues of our approach and touch on ongoing work. Complete details on the communication protocols, implementation, experimental results and comparisons to other systems are available in [28, 29, 7, 43].

2.1 Design

Our system is designed to control access to services within a *SmartSpace*. Centaurus2 is a second generation framework that enhances the original design of Project Centaurus. The framework allows clients (users and services) to move, attach, detach, and re-attach at any point within the system. A 'SmartRoom' is equipped with a Centaurus Communication Manager, which continuously broadcasts, through some medium, a client application. A person with a portable device who enters the room for the first time is given the option to install the software. Once the application is installed, it continuously reads the updated list of services. The person is able to choose a service, select a function, fill in the related options and execute the function. These services may be provided by Centaurus systems other than the one the portable device is connected to. Additionally, we provide a bridging mechanism so that a user in one domain can reach a service in another domain provided that the user has the appropriate permissions.

Centaurus Communication Protocol. The Centaurus Communication Protocol (CCOMM) is used to communicate with mobile clients and services and consists of two levels. CCOMM Level1 is used as a glue between some existing communication architecture such as IrDA stack, Bluetooth, or TCP/IP and the generic Level2 protocol. The Centaurus Level1 protocol handles connection and disconnection issues, identification and authentication of the clients and interaction with architecture specific protocols such as IrLAP and IrLMP or Bluetooth. The CCOMM Level2 protocol handles transmission of the XML messages, time synchronization, message fragmentation and re-assembly. The Centaurus Level2 protocol is designed to be insensitive to disconnections, handle multiple clients, provide minimal turnaround times and be easily portable.

In fact in the current implementation all communication managers and client communication modules use the exactly the same code base.

Components. There are five functional components within the Centaurus2 system. *The Centuarus2 Certificate Authority* is responsible for generating x.509 version 3 digital certificates for each entity in the Centaurus2 system and for responding to certificate validation queries from Service Managers. The *Centaurus2 Capability Managers* maintains a database of the group membership of Centaurus2 entities and answer requests for group membership. Next is the *Communications Managers*, which provides a communication gateway between a Client device and a Service Manager. Its sole purpose is to abstract and translate communications protocols. The fourth component is the *Service Manager*, which brokers requests between registered user-Clients and service-Clients. Finally, users and services are treated equally as *Clients*. Access rights of others to a particular client entity are exclusively determined by the entity itself; and set when it registers with a Service Manager. This equality of users and services is a minor variation from Project Centaurus original design; however, it allows a Client to access services while at the same time providing some its own services to others.

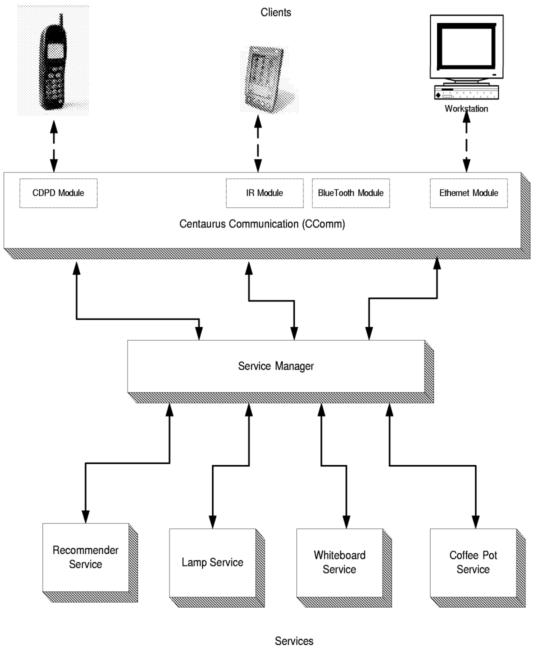

Fig. 1. Centaurus Components

Centaurus2 Certificate Authority. The Centaurus2 Certificate Authority is used to produce x.509 version 3 digital certificates [17]. In Centaurus2 certificate request and issuance is ancillary to the system. When a certificate request from a Centaurus2 entity is filled, the entity receives its requested x509 v3 certificate signed by the Certificate Authority and the Certificate Authority's self signed certificate, which is subsequently used to validate other entities' certificates. These certificates are stored and protected on a client's smartcard.

Certificate generation and signing is typically a one time occurrence for any entity within the Centaurus2 System. In a typical PKI the Certificate Authority makes its registrant's public certificates available in an on-line repository and provides an on-line Certificate Revocation List (CRL) where inclusion indicates that a given certificate is, for one of many possible reasons, invalid. As previously stated, Centaurus2 uses a simplified PKI. In Centaurus2 each entity presents its certificate to its Capability Manager when it registers. Rather than use a CRL to signal a problem with an entity, the entity's entry in the Capability Manager is blocked, consequently preventing all access by that entity to the Centaurus2 system. This precludes the necessity of maintaining a CRL, which must be signed by the Certificate Authority each time it is modified.

A Centaurus2 Service Manager verifies the authenticity of its stored copy of the Certificate Authority's certificate by sending the Certificate Authority a validation query. The Certificate Authority replies to the query with $E_{privatekey}((verifier))$. In Centaurus2 the Certificate Authority's certificate SHA-1 message digest is used as the verifier. To verify the validity of its copy of the Certificate Authority's certificate the Service Manager tests if $(CertificateAuthority'scertificateSHA-1messagedigest) = D_{publickey}(E_{privatekey}((verifier)))$. When the test passes it assumes that the copy of Certificate Authority's private key is valid and any object signed by that key is also valid.

Our Lightweight PKI, in contrast to the traditional PKI, does not maintain a CRL, does not transmit its key via the network, and does not distribute user's certificates or public keys. Rather, the Service Manager verifies its copy of the Certificate Authority's certificate with the CA as described above. In turn the Service Manager will ensure that all certificates that it receives from clients have been signed by the Certificate Authority.

Centaurus2 Capability Manager. The Centaurus2 Capability Manager is responsible for maintaining and communicating group membership(s) of all entities in the Centaurus2 system. Entities include Service Managers and Clients (users and services). Group membership may be as general as "*umbc.edu*" (meaning that only entities in the group umbc.edu are allowed access), more restrictive as "*cs.umbc.edu*", or even so granular as only the named Client "*fperic1*", which implies that only the named Client is allowed to access a particular service.

When the Centaurus2 Capability Manager is initialized it reads its x.509 v3 digital certificate and its PKCS#11 [38] wrapped private key from a secure file and stores it into local memory. It also reads and indexes the capability file containing the group membership of all entities within the system, as well as storing the time stamp of the capability file.

When a Service Manager's group membership request is received, the Centaurus2 Capability Manager compares the current time stamp on the capability file with the time stamp of the last file read, if they are not equal it re-reads the capability file. This feature allows for a dynamic administration, to include rights revocation, of the capability file.

In response to a group membership request, the Centaurus2 Capability Manager sends a message containing the subject's group memberships. The response is digitally signed with the Centaurus2 Capability Manager's private key.

We note that there could be multiple Capability Managers where one Capability Manager serves a cluster of Service Managers. In the event of multiple Capability Managers each instance will replicate the capability database. During initialization, each Service Manager learns the location of its Capability Manager from its configuration file and communicates with that Capability Manager directly. At the first communication exchange between the Service Manager and the Capability Manager, the Service Manager requests and validates the Capability Manager's certificate, which it receives encoded in a signed CCML message.

Centaurus2 Service Manager. The Service Manager is responsible for processing Client Registration/De-Registration requests, responding to registered Client requests for a listing of available services, for brokering Subscribe/Un-Subscribe and Command requests from user-Clients to service-Clients, and for sending Service Updates to all subscribed users whenever the state of a particular service is modified.

Service Managers are arranged in tree-like structure and form the core of the Centaurus2 system. Service Managers are identified by their "locations", or handles. With the exception of Group Membership requests to the Capability Manager and certificate validation requests to the Certificate Authority all messages are sent and routed through the hierarchy of Service Managers using the handle to determine where to forward each message.

All Clients (users and services) rely upon the Service Manager to which they are registered to enforce security, access control, and to broker requests for services. Consequently, each Client is only concerned with the trust relationship with its immediately connected Service Manager. In turn, Service Managers establish trust relationships with each other. Consequently, trust between Clients is transitive through the Service Managers.

Generally in a PKI system, certificates are made available in an on-line repository. Consequently, when a user needs an entity's digital certificate it is requested from such a repository and assumed to be valid once it is received and has verified the certificate signing chain along the entire path to the top level signature authority. In the general PKI implementation certificate repositories and CRLs have a high degree of administrative overhead. This overhead and the accompanying network traffic imposed by certificate acquisition and the signature verification is mitigated in Centaurus2 by its simplified PKI framework. As previously stated, each Centaurus2 entity possess its certificate and presents that certificate upon registration. All entity's attributes, to include the validity of its certificate, are validated through a single query to the Capability Manager.

In addition, the authenticity of the presented certificate is verified by ensuring that the certificate was signed by the Certificate Authority.

The Service Manager maintains a database of Client profiles for all entities registered with it. Information contained in the profile includes the Client's certificate (or the certificate of the Client's Service Manager if the Client did not initially register with the Service Manager), group memberships, location (the Service Manager to which it is immediately connected), name, and permissible access groups.

Client. Client registration and subsequent access to services occurs on an ad hoc basis. All Clients must register with a Service Manager prior to accessing any services or making its services available. At registration, a Client will, in addition to sending its digital certificate, transmit a list of group memberships required for other Clients to access its services. The Service Manager will store the Client's certificate, list of access memberships, and the list of group memberships (acquired from the Capability Manager) for the Client in the Service Manager's user profile database. Generally, a user-Client will send an empty access list indicating that no other Client may be granted access to it, whereas a service-Client will include a list of groups where membership in at least one of the listed groups is required for access to that Client's service.

Once a Client has successfully registered, it is given an interface to all services to which it has rights to and which are also registered to the same Service Manager. This includes an interface to all other Service Managers that the Client's Service Manager is aware of.

Centaurus Capability Markup Language. The CCML is divided into 'system' , 'data', 'addons', 'interfaces', and 'info'. The 'system' portion contains the header information, the id, timestamp, origin, etc. There are two variables, 'update' or 'command'. An 'update' variable is used to inform other Centaurus components about status updates of Services and Clients, whereas the 'command' is only used by Clients to send a command to a certain Service. The system also contains the listening section for a Service or Client. It specifies all the Services that a Service or Client is interested in. Using the 'addons' section, we can add a related Service to another Service, for example, add an Alarm Clock Service to a Lamp-Control Service. We are not currently using this section. All information regarding the variables and their types are contained in the 'data' section. The CCML for a Client always has one or more 'actions' in its data section that a Service Manager can invoke on it. This is used by the SM to change the state of the device.

Two actions can be conveyed in the CCML:

- *AddService*: When this action is set, the Client adds the value of this variable to its InterestList; i.e. the list of services that it is interested in.
- *RemoveService*: This is set by the Service Manager, if the Service that the Client is interested in, is no longer available. It causes the Client to stop listening or using the Service and remove the Service from its InterestList.

The ‘interface’ section contains information about the interfaces that the object (Service/Client) implements. Other details like the description, and icon for representation are in the ‘info’ section.

2.2 Conclusion

We have successfully developed a highly flexible and adaptive infrastructure for providing secure access to services within the *SmartSpace* environment. We believe that by providing a uniform infrastructure using XML-encoded data exchange we have shown that it is appropriate and effective for deploying services in such environments.

We are also working on a Recommender Service. Instead of returning a list of all possible services that are available to a Client, this service recommends a list of services that might be in the interest of the Client based on the existing environment context. For example, the system returns a coffee-maker control service during the morning to the user, and in the evening it returns a light control service to the user. It may also notice that the user generally wants to listen to to the same list of songs and provide the list as soon as the user steps into the room.

3 Distributed Trust Management in Pervasive Environments

Traditionally, security for stand alone computers and small networks was handled by physical security and logging into computers and domains. With open networks like the *Internet* and pervasive environments, issues concerning security and trust become crucial. There is no longer the physical aspect of security due to the distributed nature of the networks and the concept of user authentication to a domain is not possible. Imagine a scenario where a user, with a portable device, walking through a building, switches on the lights in the corridor and lowers the temperature of the room that he/she enters. This is an example of pervasive/ubiquitous environments that will soon be a reality. In these *ubiquitous computing* environments users expect to access resources and services anytime and anywhere, leading to serious security risks and problems with access control as these resources can now be accessed by almost anyone with a mobile device. Adding security to such open models is extremely difficult with problems at many levels. We can not assume an architecture with a central authority and access control is required for foreign users. The portable hand-held and embedded devices involved have severe limitations in their processing capabilities, memory capacities, software support and bandwidth characteristics. Moreover, there is currently a great deal of heterogeneity in the hardware and software environments and this is likely to continue for the foreseeable future. Finally, in such an open, heterogeneous, distributed environment there is a great likelihood that inconsistent interpretations will be made of the security information in different domains.

We encountered several problems with security for Centaurus. Firstly, it is not possible to have a central authority for a single building, or even a group of rooms. It is also not sufficient to authenticate users because most users are foreign to the system, i.e. they are not known. So there is no means of providing access control. Consider a *Centaurus Smartroom* in an office, equipped with an MP3 player, fax machine, several lights, a coffee maker and a printer. If a user, John, walks into a room, how does the room decide which services John has the right to access. Just authenticating John's certificate gives no information about his access rights because John is an unknown user. Unless the expected users along with their access rights are known in advance, dividing authorization into authentication and access control is not possible. Assume John does not work in the office, but in one of its partner firms. How will the system decide whether to allow him to use certain services?

We suggest enhancing security by the addition of trust, which is similar to the way security is handled in human societies. Some authorized person in the office can *delegate* the use of the services in the room to John, for the period during which he is in the office. Trust management can be viewed as verifying if the credentials of the requester fulfill the policy of the requested service [35, 4]. We propose a lightweight solution for trust management that is applicable for the *Internet*, which we are tailoring for pervasive computing environments.

3.1 What is Distributed Trust Management ?

The distributed trust management approach involves articulating policies for authentication, access control and delegation, assigning credentials to agents, allowing entities to delegate or defer their rights to third parties and providing access control by checking if the initiators credentials fulfill the policies. If an agent has credentials allowing it to access a certain service, the agent is said to have the *right* to access the service. If an agent defers a right it has to another agent, it is called a *delegation*, the former is called delegator and the latter delegatee. Delegations are very important to trust management as they help disperse trust and provide a very flexible mechanism for evaluating access rights. They allow distributed systems to be more scalable as the process of generating and evaluating access rights is no longer limited to certain central administrative entities [5].

Blaze, who coined the term *distributed trust management*, tries to solve the trust problem by binding public keys to access control without authentication [35, 4]. His PolicyMaker, given a *policy*, answers queries about trust. Though powerful, the policy definition is complicated and not easy to understand for non-programmers who are probably going to develop the policy. Simple Public Key Infrastructure (SPKI) was the first proposed standard for distributed trust management [14]. This solution, though simple and elegant, only included a rudimentary notion of delegation. Pretty Good Privacy or PGP [45] was developed to enable the sending of secure email without a secure key exchange or a central authority. In PGP, a keyholder (an individual associated with a public/private key pair) learns about the public keys of others through introductions from trusted friends. Whether or not a user accepts the information about new keys depends on the number of introduces and the degree to which they are trusted (quantized in PGP to three levels: fully trusted, partly trusted and untrusted). Some of the main problems with PGP are with key distribution and management. The Use-Condition Centered Approach [24] uses certificates for use-conditions that are created by those responsible for the resources. This can only be used when the resource is simple enough to be described by use-conditions, but in large systems there could be many types of access like read, write, execute etc. Delegation logics [32, 20] is similar to our approach, however it is not able to capture adequately the constraints associated with rights and delegations.

3.2 Previous Work

We have implemented a Distributed Trust Mechanism for a Supply Chain Management (SCM) system for the CIIMPLEX EECOMS project [42, 9]. An SCM system consists of groups of agents that are either vendors or clients. These agents need to access resources in each others domains. For example, a software consultant may need to access the database of its client. Each group of agents that are part of the same company form a policy domain, and follow the same security policy. The policy in each domain is enforced by special agents called *security agents*. Agents are identified by X.509 [18] authentication certificates and all communication is via signed messages. Security agents are able to reason about these signed messages and policies to provide authorization.

Our system sets up authorization and delegation rules, so that the information in the SCM may be accessed only by authorized agents. Security agents provide authentication and authorization within a particular domain, and are trusted within the company and by the company's buyers and sellers. They also represent the company in some sense as they interpret the company policy. This policy describes certain rules for rights, delegation and for reasoning about them.

In order to allow the buyer's employees to access certain information within its company, the security agent of the seller gives the security agent of the buyer the permission to access that information, and the ability to delegate this right. To propagate this trust within its own company, the security agent of the seller delegates this right to some of its employees based on the policy. Depending on the previous delegations, the employees can further delegate this right to other employees, forming a chain of delegation from the buyer's security agent to the seller's security agent to the seller's employees. If at any point a delegation fails or is revoked the access cannot go through. The same holds if the situation is reversed and the supplier gives the buyer access to some of its resources. Delegation chains should always trace back to a security agent to be valid. Security agents are responsible for all accesses originating from its company and act as gateways. All access to information outside the company must go through a security agent. This agent will authenticate the requester, check the delegation chain and verify that the requester has the right to access the requested information.

The security agent creates an authorization certificate for the requesting agent, that the requesting agent can use for access.

This framework led us to view trust management as a very effective method for resolving several issues related to security in distributed systems.

3.3 Trust Architecture

Our architecture supports authentication through a PKI infrastructure based on X.509v3 certificates. After entities are authenticated, they are assigned roles either statically, based on a role list or dynamically, based on a set of rules that describe the credentials an entity must possess, to be in a certain role. In our previous work with supply chain management [27], trusted third parties were responsible for delegating abilities, and authorizations were deemed valid, if they were supported by these trusted entities. However in our current work, trusted third parties are not responsible for delegating abilities. Any entity can delegate any right that it has the ability to delegate. However all delegations are sent to and stored by Service Managers, that are trusted to reason about the policy and delegations and provide access control within a *SmartSpace*. The Service Manager still employs trust based mechanisms to decide access rights of an entity by reasoning over the global policy of the organization and its own local policy. The global policy will include the roles in the organization and role based access rights. Vigil does not use delegation certificates because most mobile devices do not have the resources to store so many certificates. This semi-centralized approach makes revocations rather straightforward. The appropriate Service Manager needs to be informed about the revocation and it is reflected immediately. This approach works very well for *SmartSpaces* because of the localized nature of these spaces.

Rights or privileges can be given to trusted agents who are responsible for the actions of the agents to whom they subsequently delegate the privileges. These rights are generally associated with the role of the agent. So the agents will only delegate to agents that they trust. This forms a delegation chain. If any agent along this chain fails to meet the requirements associated with a delegated right, the chain is broken and all agents following the failure are not permitted to perform the action associated with the right [27].

Agents can make requests, either for certain actions or to ask for permission to perform an action, and while doing so they attach their ID certificate the request. The Service Manager authenticates the requesting agent and checks its delegations before authorizing the request. An agent is allowed to execute any action that it has the permission to execute, or if the ability has been delegated to it by an agent with the right to delegate. In our system we view 'delegation' as a permission itself. Only an agent with the right to delegate a certain action can actually delegate that action, and the ability to delegate, itself can be delegated.

We have developed a representation of trust information in Prolog, that allows flexibility in describing requests and delegations. Delegations can be constrained by specifying whether the delegatee has the permission to delegate and to whom it can re-delegate.

Consider an example of John entering a *SmartRoom*. John is an employee of one of the office's partners. He approaches one of the managers, Susan, and asks for permission to use the services in the *SmartRoom*. According to the policy, Susan has the right to delegate those rights. Susan delegates to John the right to use the lights, the coffee maker and the printer but not the fax machine. Susan sends a message to the *Centaurus Service Manager* of the office, associating John's identity certificate with the rights. When John enters the room, he sends a request to use the printer to the the Service Manager of the room. The Service Manager reads John's identity certificate but cannot locate any access rights. So it asks the Service Manager above it in the hierarchy, this continues till the request reaches the root. This Service Manager has the delegation made by Susan. The service manager sends the access right back down the chain. On receiving the access right, the SmartRoom's Service Manager authorizes John's request and John is allowed to use printer. One optimization is for the root Service Manager to send back the entire list of access rights. The SmartRoom's Service Manager creates an authorization certificate containing all the rights and returns it to John. While the authorization certificate is valid, John can access the services without the Service Manager having to re-check his rights. However once the certificate expires, the Service Manager checks with the root if the access rights are still valid and creates another certificate. By having very short periods for the certificate, the system handles *revocation* of rights easily. This scenario demonstrates the importance of trust over simple security.

3.4 Ongoing Work

We are working on integrating trust into the security infrastructure for *Centaurus*. We believe that trust will add a new dimension to pervasive computing, allowing more flexibility and control over access control.

At the same time, we are improving our trust architecture. The system is being extended to include entitlements, prohibitions and obligations and the ability to delegate them. We are not sure if these delegations can be interpreted correctly. In many contexts, an agent may want to delegate some obligation to another agent who is willing and able to assume it. If the obligation is not fulfilled, then hopefully the former agents reputation will not suffer as much as the agent which took on the obligation. Entitlements are stronger than permissions and prohibitions are negative permissions, both of which can be delegated in the same way as permissions.

Our approach is to treat delegation as a "speech act" and to associate with it appropriate conversational protocols along the lines of those used in agent communication languages [31]. In some cases it may be necessary to allow or even require the delagatee to accept or reject the delegated object. Although we typically view privileges as extending our capabilities and thus being purely beneficial it is not always the case. If we allow privileges to entail corresponding obligations when the privilege is exercised, an agent may not want to accept the delegation of the privilege. Similarly, an agent may want to reject the delegation of an obligation or a prohibition, if it is allowed to.

Another important issue with distributed networks is that of privacy. Users do not want their names and actions to be logged, so we are trying to do away with with X.509 certificates and replace them with XML signatures [44] from a trusted authority and does not include the identity of the bearer, but only a role or designation.

Our past work on distributed trust represented actions, privileges, delegations and security policy as horn clauses encoded in Prolog. In order to develop a approach that is better suited to sharing information in an open environment, we are recasting this work in DAML [11], the DARPA Agents Markup Language. DAML is built on XML and RDF and provides a description logic language for defining and using ontologies on the web. Using DAML, we are defining the basic ontologies for actions, agents, roles, privileges, obligations, security policies and other key classes and properties. In applying our framework, one must extend the initial ontology[1] by defining domain specific class of actions, roles, privileges, etc. and creating appropriate individuals.

3.5 Conclusion

According to the current paradigm of pervasive computing, we will soon be able to access information and services virtually anywhere and at any time via any device, whether it is our phones, PDAs, laptops or even watches. In environments like this, security is very important. But security by itself is insufficient and trust management is required, as normal security mechanisms like authentication and access control are unable to meet the requirements of these environments. Trust management is the development of security policies and credentials, the checking of presented credentials against the policy and the delegation of permissions. Earlier in this section, we presented the design of a trust based architecture for pervasive systems based on some of our previous work with distributed trust management. To help make the vision of ubiquitous computing a reality, we firmly believe that trust should be included in the security infrastructure for pervasive environments.

4 Agents2Go

In this section we describe the Agents2Go system that addresses some of the problems of location dependent service discovery in an M-commerce environment. Agents2Go is a distributed system that provides mobile users with the ability to obtain location dependent services and information. One of the most critical requirements for M-Commerce is the ability to discover services in a given context. Context aware computing [40, 2] is a challenging goal. It involves recognizing a users location, current focus of attention, immediate objectives and even ultimate underlying goals. Our objective in Agents2Go system is to provide location dependent service and information discovery. Our system automatically

---

[1] http://daml.umbc.edu/dt.daml

obtains a user's current geographical location in CDPD (Cellular Digital Packet Data) based systems without relying on external aids such as GPS (Global Positioning System). This location information is used to find a local provider for a requested service. For example, a user arriving at a location that he/she has never visited before should be able to find a local taxicab service.

Current mobile devices have well known inherent limitations [25] like limited power supply, smaller user interface, limited computing power, limited bandwidth and storage space. To overcome these limitations, it is necessary to develop systems that provide mobile users with high quality, accurate context relevant information. It is important that these systems be highly scalable since the demand for service searches will increase in the future.

A location dependent search utilizes a user's current geographical location to refine the search and provide access to locally available services without going though a centralized infrastructure. One of the challenges for location-based searches is determining the user's current location. Users are often uncertain, or even completely unaware, of their current geographical location making location based searching more difficult. An automated detection of the user's current location is very helpful in overcoming this issue.

Location dependent systems are naturally described and implemented as distributed systems. This improves their fault tolerance and scalability. For instance, service information can be grouped by location and managed by a server responsible for that specified geographical region. In such a decentralized scheme user requests are processed at the local server and do not burden the rest of the system. This makes the system more efficient, responsive and scalable.

4.1 Related Work

There are a number of platforms that provide multi-agent infrastructures to allow inter-agent communication and collaboration. These platforms can be used to create collaborative intelligent agent environments that could provide location based information services. One such infrastructure is the Lightweight Extensible Agent Platform (LEAP) [1]. LEAP provides a lightweight platform that is executable on small devices such as PDAs and phones. It is FIPA [16], compliant and also supports WAP and TCP/IP. The YellowStone Project [39] from Reticular Systems, Inc. also deals with location dependent services. Essentially, there are communities of software agents called agencies, which provide information services and e-commerce support for a particular geographical area. However, a participating user has to specify his current geographical location. Very recently researchers at AT&T Research Labs have described a project with similar goals [36], which also locates the user in a CDPD [41] environment using cell tower IDs.

There are several features that distinguish Agents2Go from other existing location dependent service discovery systems. Primarily, Agents2Go is a platform for supporting deployment of any type of location dependent services, and not just limited to providing location dependent information. Our system automatically discovers user's location and utilizes this information to refine its searches to

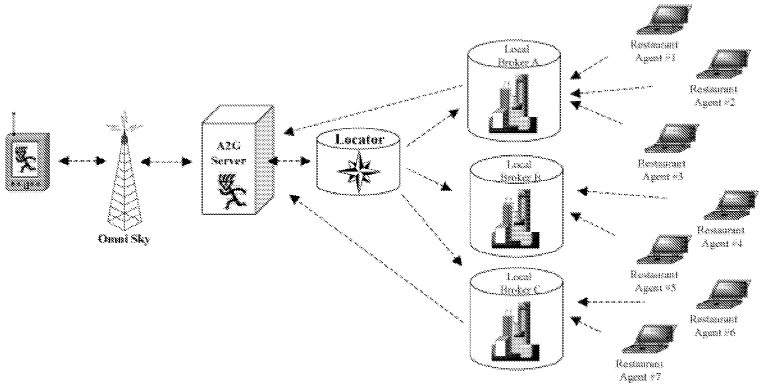

Fig. 2. Agents2Go is a simple prototype application exploring which makes use of location aware services.

provide the most relevant information to the user. In addition, the Agents2Go system allows service providers to actively participate in the system. Service providers routinely supply the system with dynamic information updates. This improves the quality of information or services presented to the user and ensures that the service providers are able to send their latest promotions/updates to their users.

4.2 Components of the Agents2Go System

The Agents2Go System, illustrated in Figure 2, is composed of several components: the PalmApp, the Agents2Go Server, the Locator, the Agents2Go Information Repository, the restaurant Brokers and participating Restaurant Agents.

PalmApp. The PalmApp is the end user interface to the Agents2Go System. This component runs on the user's PDA equipped with a CDPD modem. Essentially, it is a generic "form visualizer" that is independent of the system functionality. we used in-house markup tags to specify the layout and components of the form. The PalmApp captures a user request, converts it to an appropriate format, and then forwards that request to the Agents2Go Server. The PalmApp also handles the responses from the Agents2Go Server and presents them to the user. Our initial implementation was for the PalmOS platform.

Communication and Location Detection. All the messages that are exchanged between the Agents2Go Server and the PalmApp are sent using the CCOMM described in section two. Our current Agents2Go System uses CDPD Level1 Module, which is an extension of the UDP Level1 module. CDPD provides an infrastructure that allows the transmission of data over idle capacity of already existing cellular voice networks. Cellular networks consist of cell towers with a unique ID and a specific geographical cell over which it provides services. Our system employs these tower IDs to identify a user location. We have developed a library that allows us to control and interact with Novatel wireless modems through MSCP (Minstrel Status and Configuration Protocol). The PDA's CDPD module employs this library to obtain periodic status reports which contain information like cell tower signal strength (which can be used to minimize packet loss) and the tower ID.

Messages. All the messages that are exchanged between the PalmApp and the Agents2Go Server are encapsulated in a generic message format which specifies a sender ID, a message type and message content. Messages sent from a PDA to a Server use that PDA's ID as the Sender ID. Messages sent from a Server to a PDA use the Server's ID as the Sender ID. The message type field can contain one of three message types: “response form message”, “form request message”, or “form data message”. The Message Content field contains the message itself. The PalmApp uses a generic “form request message” to request forms from the Agents2Go Server that are displayed to the user. This message specifies the form name that the PalmApp is requesting and the cell tower, with which it is currently communicating.

On startup, PalmApp requests an “initial query form” from the Agents2Go Server. This form is used to issue user requests. These requests are converted into a “form data message” and sent to the Agents2Go Server. The response is a form that the PalmApp displays to the user that may contain a “home” button that causes the PalmApp to generate the “initial query form” message to allow the user to enter a new query. Unlike a “form request message” or a “form data message”, the “response form message” is initiated at the Agents2Go Server and destined for the PalmApp. This message contains a form that the PalmApp is required to display to the user. This message may contain a response to the user's query, an error message, etc.

Agents2Go Server. The Agents2Go Server is the component that handles messages to and from a PalmApp. User queries are forwarded to the Locator, and the corresponding responses are forwarded back to the PalmApp. Upon receiving a “form request message”, the Agents2Go Server reads the requested form from a file. If the desired form cannot be located, a suitable error form is sent back to the PalmApp. Once the desired form is located, the Agents2Go Server uses a lookup table to map the specified cell tower ID (obtained from the request message) to its neighborhood name. This neighborhood name is inserted into the location field of the form that needs to be displayed to the user. This neighborhood name, which can be changed by the user, is inserted into location field of the form displayed to the user. Thus a user, regardless of his current location, can find information about any participating region. This is encapsulated in a “response form message” and sent back to the PalmApp. When the Server receives a “form data message” from a PalmApp, it forwards it to the Locator. Other alternative designs could be used, a “form data message” could be forwarded to the corresponding Broker. Once the Agents2Go Server receives a response from either the Locator or a Broker, it generates the corresponding “response form message” and sends it to back the PalmApp.

Locator. The Locator is the component that plays the role of a proxy between the Agents2Go Server and a set of Brokers. The Locator receives requests from the Agents2Go Server, determines which area the request originated form and then forwards that request to the Broker responsible for that area. The Locator maintains a dynamic table that maps geographical areas to Brokers. The Locator listens on a well defined port for registration messages from Brokers. A Broker registration message specifies a port on which the Broker will accept requests that are forwarded by the Locator. The registration message also specifies the geographical area for which that Broker is responsible. Upon receiving a request form the Agents2Go Server, the Locator looks inside the request string and extracts the point of origin information. This information is used to determine the designated Broker. The Locator then forwards the request to that Broker. If the Locator is unable to locate a suitable Broker for the given request, the Locator sends a "broker not found" message back to the Agents2Go Server. A reliable communication channel is maintained between the Agents2Go Server and the Locator for all message transfers.

Broker. The Broker maintains information about restaurants in its designated geographical region. It also processes requests from users and generates suitable responses. These requests are forwarded to the Broker from the Locator and the generated responses are sent to the Agents2Go Server for forwarding to the requesting PalmApp. The system partitions participating restaurants into sets based on the geographical region in which these restaurants are located. Each coverage region is assigned a unique name and is serviced by a designated Broker. This Broker is responsible for generation of replies for requests pertaining to its coverage region. Our current implementation allows grouping of several geographical regions or partitioning a single geographical region to construct a coverage region.

Every Broker in the Agents2Go System is also associated with a specific Agents2Go Information Repository – a set of databases that contain information about participating restaurants in a Broker's coverage region. Restaurant information like name, address, cuisine etc. of all participating restaurants in that coverage region is distributed among these databases. This information can be classified as static, since it rarely changes. The Broker is also responsible for frequently changing restaurant information like waiting times and promotions. This kind of information can be classified as dynamic. This dynamic information is maintained within the Broker itself. The separation of dynamic and static information is done to reduce the number of messages that is exchanged between the Agents2Go System components.

It is common for wireless cells to overlap. If a user is in a cell overlap region, then that user's PDA connection can hop from one overlapping cell to another. So, the cell tower ID that the user's PDA picks up can change quite frequently. If the cell overlap is contained within a single coverage region, then cell hopping is not an issue. In this case any cell ID that is picked up in that cell overlap will map to the same coverage region name and is managed by the same Broker. However, if the cell overlap is on the border of two or more coverage regions, the user's PDA may pick up IDs that belong to cell towers that service neighboring coverage regions. This could create a scenario where a user's request query could be routed to a neighboring Broker that has absolutely no information about the current location of that user.

The Agents2Go System solves this issue by imposing a policy that prohibits coverage regions from overlapping unless there are some cell overlaps falling on their borders. We also require a special configuration for the Information Repositories that are associated with these overlapping coverage regions. Restaurants in overlapping regions are partitioned into two disjoint sets: a shared set and a native set. A shared set contains restaurants that are located in the areas of cell overlap (that fall on the borders of coverage regions) and a native set contains the remaining restaurants that are in the Broker's coverage region but not in the cell overlap. During his interaction with Agents2Go a user might move to a different coverage region, while the request is being processed. In this scenario, the reply to the request contains the information relevant to the region from where the request originated. This information could still be of relevance to the user since he/she is not far form the initial coverage region.

On initialization, a Broker establishes connection with its Agents2Go Information Repository. The Broker queries its repository to obtain IDs of restaurants for which it will broker information. If the Broker is unable to establish required connection(s) with its repository, the initialization fails and the Broker exits gracefully. Once a successful connection is established, the Broker builds a "waiting time" table. The "waiting time" table is a data structure that the Broker uses to maintain the dynamically changing restaurant information. The restaurant IDs and the location identifiers for the restaurants are used as keys of the table. The values of the table are the waiting time information, promotion information and time stamps of updates. Once the table is built the Broker registers itself with the Locator component. The registration contains geographical regions that this Broker administers and the port on which the Broker will listen for forwarded requests from the Locator. If the registration with the Locator fails, the Broker exits gracefully.

Once initialized, the Broker starts to listen for updates sent by the local Restaurant Agents and requests forwarded by the Locator. When a Broker receives a wait time update and some promotion information from a Restaurant Agent, it timestamps it and then caches this information into the "waiting time" table. When the Broker receives a request from the Locator, it first checks for the validity of the request. If the request string does not match the expected format, further processing of that request is terminated and an error message is sent back to the user. For valid requests, corresponding database queries are dynamically generated. Request parameters are dynamically incorporated into a database query.

If the request contains a waiting time limit, then the Broker searches its "waiting time" table for the restaurants that have their waiting time below the requested time limit. This search returns IDs of the restaurants that have suitable waiting time. Returned IDs are also incorporated into the database search query. Once the query is constructed, it is executed on the Broker's Agents2Go Information Repository. If no records are found, then a "No record found" message is returned to the user. If matching records are found, a timestamp for each result record is evaluated. This timestamp can belong to one of three age groups: "fresh" age group, "aged" age group, or "trashed" age group. A record will be treated differently depending on its age group, and on whether the user is interested in dynamic information.

To identify the age group of a timestamp, the difference between the value of that timestamp and current system time is calculated. This difference is the age of the timestamp. The timestamp age is compared against two threshold values. The first, lower, threshold denotes the limit between the "fresh "age group and the "aged" age group. The second, higher, threshold denotes the limit between the "aged" age group and the "trashed" age group. Hence, if a timestamp is "fresh", the record that has been selected is sent to the user, and the dynamic information is displayed in its regular format. Else, if a timestamp is "aged", the record is sent to the user along with a warning that the record is not up to date. And finally, if a timestamp is "trashed", the user request is analyzed to determine if the user is interested in dynamic information. If the user's request specifies a waiting time limit, the record is dropped from the result set. On the other hand, if there is no time limit specified, the record is sent to the user, but the dynamic, outdated portion of that record is replaced with an "Information is unavailable" message. This classification of the record's timestamps gives users some flexibility. Users themselves can determine if the dynamic information is useful.

Once the response to the query is formed, it is converted into an appropriate format and sent to the Agents2Go server. So there are two types of Restaurant Information messages that could be sent to the Agents2Go Server.

The Restaurant Agent. The Restaurant Agent resides and runs at the location of the participating restaurant and allows the restaurant manager to update dynamic information such as waiting times, promotion information, etc. Updates from the restaurant are sent to the Broker that is responsible for the geographical area in which the restaurant resides. If the restaurant is located in a cell overlap region, which is managed by several Brokers, then the update message is sent to every Broker that manages the overlap region. The update message contains the restaurant id, and other relevant information like the value of the wait time for table for two, the wait time for table for four, the wait time for table for six, etc. Each update message also contains a timestamp specifying the creation time. The Broker, upon receiving an update message, extracts the relevant values from the message and inserts these values into the appropriate row of its "waiting time" table.

We are currently developing an extended version of our system. For this version, we are porting our system to the iPAQ platform. We are also using 802.11b as the networking protocol. The functionality of the Locator and the Agets2Go Server have been merged with the PalmApp and the Broker components. In this version, the PalmApp directly communicates with the local Broker. We are currently working on a mechanism that allows The PalmApp to automatically discover local Brokers by receiving their heartbeat broadcasts.

An alternative way of discovering local Brokers is to use the Bluetooth Service Discovery Protocol. When a Bluetooth enabled device running a PalmApp component comes into proximity of a Bluetooth enabled server that is hosting a local Broker, the PalmApp will be able to discovers this Broker.

4.3 Conclusion

We have implemented a working prototype of the Agents2Go System as a location aware, distributed system that allows mobile users to request and receive various services information that is of most relevance to their current geographical location. Thus, the mobile users will not be burdened with extraneous information for services in remote locations. Also, the Agents2Go System allows service providers to supply dynamic service updates. This dramatically improves the value of the service for the providers and gives users more refined service information. Our implementation currently deals with restaurants, but it could be easily updated to work with other location specific services. All of the above mentioned features make our system well suited for various M-Commerce experiments using the existing CDPD infrastructure in use in the United States.

5 Conclusions

The *pervasive computing environments* of the near future will involve the interactions, coordination and cooperation of numerous, casually accessible, and often invisible computing devices. These devices, whether carried on our person or embedded in our homes, businesses and classrooms, will connect via wireless and wired links to one another and to the global networking infrastructure. The result will be a networking milieu with a new level of dynamism and openness. The localized and dynamic nature of their interactions raises many new issues that draw on and challenge the disciplines of agents, distributed systems, and security. This paper describes recent work by the UMBC Ebiquity research group which addresses some of these issues ranging from the need to develop a common communication infrastructure, to requirements for security and access control to techniques for location recognition.

References

1. A FIPA platform for handheld and mobile devices. In *Proceedings of the 2001 Workshop on Agent Theories, Architectures, and Languages*, 2001.
2. 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.
3. Harini Bharadvaj, A. Joshi, and Sansanee Auephanwiriyakyl. An active transcoding proxy to support mobile web access. In *Proc. IEEE Sumposium on Reliable Distributed Systems*, October 1998.

4. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.
5. M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis, The Role of Trust Management in Distributed Systems. *Secure Internet Programming, LNCS vol. 1603, Springer, Berlin, 1999, pages 185-210*,1999.
6. E.A. Brewer, R.H. Katz, Y. Chawathe, A. Fox, S.D. Gribble, T. Hodes, G. Nguyen, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. Padmanabhan, and S. Seshan. A network architecture for heterogeneous mobile computing. *IEEE Personal Communications Magazine*, 5(5):8–24, 1998.
7. A. Cedilnik, L. Kagal, F. Perich, J. Undercoffer, and A. Joshi. A Secure Infrastructure for Service Discovery and Access in Pervasive Computing. Technical Report TR-CS-01-12, CSEE, UMBC, August 2001.
8. Harry Chen, Anupam Joshi, Tim Finin, and Dipanjan Chakraborty. Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether. *Baltzer Science Journal on Cluster Computing, Special Issue on Advances in Distributed and Mobile Systems and Communications*, 2001.
9. R. Scott Cost, Tim Finin, Yannis Labrou, Xiaocheng Luan, Ian Soboroff Yun Peng, James Mayfield, and Akram Boughannam. An agent-based infrastructure for enterprise integration. In *First International Symposium on Agent Systems and Applications*, October 1999.
10. Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. 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, 1999.
11. DAML. Darpa agent markup language specification, http://www.daml.org/.
12. M. Dunham, P. Chrysanthis, A. Joshi, V. Kumar, K. Ramamritham, O. Wolfson, and S. Zdonik. Issues in wireless data management. Discussion of the wireless data management panel at NSF IDM PIs Meeting, March 2000.
13. M. Dunham, A. Helal, and S. Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2(2):149–162, 1997.
14. Carl M. Ellison, Bill Frantz, and Brian M. Thomas. Simple public key certificate. Internet document, 1996.
15. Mike Esler, Jeffrey Hightower, Tom Anderson, and Gaetano Borriello. Next century challenges: Data-centric networking for invisible computing. In *Mobile Computing and Networking*, pages 256–262, 1999.
16. FIPA. FIPA 97 specification part 2: Agent communication language. Technical report, FIPA - Foundation for Intelligent Physical Agents, october 1997.
17. R. Housley, W. Ford, W. Polk, and D. Solo. RFC2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile, January 1999.
18. Internet Engineering Task Force. Public-key infrastructure (x.509), http://www.ietf.org/html.charters/pkix-charter.html.
19. Armando Fox, Brad Johanson, Pat Hanrahan, and Terry Winograd. Integrating information appliances into an interactive workspace. *IEEE Computer Graphics and Applications*, 20(3), 2000.
20. Benjamin Grosof and Yannis Labrou. An approach to using xml and a rule-based content language with an agent communication language, 1999.
21. Z. J. Has. Panel report on ad hoc networks - MILCOM'97. *Mobile Computing and Communications Review*, 2(1), January 1998.

22. A. Helal, M. Wang, A. Jagatheesan, and R. Krithivasan. Brokering based selforganizing e-service communities. In *Fifth International Symposium on Autonomous Decentralized Systems (ISADS)*, Dallas, Texas, March 2001.
23. T. Hodes and R. H. Katz. A document-based framework for internet application control. In *Proceedings of the Second USENIX Symposium on Internet Technologies and Systems*, October 1999.
24. W. Johnston and C. Larsen. A use-condition centered approach to authenticated global capabilities: Security architectures for large-scale distributed collaboratory environments. Technical Report Technical Report 3885, Lawrence Berkeley National Laboratory, 1996.
25. A. Joshi, S. Weerawarana, and E. N. Houstis. Disconnected Browsing of Distributed Information. In *Proc. Seventh IEEE Intl. Workshop on Research Issues in Data Engineering*, pages 101–108. IEEE, April 1997.
26. Anupam Joshi. On proxy agents, mobility and web access. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2000. (accepted for publication, also availbe as UMBC CS TR 99-02).
27. Lalana Kagal, Tim Finin, and Yun Peng. A framework for distributed trust management. In *To appear in proceedings of IJCAI-01 Workshop on Autonomy, Delegation and Control*, 2001.
28. Lalana Kagal, Vlad Korolev, Harry Chen, Anupam Joshi, and Tim Finin. Centaurus: A framework for intelligent services in a mobile environment. In *Proceedings of International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, The 21st International Conference on Distributed Computing Systems (ICDCS-21) April 16-19, 2001.
29. Lalana Kagal, Vladimir Korolev, Sasikanth Avancha, Anupam Joshi, Timothy Finin, and Yelena Yesha. A highly adaptable infrastructure for service discovery and management in ubiquitous computing. Technical Report TR CS-01-06, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 2001.
30. R. H. Katz, E. A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G. T. Nguyen, V. Padmanabhan, and M. Stemm. The bay area research wireless access network (barwan). In *Proceedings Spring COMPCON Conference*, 1996.
31. Yannis Labrou, Tim Finin, and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, / 1999.
32. Li, Feigenbaum, and Grosof. A logic-based knowledge representation for authorization with delegation. In *PCSFW: Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.
33. M. Liljeberg, M. Kojo, and K. Raatikainen. Enhanced services for world-wide web in mobile wan environment. http://www.cs.Helsinki.FI/research/mowgli/mowgli-papers.html, 1996.
34. D. Maltz and P. Bhagwat. Msocks: An architecture for transport layer mobility. In *Proc. IEEE Infocom 98, San Francisco*, pages 1037–1045, April 1998.
35. M.Blaze, J.Feigenbaum, and J.Lacy. Decentralized trust management. *IEEE Proceedings of the 17th Symposium*, 1996.
36. S. Muthukrishnan, Rittwik Jana, Theodore Johnson, and Andrea Vitaletti. Location based services in a wireless wan using cellular digital packet data (cdpd). In *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE01)*, May 2001.

37. B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R.Walker. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles.*
38. RSA Laboratories. PKCS 11 - Cryptographic Token Interface Standard, January 1994.
39. Reticular Systems. The yellowstone project.
40. Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.
41. Mark Taylor, William Waung, and Mohsen Banan. *Internetwork Mobility: The CDPD Approach*. Professional Technical Reference. Prentice Hall, 1996.
42. W. J. Tolone, B. Chu, J. Long, T. Finin, and Y. Peng. Supporting human interactions within integrated manufacturing systems. *International Journal of Agile Manufacturing*, 1998. To appear.
43. 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*, 2002.
44. W3C. Xml signature http://www.w3.org/signature/.
45. Philip R. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, MA, USA, 1995.