Me-Services: A Framework for Secure & Personalized Discovery, Composition and Management of Services in Pervasive Environments[⋆]

Anupam Joshi, Timothy Finin, and Yelena Yesha

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
{joshi,finin,yeyesha}@cs.umbc.edu

Abstract. Increasing computational capabilities and decreasing costs of commonly used devices, and the concomitant development of short range, ad-hoc networking technologies, will help realize the pervasive computing paradigm. In this paper, we present the use of semantically rich descriptions for devices to discover, and cooperate with, others in their vicinity. In particular, we describe our ongoing projects that have used DAML for service discovery, service composition, data management, and trust based security in pervasive computing environments.

1 Introduction

Computing today is on the way to becoming pervasive. The computational capability of commonly used devices, like cellular phones and handhelds, and their ability to wirelessly communicate with other devices are driving forces behind the progress of pervasive computing. Computing capability will increasingly be embedded in all manner of manufactured artifacts, from roads and bridges to even the clothes we wear. Wireless technologies are playing an important role in allowing the multitude of devices to spontaneously form short-range, short-term, *ad-hoc* networks. The various flavors of the IEEE 802.11 standard and Bluetooth are the technologies du jour, and their variants will likely dominate the near future.

Traditionally, much of the research in academia and industry has been, and continues to be, focused on the development and improvement of these protocols. The next, and perhaps most important (from user perspective) step, is for useful applications to exist on the devices forming these networks. In the past year or two, the research community has seen plenty of hype associated with wireless / 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 few months have also seen pundits declaring, with equal certainty, that M-Commerce is either a non-starter or that it is dead. In large

⋆ This work supported in part by NSF awards 9875433, 0070802, and the DARPA DAML program.

part, the blame for both the initially 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, some combination of PDAs and/or cell phones essentially 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 stuff via their cellphones 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, and connection is made via cellular phone based wide area networks. This approach (typically based on a client–proxy–server model) has been developed by the academia and industry over the last five–six years in contexts such as web access from mobile platforms (for instance [10, 14, 5, 16, 19, 2, 11]) . 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 is leading efforts amongst 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.

Yet 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 pervasive computing/ad-hoc networking scenarios enabled by Bluetooth like technologies. Staying with the M-Commerce idea, consider a move away from the prevailing mobile storefront 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. 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. In such pervasive environments, much of the interaction between the users' devices and the services available around them must happen in the background, since the user can be overwhelmed by the services all around her. M-Commerce will thus need to be transformed into what we refer to as Me-Commerce, personal as well as pervasive!

In order to satisfy user requests, applications on a device *discover* services or information on other devices in their vicinity. In some situations, information or services available on multiple devices are *composed* together before being presented to the user. Users rarely *query* the system in the traditional database sense, but often declaratively specify things they are interested in. The system in turn needs to query data sources and service providers, most of which are volatile in as much the vicinity of a device keeps changing with movement. Moreover, much of this interaction with other devices happens outside of a users home environment, which means that devices in the vicinity cannot by default be assumed to be trustworthy. The eBiquity group (`http://research.ebiquity.org`) at the University of Maryland, Baltimore County is involved in the design, development and evaluation of an integrated framework that addresses these issues. This paper describes the vision behind our work, the broad design of the system, and some themes that emerge in the design. It also briefly describes our ongoing efforts to realize a framework to enable this vision.

2 A Scenario for the Future

Let us now look at a scenario set in the future. Philip is traveling in a car with a built-in Global Positioning System (GPS) device and has a IEEE 802.11/Bluetooth-enabled personal digital assistant (PDA). His colleague in the car, Mark, suddenly takes ill and must be taken to a hospital immediately. Philip has to find the route to the nearest hospital and the traffic conditions on the road are really bad. Philip does not want to simply find any route to the hospital, but one that consists of least crowded highways, roads and streets. This makes static car based route generation systems useless.

The traditional infrastructure oriented mobile systems solution to this problem is to ensure that Philip's PDA establishes a connection to some central server that can provide directions, maps and some traffic information. One can assume that Philip's cellular phone would provide near-continuous connectivity to the Internet, thus allowing the PDA to pose certain queries to the server and obtain answers so that it can perform the required tasks. The implicit assumption here is that any arbitrary service we might need will be provided as a packaged, monolithic entity on the wired side. This assumption is questionable – the user base for some services will be so small (sometimes 1 person!) that it may not make economic sense to provide them. Even if we stick only to technical issues, We believe that this solution is not the most efficient. One obvious problem is the scalability of the centralized system that handles service requests. With a large number of wireless devices attempting to connect to it, the traffic/route server quickly becomes a bottleneck. Another problem is that dynamic information such as current traffic conditions are unlikely to be up-to-date on the server. We note that Philip in this scenario does not benefit from the knowledge that the outer loop of the DC Beltway is "slow" from Silver Spring to the American Legion bridge (a stretch of 10-15 miles); however he would benefit immensely from knowledge that a particular portion of the highway (270-Beltway merge) or some major intersection has suddenly become congested perhaps due to traffic volume. Yet another problem is the latency of the connection between the PDA and the server. In a dynamically changing environment, information that takes too long to reach the PDA might well be useless. If, for example, the PDA instructs Philip to "take exit 33" after he has passed it, that information is stale and useless.

Consider an alternate approach. The neighborhood of Philip's car is information and service rich because devices in vehicles around it, and the vehicles themselves are continuously engaging in conversations with their neighbors thus obtaining (and perhaps storing) useful information & services. Using our architecture, therefore, Philip's PDA behaves as follows: It decides that in order to satisfy the goal, it needs to discover a dynamic traffic information service provider, a road map service provider and a route calculating service provider. First, the PDA determines its current location by connecting to the car's built-in GPS device. It then discovers the presence of a Road Map Atlas

CD-ROM at the neighboring car, provides it with the (current location) and (any hospital) as the end points, and asks for several alternate routes. Once it knows which roads it could use, it discovers current traffic condition information from cars traveling in the opposite direction. This information, along with the location and map information, are given as inputs to an on-board route calculating service in a nearby high end BMW to finally obtain the *best route to the nearest hospital*.

Our attempt is to design frameworks that would enable such systems to be built in a straightforward manner.

3 Design Principals and Architecture

An abstract view of the framework we are building is illustrated in Figure 1. We assume the existence of functionalities that roughly come up to layer 4 of the OSI model (i.e. transport layer) and build our framework on top. The functionalities of our system are:

- *Semantic Service Discovery*: The process of discovering information and services – described in a structured manner – based on the *semantics* of the descriptions rather than their syntax. This capability is critical in order to allow devices to know about what else is in their vicinity in unknown environments.
- *Service Composition*: The process of creating new services from existing services by dynamically discovering, integrating and executing the existing services in a planned manner. Most users are used to accessing certain services in their home environment. Such a facility would allow users to get services they are used to or need in environments where they may not be available by default.
- *Profile-Driven Management*: The process of intelligently managing the access, storage, monitoring, and manipulation of data and information based on context constraints. The context involves the environment (such as communication bandwidth, cache space, location, battery power) as well as the users preferences and intentions. The latter requirements are assumed to be specified in user profiles.
- *Distributed Trust Management*: The process of using trust relationships as a way of authenticating users and providing access control. Pervasiveness of computing means that a user would often be interacting with environments (e.g. an airport, other cars around on a highway) where her identity and the identity of those around her are not a priori known. Distributed Trust based approaches provide the framework on which we build to create security (and privacy) in such systems

In the process of designing the system, several interesting themes and issues stand out.

Perhaps the most important is the need for **adaptivity**. This need has been identified even by prior work on Mobility, for instance, Satyanarayanan's Odyssey system[18], the Berkeley TACC system[5], and our work on Mowser[2, 10]. However, the parameters to which the system had to adapt were relatively few (mostly network resources), and the adaptation process itself was in general well understood. The environment we work on depends adaptivity at multiple levels, including to network and other resources. Some of this is extending existing approaches (e.g. The device is in location x, find all Italian restaurants around it). Other kinds of adaptation are far more difficult to characterize

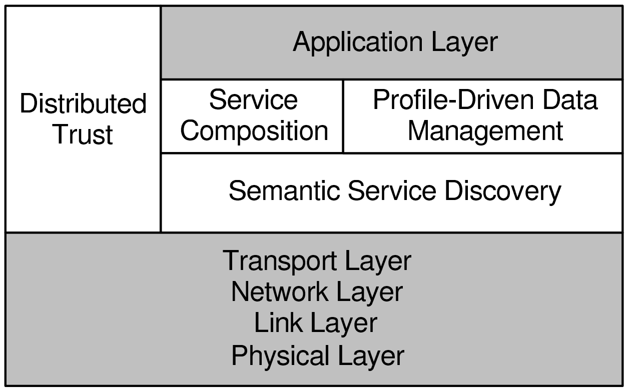

Fig. 1. Architecture for Secure Information Discovery, Composition and Management

and implement. Consider for instance the following cache replacement algorithm: Since it is 6pm and the user is not at home, I should throw away items cached from the TV guide and replace them with Italian restaurant advertisements that I gather as my user walks in the downtown area. What is cached (and thus what should the replacement policy be) is a fairly complex function of the user, her location (away from home), her habits(watches TV when at home, eats dinner around 7pm). Many of the components of our environment seek to support this.

We believe that two classes of "tools" are essential building blocks to enable adaptivity: *semantic models/languages* and *reasoning engines*. A semantic model or language is necessary to describe information in a well-defined and structured manner so that machines, rather than humans, can "read" and "understand" it. The Resource Description Framework (RDF)[15, 6], and the DARPA Agent Markup Language (DAML)[9] are examples of a semantic model and language, respectively. Information in a structured manner lends itself to easy conversion to statements in first-order logic. Reasoning engines can use these statements to derive new facts and rules that would lead them closer to the "ideal" adaptation to a user. From a systems aspect, integrating these traditionally "heavy" components into the generally "lightweight" pervasive environments presents interesting challenges.

In the environment we envisage, users and their devices will interact with many others around them. Each device/user in such circumstances is an autonomous entity, and her interactions with others are governed by her beliefs, desires, and intentions. The discipline of multi-agent systems has studied the interactions of such entities, and we feel that a **multi-agent abstraction** is best suited to develop the framework for how devices behave in an mobile ad-hoc connectivity environments. The BDI formalism[22] can provide a theoretical framework in which to examine interactions in an ad-hoc network, and evolve strategies for appropriate adaptation. Semantically rich profiles of a user or device present one way in which to capture beliefs/desires/intentions. These can either be pre-declared (The user checks off a box saying I prefer Chinese food), or inferred (the users personal agent "mines" his actions offline and figures out that the user eats dinner around 8pm when at home and 7pm when on a business trip).

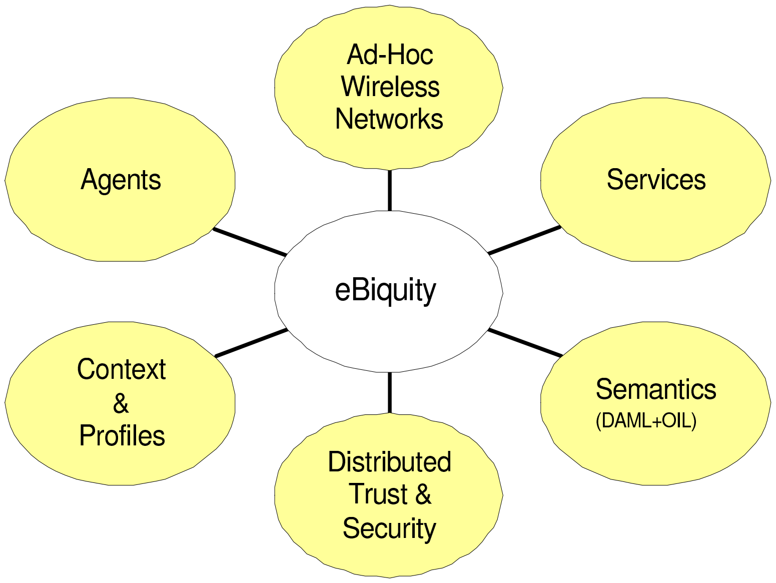

Fig. 2. Elements of the Design Framework

In summary, we feel that the challenges posed by this new generation of mobile systems can be addressed by combining elements of multi-agent systems, wireless ad-hoc networks, semantic languages/models, distributed trust, context driven systems, and service based architectures. Figure 2 presents this confluence.

4 Semantic Service Discovery

The fundamental components of a service discovery protocol or mechanism are the *service description* and the *matching technique*. In most prevailing protocols, the service description typically contains the following information: name of the service, a unique service identifier, the expected *inputs* to the service, the name and/or address of the service provider, the protocol(s) required to use the service and other attributes describing specific service capabilities. The matching technique is typically tied to either the service name, the unique identifier, the service invocation mechanism or a combination of all three. For example, the matching technique in Jini– the well-known Java-based service discovery protocol – uses the *method interface* to match service requests against the service descriptions, while that in the Bluetooth Service Discovery Protocol (SDP) uses *Universally Unique Identifiers* (UUID). Other commonly used protocols like Salutation, Microsoft's UPnP and UDDI use description and matching techniques similar to those of Jini and Bluetooth SDP. The common thread in all these protocols is the use of *patterns* – UUIDs, interfaces, names – to implement the matching technique. We argue that service discovery based on pattern matching is inefficient in a mobile, ad-hoc environment because the heterogeneity of service interfaces in such a domain will result in more failed matches than successful ones. For example, the road map service discovery attempt by Philip's PDA may fail simply because it specifies a "RoadMapService" in its request, whereas the interface to the map service on a neighboring device is called "MapService". Such failures lead to inefficient service discovery because many neighboring devices may have to be queried before a match is found. In a dynamically changing environment, such as a car on a highway, querying multiple devices may not be an option. On the other hand, *semantic* description of both the "RoadMapService" and the "MapService" would contain information that allows the matching technique to *infer* that the two are equivalent in terms of functionality. In addition, semantic matching allows the service discovery requester to specify additional constraints on the attributes such as priority, matching rule etc.

Semantic service discovery uses a semantic model/language to describe a common ontology for services and a reasoning engine to draw inferences between various descriptions, based on the ontology. The ontology attempts to capture and represent as much knowledge as required, about services in an ad-hoc network environment. In general, an ontological description allows all parties in a transaction to understand each other unambiguously. Service discovery requests also use the semantic language, so that the reasoning engine can determine the *closest* possible match between the request and the available services.

We have enhanced the Bluetooth SDP to use semantic information associated with services and attributes to decide the success or failure of a query. We have developed two versions of Bluetooth SDP that support semantic matching and provide service registration. The first version uses the RDF/RDF-S data model and the second version uses DAML. We have conducted experiments to evaluate the performance of the enhanced SDP in terms of Round Trip Time (RTT) and server side processing time. In comparison to regular SDP, we observe that the use of a heavyweight reasoning engine imposes little or no penalty on service discovery in terms of processing time. This result implies that while the negative effects of the reasoning engine are negligible, the positive aspects like more successful service discovery ensure that the enhanced SDP is a good choice. The system and experimental results are detailed in [1]. We plan to extend this implementation to other service discovery protocols over different communication media.

5 Service Composition

The need for service composition in a pervasive environment arises when the functionalities of more than one service are required to satisfy a given request. Although it is possible to statically combine the multiple functionalities into one monolithic service, one can easily see that this is not a good idea. An alternative is to statically decide "useful" combinations of all services, and then pre-compose them. Of course, it is virtually impossible to determine all possible combinations of useful services that can be created from a given set of services, since the notion of usefulness is a function of the user. Given that we do not a-priori know who the user might be, defining usefulness is not possible up front. In other words, static combinations of services in a mobile adhoc networks are unlikely to be used frequently, unless they are very common tasks.

We believe that *dynamic* service composition is the ideal solution for pervasive environments. Service composition has been studied in depth in the context of wired or Internet-based services. Service discovery and composition platforms like UDDI and HP's eFlow engine (a part of the eSpeak architecture)[7] provide a centralized platform for composition of e-services for complex queries like planning a business trip. Languages like WSDL and DAML-S have been proposed to describe composite services in a structured manner. The fundamental assumption of these composition platforms – one that often fails in dynamic, pervasive environments – is that the services exist in a stable wired environment and that the composition can be centralized.

We have developed a distributed architecture to perform service composition in a pervasive environment. Central to our system is the concept of a distributed broker that can execute in any node in the environment. An individual broker handles each composite service request, thus making the design of the system immune to central point of failure. A broker may be selected based on various parameters like resource capability, geometric topology of the nodes and proximity of the node to the services that are required to compose a particular request.

We have developed a system that is capable of *reactive* service composition in a pervasive environment by dynamically discovering, integrating and executing individual services available in nodes that are connected in an ad-hoc manner. We have described composite services using DAML-S[8] in terms of subset of individual services that might be able to satisfy a composite request. The individual services are described based on inputs, outputs, functionality classification, functional similarity to other services and invocation mechanisms. The service description also incorporates platform specific information like the current execution platform and resources available in that platform. When a request for a composite service comes to the composition engine, it takes help of the underlying semantic service discovery mechanism to obtain information about the availability of the necessary services in the vicinity. The composition engine decides on the best available platform to carry out the composition based on the resource availability of the platforms. It obtains this information while doing service discovery. Each request in this environment can thus be assigned a separate platform to carry out the composition. Reactive service composition in highly dynamic environments where the services around a device are rapidly changing (due to mobility) can be done by discovering and executing services in a sequential manner. In environments where the services are liable to be available for a longer duration (say a group meeting in a conference room), the discovery process can be done first, followed by execution where certain optimizing criteria can be applied. The current prototype of the system has been implemented over Bluetooth [20].

Our current work includes designing a *proactive* mechanism for service composition, which allows a node to obtain and compose commonly used services, before the application requests such a service. We also plan on implementing a distributed brokering approach where the task of a single composite service hops from one broker to another depending on various factors like service availability and number of hops.

6 Profile-Driven Management

The management peer-to-peer services is fundamental to pervasive environments. It is crucial in enabling mobile, wireless devices to collaborate with peers to achieve high service availability. An ideal system allows every device complete access to local and remote services. For example, in the scenario developed earlier Harry's PDA could cache the traffic conditions on particular roads that he obtained by composing services/information from nearby cars, and offer to provide this updated information as a service to others so that they do not have to go through the discovery/composition procedures should they also need that information. Some neighboring devices would accept this new service, because they know from their user's profile that she has several alternate routes to use and likes to be alerted if her usual route gets backed up. Others will decline to use the service, because they have no use for it, or they do not feel that information provided by Harry's PDA can be trusted. Moreover, Harry's own profile could prevent arbitrary propagation of available services. In general, for pervasive systems to succeed, much of the interaction between the devices must happen in the background, without explicit human intervention. These interactions should be executed based on information in the profile. For instance, a diabetic user's profile can say "Always keep track of the nearest hospital", influencing what services will seek to obtain and which service providers it will interact with. Of course, the question arises: "what exactly should a profile contain?"

We argue that the existing notion of profiles, which explicitly enumerate data and its utility, are not sufficient. A profile should not simply consist of information about utility values of fixed service/data domains, since in pervasive computing environments both the domains of service/data which a user may need, as well as its utility, will change with the changing context of the user. We believe that a profile should be described in terms of "beliefs", "desires", and "intentions" of the user, a model which has been explored in multi-agent interactions [22]. The "beliefs" represent information that should be treated as facts, and assigned "utility" and "reliability" values or functions to enable comparison with other information stored in a profile. For example, these may include information about user's schedule or cuisine preferences. A "desire" represents a wish the user would like to accomplish. Each desire is also assigned "utility" value and function. Lastly, an "intention" represents a set of intended tasks – these can either be deduced from "desires", or be explicitly provided. For example, the profile may contain user's desire to listen to country music as performed by Shania Twain. The system should deduce from this the intention to download Shania Twain MP3s. This intention would then influence the behavior of the user's PDA. Alternately, a user may explicitly provide an intention to purchase Shania Twain CDs. Upon entering a mall, the device will try to obtain information about new releases from the local music stores. The "intentions" of user, modulated by the "beliefs" as well as contextual parameters, including location, time, battery power, and storage space, allow each entity to determine what services to obtain and their relative worth. We note here that the context information could also be regarded as "beliefs" that were dynamically asserted and retracted.

We think that a semantically rich language, such as DAML+OIL/DAML-S, provides a rich framework to represent our enhanced profiles. The advantages of this selection are two fold: By adhering to an already existing language, the syntax and rules do not have to be duplicated by creating a new formal language. Secondly, by utilizing a language used by the Semantic Web, the devices will be able to use the vast resources available on the Internet as well as the resources available in ad-hoc networks. We have designed and implemented a prototype called MoGATU[21] that provides some of this functionality. The user profile contains preferences and other information, and serves a base for predicting future data interaction. Our current implementation of this system works with both Bluetooth and 802.11 (both base-station oriented and ad-hoc).

7 Distributed Trust Management

Traditionally, stand-alone computers and small networks rely on user authentication and access control to provide security. These physical methods use system-based controls to verify the identity of a person or process, explicitly enabling or restricting the ability to use, change, or view a computer resource. However, these mechanisms are inadequate for the increased flexibility that pervasive computing environments require. The main assumptions of existing mechanisms are that the environments possess central control and that their users are all predetermined. Even role based access control systems, which relax some of these assumptions, still require that roles be predetermined and known universally. Users of pervasive computing environments expect to access locally hosted resources and services anytime and anywhere via some handheld device, causing serious security risks and access control problems.

We suggest using *distributed trust* as a way of authenticating and authorizing users through the help of trusted entities, without having to completely authenticate the users or find their exact access rights in the system.

We view trust management as the *establishment of trust relationships* instead of quantifying trust. Trust relationships are formed through *delegations*. If a trusted user (by trusted we mean a user whose credentials are acceptable and whose reputation is fair) delegates certain rights to another user, this delegation is also trusted. A trust relationship is formed between the system and the delegatee for the delegated rights, that is the user is trusted by the system only in relation to those rights. The delegatee is given those rights based on the conditions specified in the delegation statement.

Our approach involves articulating policies for user authentication, access control, and delegation; assigning security credentials to individuals; allowing entities to modify access rights of other entities by delegating or deferring their access rights to third parties and revoking rights as well; and providing access control by checking if the initiators' credentials fulfill the policies [3, 4] of the local domain. The policies can be specified in DAML (our present implementation uses prolog, but we are in the process of transition). In general, a previously unknown user is assigned some generic rights deduced by reasoning over his/her credentials, the security policy of the environment, and delegations from other users.

Users can also make requests to access other services. Appropriate users with these access rights can decide to delegate the requested right to them. A user can delegate all rights that it has the permission to delegate. Rights can also be revoked; so rights are no longer static, but can change based on delegations and revocations.

We have applied this distributed trust model to several scenarios [12, 13]. Vigil is a system we developed recently that is designed to provide security and access control in distributed systems, and has been optimized to work in pervasive computing environments, where most of the clients are handheld devices. We have developed Vigil by combining SPKI and Role Based Access Control [17, 23] with *trust management*.

The system contains a Security Agent that is responsible for maintaining distributed trust. It enforces the security policy of the organization. It interprets the policy to provide controlled access to Services and uses distributed trust. A policy includes rules for role assignment, rules for access control, and rules for delegation and revocation.

The Security Agent uses a knowledge base and sophisticated reasoning techniques to handle security and distributed trust. On initialization, it reads the policy and stores it in a Prolog knowledge base. All requests are translated into Prolog, and the knowledge base is queried. The policy contains *permissions* which are access rights associated with roles, and *prohibitions* which are interpreted as negative access rights. The policy also contains rules for role assignments, access control and delegation. A user has the ability to access a service if the user has not been prohibited from accessing the service by an authorized entity and if it either has the role based access right or if some authorized entity has delegated this right to it. An entity can only delegate an access right that it has the ability to delegate. When a user needs to access a service that it does not have the right to access, it requests another user, who has the right, or the service itself, for the permission to access the Service. If the entity requested does have the permission to delegate the access to the Service, the entity sends a delegate message, signed by its own private key, along with its certificate, to the Security Agent and the requester. The Security Agent checks the roles of the delegator and the delegatee and ensures that the delegator has the right to delegate, and that the delegation follows the security policy. It then adds the permission for the Client to access the Service, but sets a very short period of validity for the permission. Once this period is over, the Security Agent has to reprocess the delegation. This is very useful in case of revoked certificates, delegations or rights. If any one entity in the delegation chain loses the permission, then it is propagated down the chain very quickly, till everyone after the entity loses the ability.

We have developed permissions and prohibitions in Vigil and are looking into the usefulness of obligations and entitlements as additional security mechanisms in pervasive systems. We are exploring *distributed belief* as a way for the Security Agent to garner the required trust information. The policy could include rules for belief as well. For example, the Security Agent would believe that Mark had the role of Manager in XYZ, if two registered and trusted clients say that it is true.

8 Conclusions

We have described the work of the eBiquity group at UMBC toward building a framework that enables discovery, composition and management of services and information in a pervasive computing environment. We describe the overall vision and architecture, show how DAML/DAML-S are useful, and present brief snippets of the ongoing projects to realize the framework.

References

1. S. Avancha, A. Joshi, and T. Finin. Enhancing the Bluetooth Service Discovery Protocol. Technical report, University of Maryland Baltimore County, August 2001. TR-CS-01-08.
2. H. Bharadvaj, A. Joshi, and S. Auephanwiriyakyl. An Active Transcoding Proxy to Support Mobile Web Access. In *Proc. of the IEEE Symposium on Reliable Distributed Systems*, October 1998.
3. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. *The Role of Trust Management in Distributed Systems*, volume 1603 of *LNCS*, chapter Secure Internet Programming, pages 185–210. Springer, Berlin, 1999.
4. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. *IEEE Proceedings of the 17th Symposium*, 1996.
5. 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.
6. D. Brickley and R. Guha. Resource Description Framework (RDF) Schema Specification 1.0 - W3C Recommendation. `http://www.w3.org/TR/2000/CR-rdfschema-20000327`, 2000.
7. F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and Dynamic Service Composition in eFlow. Technical Report, HPL-200039, Software Technology Laboratory, Palo Alto, CA, March 2000.
8. The DAML Services Coalition. DAML-S: Semantic markup for web services. http://www.daml.org/services/.
9. J. Hendler. DARPA Agent Markup Language. `http://www.daml.org`, 2000.
10. A. Joshi. On proxy agents, mobility and web access. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2000.
11. A. Joshi, S. Weerawarana, and E. N. Houstis. On Disconnected Browsing of Distributed Information. In *Proc. of the Seventh IEEE Intl. Workshop on Research Issues in Data Engineering*, pages 101–108. IEEE, April 1997.
12. L. Kagal, T. Finin, and Y. Peng. A framework for distributed trust management. In *Second Workshop on Norms and Institutions in MAS, Autonomous Agents,2001*, 2001.
13. L. Kagal, J. Undercoffer, T. Finin, and A. Joshi. Vigil : Enforcing Security for Pervasive Enviroments. In *Under review*, 2001.
14. 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.
15. O. Lassila and R. Swick. Resource Description Framework. `http://www.w3.org/TR/1999/REC/rdf-syntax-19990222`, 1999.
16. 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.
17. E. C. Lupu and M. Sloman. Towards a Role Based Framework for Distributed Systems Management. *Journal of Networks and Systemss Management, Plenum Press*, 1996.
18. B. D. Noble, M. Price, and M. Sathyanarayanan. A programming interface for application-aware adaptation in mobile computing. In *Proceedings of the Second USENIX Symposium on Mobile & Location-Independent Computing*, Ann Arbor, MI, April 1995.
19. 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*, 1996.
20. Bluetooth White Paper. World Wide Web.

21. F. Perich, S. Avancha, A. Joshi, Y. Yesha, and K. Joshi. Query routing and processing in mobile ad-hoc environments. Technical Report TR-CS-01-18, CSEE, UMBC, November 2001. short version accepted for publication at DEXA 2002.
22. A. Rao and M. Georgef. Modelling rational agents within a bdi architecture. In *Proc. of the Int. Conf. on Knowledge Representation and Reasoning*, pages 473 – 484, San Mateo, CA, 1991. Morgan Kaufmann.
23. R. S. Sandhu. Role-Based Access Control. In M. Zerkowitz, editor, *Advances in Computers*, volume 48. Academic Press, 1998.