An Integrated Architecure for Secure Information Discovery, Composition and Management in Pervasive Environments

Sasikanth Avancha, Dipanjan Chakraborty, Lalana Kagal, Filip Perich, Anupam Joshi, Timothy Finin, Yelena Yesha
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
e-mail : {savanc1,dchakr1,lkagal1,fperic1,joshi,finin}@cs.umbc.edu

I. Introduction

Computing in this day and age is on the way to becoming completely ubiquitous and 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. Wireless technologies of the hour – the various flavors of the IEEE 802.11 standard and Bluetooth – are playing an important role in allowing the multitude of devices to spontaneously form short-range, short-term, *ad-hoc* networks. In order to support and maintain these networks, efficient routing and transport protocols are required. Much of the research in academia and industry has 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 applications on the devices forming these networks to exchange information with one another. Unlike devices in infrastructure-based networks, however, those in ad-hoc networks are often restricted in terms of power and memory capabilities. These restrictions reduce the possibility that the devices can store all required information and services locally. Thus, in order to satisfy user requests, applications on a device must be able to *discover* information on other devices in the ad-hoc network. In some situations, information available on multiple devices must be *composed* together after being discovered, before being presented to the user. Users often pose *queries* to applications, which in turn may query other applications in the ad-hoc network, to eventually obtain an answer. Needless to say, all of these tasks must be performed in a *secure* manner.

To this end, the eBiquity research group (`http://research.ebiquity.org`) at the University of Maryland, Baltimore County is involved in the design, development and evaluation of an integrated architecture whose primary components 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 patterns present in them.
- *Service Composition*: The process of creating complex services from simple, easily executable lightweight services by dynamically discovering, integrating and executing the simple services in a planned manner, to satisfy a client request.
- *Profile-Driven Data Management*: The process of intelligently managing the access, storage, monitoring, and manipulation of data and other resources, such as communication bandwidth, disk bandwidth, and cache space, based on user data requirements specified in profiles.
- *Distributed Trust Management*: The process of using trust relationships as a way of authenticating users and providing access control.

In designing this architecture, we have chosen two classes of "tools" that we believe are essential building blocks: *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) and the DARPA Agent Markup Language (DAML) 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 statements that would lead them closer to the "ideal" response to a user request. The most common reasoning engines are based on Prolog.

In the following sections, we describe the applicability and overview of this architecture and briefly discuss the components.

II. A Scenario for the Future

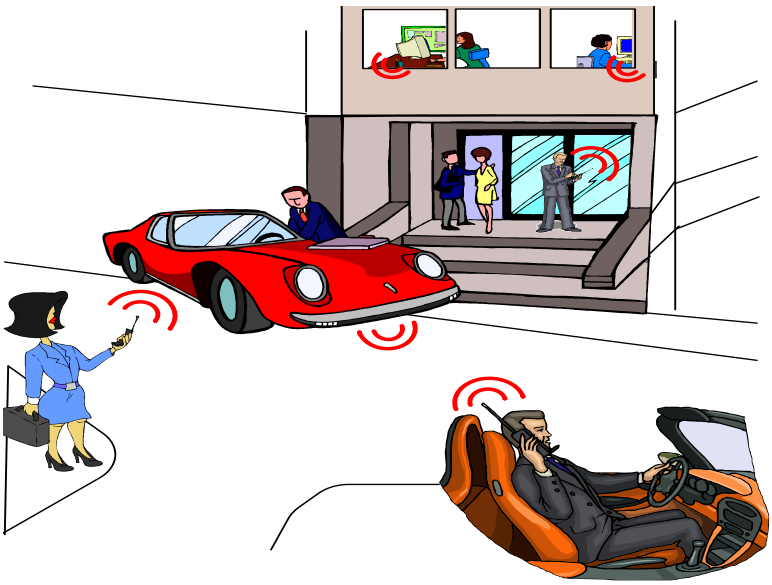

Fig. 1. Philip's PDA/Phone in an Ad-Hoc Environment

Let us now look at a scenario set in the future. Philip is traveling in a car and has a IEEE 802.11b/Bluetooth-enabled personal digital assistant (PDA) with a Global Positioning System (GPS) device attached to it. His colleague in the car, Mark, suddenly takes ill and must go to hospital immediately. Philip has to find the route to the nearest hospital and the traffic conditions on the road are really bad. So, Philip does not want to simply find any route to the hospital, but one that consists of least crowded highways, roads and streets. The first task for Philip's PDA is to *discover* services in his (mobile) neighborhood that could help achieve Philip's goal. These services could be in other cars alongside or in buildings that he is driving (at possibly 60 mph) past. The PDA should be able to *compose* these services together in order to obtain the "best" route. It must *query*, *obtain* and *manage* information provided by the services it discovers.

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 using the GPS device. It then discovers the road map service and provides this location and the hospital's location as inputs to the road map service provider. The traffic information service is discovered next; its input is the result from the road map service. Current traffic conditions are thus obtained. These, along with the location and map information, are given as inputs to the route calculating service to finally obtain the *best route to the nearest hospital*. Philip now realizes that Mark's hospital records must be obtained from his doctor's office and made available to the hospital they are headed to. He does this using his *trust* relationship with Mark.

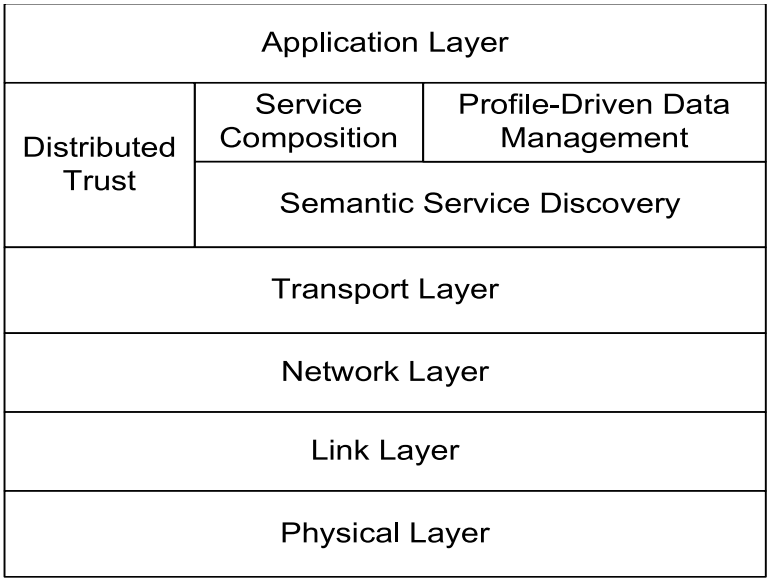

Fig. 2. Architecture for Secure Information Discovery, Composition and Management

III. Overview of the Architecture

IV. 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 disocvery 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 addition, semantic matching allows the service disocvery requestor to specify additional constraints on the attributes.

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 disocvery 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 [1]. 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 plan to extend this implementation to other service discovery protocols over different communication media.

V. 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 imagine a situation where one would require the service to possess additional capabilities. So, this service needs to be combined with at least another service to produce the desired results. Static combinations of services in a dynamic environment, such as a mobile ad-hoc network, are likely to be used infrequently, unless they are very common. For example, because Philip's PDA is attempting to determine the least crowded route to the nearest hospital, a road map service must be combined with a route calculating service. This is required only in emergencies. Hence, for some device to store a combination of these two device might simply be a waste of resources. Thus, storing such combinations on low-power, resource constrained devices may not be feasible. This leads us to believe that *dynamic* service compostion is the ideal solution for pervasive environments. Service composition has been studied in depth in the context of wired or Internet-based services. We are aware of service discovery and composition platforms like UDDI and HP's eFlow engine (a part of the eSpeak architecture)[4] that 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 are developing 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. Our 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.

VI. Profile-Driven Data Management

The foundation of peer-to-peer data management is fundamental to pervasive environments. It is crucial in enabling mobile, wireless devices to collaborate with peers to achieve higher data availability and currency. An ideal peer-to-peer data management system provides both *read* and *write* modes of transactions to allow every device complete access to local and remote data. For example, Philip's PDA could cache the route information to the nearest hospital and update its neighbors, if any, so that they do not have to go

through the discovery/composition procedures. If, however, Philip's profile prevents arbitrary propagation of available information proactively, the PDA could still respond to queries regarding the location of the nearest hospital. Existing mobile data management solutions are based on the client/server model. Typically, mobile/wireless devices are clients and servers reside on a wired network. The clients are treated as data consumers only, and the main emphasis is often on the development of disconnection management protocols. Such solutions are not flexible or comprehensive enough to deal with peer-to-peer data management. We believe that a peer-to-peer data management system should address additional challenges to common distributed database frameworks that arise in pervasive environments. These include the temporal and spatial variation of data source availability, the use of both implicit and explicit queries and the absence of *a priori* schema translation.

To this end, we have designed and implemented a framework prototype, MoGATU [8], through the use of DAML-S. The primary component of our framework is *InforMa* - a powerful information manager that allows applications to query and obtain responses from their dynamically changing entities. Each device is represented by one *InforMa* and multiple information providers, which can range from local databases to applications generating data dynamically. *InforMa* serves as a mediator among any pair of information providers and consumers, and is responsible for query routing and processing. It provides data routing and forwarding capabilities among mobile units in its vicinity over Bluetooth and Wireless LAN networks. It is responsible for matching requests with possible instances of required answers and/or appropriate information providers. Finally, it caches data that was requested locally as well as data that was advertised by other remote devices. The present form of *InforMa* supports the Least Recently Used (LRU) and preliminary semantic cache replacement algorithms. To replace existing data with new data when the cache is full, the LRU algorithm places the new data in the location of the least recently used items. On the other hand, the semantic algorithm tries to place the new data in the space of the item that is least likely to be used or required again based on the device user's profile. The user profile contains preferences and other information, and serves a base for predicting future data interaction.

Currently, the framework supports only read-mode transactions. Our present task is adding the necessary capabilities to provide write-mode transactions as well. In addition, we plan provide *InforMa* with Prolog-based reasoning capability. In the long term, we envision the framework as a base for building an *ad-hoc* transaction management to allow more sophisticated interactions among peers.

VII. 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 distributed networks such as the Internet and pervasive computing environments require. The main assumptions of existing mechanisms are that the environments possess central control and that their users are all predetermined. Users of these 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], [2]. A user is assigned some generic rights deduced by reasoning over his/her credentials, the security policy 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 [6], [5]. Vigil is a system we developed recently and is designed to provide security and access control in distributed systems, and has been optimized to work in dynamic mobile environments, where most of the clients are handheld devices. We have developed Vigil by combining SPKI and Role Based Access Control [9], [7] 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.

We have developed permissions and prohibitions in Vigil and are looking into the usefulness of obligations and entitlements as additional security mechnisms 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. The trust information is encoded in Prolog; we are planning to use either RDF/RDFS or DAML in addition to Prolog.

VIII. Conclusions

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] 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.

[3] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. *IEEE Proceedings of the 17th Symposium*, 1996.

[4] 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.

[5] 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.

[6] L. Kagal, J. Undercoffer, T. Finin, and A. Joshi. Vigil : Enforcing Security for Pervasive Enviroments. In *Under review*, 2001.

[7] E. C. Lupu and M. Sloman. Towards a Role Based Framework for Distributed Systems Management. *Journal of Networks and Systemss Management, Plenum Press*, 1996.

[8] 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. submitted for conference.

[9] R. S. Sandhu. Role-Based Access Control. In M. Zerkowitz, editor, *Advances in Computers*, volume 48. Academic Press, 1998.