Discovering Services is not Enough[1]

O. Andrei Dragoi and James P. Black
oadragoi@shoshin.uwaterloo.ca, jpblack@uwaterloo.ca

School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Abstract: In the future, mobile users will find themselves in unfamiliar and "chaotic" environments with multiple providers of electronic devices or services, and competing internet-service providers. Users need to discover what services are available, how to apply them to the task at hand, and how to acquire the appropriate rights to use them successfully. The key contribution of this paper is linking service discovery with the acquisition of role-based credentials for services, and showing how the combination can be exploited easily and effectively by users of ubiquitous-computing environments. Our design facilitates flexible authorisation and access control, requires no special-purpose client devices or software, and enables an open market where all participants have a niche that they perceive as bringing value to them. The design has been refined, implemented, and validated as part of a larger prototype implementation, parts of which are described. Middleware, running in a web proxy, transforms web objects by adding specific software tools for the user to discover and invoke the currently relevant services.

Keywords: ubiquitous computing, role-based access control, service discovery, middleware, web proxy

1. Introduction

As the web becomes increasingly woven into all our activities, and our dependence on it grows, opportunities arise to provide value through middleware that can simplify and structure interactions with ubiquitous services. This is particularly true for mobile users who often find themselves in an unfamiliar ubiquitous-computing environment where they need to interact with surrounding services. The services may be as varied as a wall-mounted display, an infospot, a parking meter, a printer, a cash dispenser, a digital camera, a device belonging to another user, or specific software services related to a physical space, such as a patient database in a hospital. Having discovered the services, the users (or their devices, in fact) must also determine how to invoke them, including what credentials might need to be presented during the interaction. To confound the user further, the services, realistically, may be spread across multiple administrative domains. The services can have arbitrary semantics and purposes. Even

[1] This research was supported by Bell University Laboratories at the University of Waterloo, and by the Natural Sciences and Engineering Research Council of Canada.

communicating with them may require discovering and obtaining credentials for an appropriate Internet Service Provider (ISP).

Carefully designed middleware can bring significant value to such "chaotic" scenarios, presenting convenient abstractions and interfaces to users, providers of ubiquitous computing services, ISPs, tool/application developers, and administrators. The middleware should assist in: (1) discovering and interacting with local services in unfamiliar ubiquitous-computing environments; (2) authentication and authorisation; (3) mediating among multiple ISPs, service providers, tool developers, and administrative domains; and (4) coping with limited *a priori* relationships among the players.

We assume conventional web-based interactions in which human users employ standard web browsers (on various devices) to access web pages and services using standard HTTP mechanisms. Mobility and wireless communication are assumed the norm, and the user is likely to visit unfamiliar locations with unknown services available. (These assumptions make familiar environments easy special cases.) Because the user is in an unfamiliar location or situation, we further assume that he or she participates actively in discovering and exploiting the electronic context, with an awareness that "strangers" may need to acquire and exploit various local credentials if they are to use local services successfully. This is similar to asking someone for directions, acquiring a room key in a hotel, using a conference badge to gain access to an event, or arranging payment for a service by presenting a credit card.

We assume the benefits of convenience overcome the risk of potential security breaches. Target settings are, for example, a mall, a hotel, a university, an airport, or a commercial street. The assumption is that breach of trust produces only nuisance, or at most the loss of small amounts of money. Provisioning for non-repudiation and for tracing the chain of trust delegation can be used to prevent abuse of the system, through mechanisms outside the model (e.g., legal action). When security is a larger issue, it is assumed that the owners of a service (employers, businesses, governments, law enforcement agencies, the user, etc.) will choose security over convenience, and employ classic solutions such as virtual private networks (VPNs), explicit access-control lists, and end-to-end encryption.

The security issues related to communication are usually dealt with by mechanisms proven in the "wired" world, such as link-level encryption or the insertion of upper-level encryption layers, like TLS [6]. To assist users, the proxy must be able to observe and modify web objects, which it cannot do if end-to-end encryption is used. Communication is thus secured separately from a user to the proxy and from the proxy to services it invokes on the user's behalf. With these assumptions, a good middleware design should have a number of desirable properties. First, there should be no need for special-purpose code to be preinstalled on user devices, and any code downloaded during web browsing should be minimized. Second, it should support incremental deployment, facilitate the use of existing legacy services, and allow

graceful degradation of functionality when that is not possible. Third, the design should require only local implementations, without any large-scale changes to the internet, and should be able to exploit co-location of users, service providers, and ISPs, when that occurs. Fourth, it should facilitate offerings by multiple, competing, and perhaps small service providers. Fifth, it should enable the creation of markets where users, service providers, ISPs, and software developers all realize value or obtain revenue. Most importantly, it should be intuitive and non-intrusive for users.

The design we propose decomposes the problem into four components: (1) proxies that transform the web traffic to allow the user to interact with the middleware and the contextual services, (2) a software-development framework for ubiquitous computing, (3) service directories and an authorisation framework for the ubiquitous services, and (4) role assertions and service discovery for users. In this paper, we focus on the last two.

Regarding the first two components, the technical key to enabling this electronic discovery and use of contextual information and services is a transforming web proxy that manipulates web requests as they pass through it. Such proxies are commonly used to transcode web content for specific devices, or to interpose access controls between unknown users and a network that requires either authenticated or paying users. Our solution goes beyond conventional transcoding technology, introduced for the web by Brooks *et al.* [2] and, as a paradigm, explored extensively by Zenel [25].

The proxy "decorates" web pages with hotspots. In turn, the user interacts explicitly with the hotspots to get to local context and services. These services are exposed through software tools that facilitate the interaction. The middleware (the proxy) provides mechanisms to invoke the services with appropriate credentials. A hotspot associates tool with the data to which they apply. How it does this depends on the particular hotspot implementation and the data to which it refers. For instance, an icon in the corner of a page associates tools with the whole page. An icon near an HTM: table could associate tools with the data in the table. An image can act as a hotspot for itself such that clicking on the image displays a menu to access contextual services that can be applied to it, such as printing it on a local self-service printer, transcoding it to a more compact form, or showing it on a wall display nearby. This requires cooperation between scripting software added to the page and executed by the user's browser, and software in the proxy server. The tool either implements a service directly, or invokes it as an external contextual service.

The tool abstraction and the software-development framework at the proxy provide support to developers for coding, instantiating, and executing the tools. They include templates for user-interface components, an interface for a method of the tool to indicate whether it should be "published" (included in a hotspot added to a page), support for communication between the applet executing in the client browser and the servlet executing in the proxy, support for standard service-discovery protocols, and support for

maintaining tool state in the applet and caching it at the proxy. Details of the proxy facilities and development framework can be found in a different technical report [7].

In this paper, we describe the combined design for the other two components, which use service directories and an authorization framework in conjunction with role assertions and service discovery for users. The next section provides additional background and describes related research. Section 3 describes the basic features of our design. Section 4 shows how the functionality and the mechanisms proposed here have been integrated with a larger-scope prototype system, called Continuum, and introduces it. Section 5 presents some scenarios to motivate our framework for role and service discovery. Section 6 uses one of the scenarios to describe in detail how the design enables role-based discovery and use of services, while Section 7 presents some conclusions and suggestions for further research.

2. Background and Related Work

Convenient access to ubiquitous services brings a wealth of trust issues, privacy issues (privacy of the attributes of the user and his/her mobile device), and communication issues (end-to-end or mobile-to-base-station secret communication).

In ubiquitous computing, the privacy issues can be more severe than in the wired world [15]. Privacy becomes not only a matter of concealing what data the user is accessing, but also a matter of concealing the user's location or daily routine. Privacy flaws in ubiquitous computing can have serious consequences that can go as far as physically endangering the user. Nevertheless, there is trade-off between convenience and privacy: access to a service may require that personal information be provided before an appropriate role can be acquired or delegated. However, the decision to release that information and use the service rests with the human user, rather than the middleware (the proxy in our case) or the issuer of the role. Similar assumptions are made in WS-Federation and Liberty Alliance [16, 17, 19].

The ubiquitous-computing paradigm also increases the complexity of authorisation and access control, even compared with traditional distributed computing. The area has been the subject of much research in distributed systems. The solutions do not apply readily to ubiquitous computing, due to the inherently ad-hoc, unpredictable, interactions, the user mobility, and the multiple administrative domains. Here, the interactions are more like those in a real society, which is not always the case in "traditional" distributed computing. In society, the notion of trust is subjective, depends on the situation, and evolves, as information becomes available [9]. The authors point out that trust comes from personal observations, from recommendations, and from reputation, a dynamism that is not generally modeled in the current ubiquitous computing environments. Our use of role assertions acquired from other users considers the dynamism.

In distributed systems, a common approach to authorisation is the use of role-based access control (RBAC). In RBAC models, users are assigned roles, and permissions are associated with roles [21]. Policies are defined around each role. While RBAC models work well within the boundaries of one organization, they are too static for ubiquitous computing scenarios. They cannot be used directly when the set of roles is not known in advance and when there is no central organisation to manage the role database, as is the case for ubiquitous computing.

RBAC models push the complexity of access control into the policies that use the roles as inputs. This motivated much research on how to encode, how to evolve, and where to place such access policies, in other words, how to manage trust. In the area of distributed trust management, Blaze *et al.* [1] proposes to replace identity certificates with assertions that a certain key is trusted for a certain purpose. This is combined with local control over the trust relationships through local "trust management engines" (PolicyMakers). Essentially, in an assertion, authorised entities are extending the rules of the access policy of a service. From the point of view of the services that make the authorisation decision, they can be treated simply as instances of the notion of assertions proposed by the PolicyMaker.

The framework proposed in this paper applies the general notion of role(s) of a user to the ubiquitous computing environment. The Centaurus project [14] is another example of a related research effort to adapt RBAC and trust delegation to the ubiquitous computing situation. Centaurus proposes an authorisation framework for a hierarchy of smart spaces. "Security Agents" record delegations and revocations of rights between services. This limits the applicability of the solution across non-cooperating administrative domains, which we assume. In contrast, using tools for issuing and acquiring "role assertions," the framework described in this paper investigates trust delegation without resorting to additional entities. In addition, we also incorporate peer-to-peer communication, natural in wireless environments. In Centaurus, this capability is not exploited, mainly because of the requirement to access the Security Agent for delegation and revocation operations.

In ubiquitous computing, in general, the problem of authorising access was mostly considered in self-contained smart spaces. The Aware Home project proposes the use of environment roles [4], which can be organised in hierarchies to simplify their generation. These environmental roles are automatically determined by the system based on environmental conditions monitored by context widgets [5]. Some of the sample environmental roles used in the Aware Home project (e.g., *weekdays*, *business hours*) seem too far-fetched to qualify as roles although they can influence the user roles, while others are closer to roles as considered in Continuum (e.g., *mom*, *dad*, *parent*). Since checking such roles continuously can be expensive, the Aware Home uses the notion of session for authorisation with an application. Unlike the framework considered in this paper, the Aware Home system, i.e., the user's house, knows the semantics of all these roles and of the rules to generate them. Such a requirement can always be satisfied precisely

because the Aware Home is more or less a closed smart space, with a system administrator. Environment roles local to smart spaces could complement the role assertions as defined in this paper, being just another attribute used to make publication decisions.

The Cerberus authentication module [20], a service for Gaia smart spaces [Roman2003], starts from another simplifying premise that all the principals are known. Then a Prolog inference engine works on authentication predicates and context predicates to reason about allowing access. The authentication predicates are provided by a service that authenticates with variable confidence levels. Somewhat similarly, in the Aura project, users can gain one of three levels of access [24]: maximum, when the user is positively authenticated through a physical device, weak, when the user has been authenticated through more error-prone methods such as video recognition, and unauthorised. Earlier, the Gaia project looked at the opposite problem of how to define access-control policies for an access space [22]. Sampemane *et al.* identifies three kinds of roles. *System* roles are assigned when users are created. From the point of view of the authorisation framework we describe, these are a less useful set, since often the provider will not know all its users in advance nor will it create their identities. *Space* roles are based on how the access control is defined. They most closely resemble the roles in the role-assertions. The *Application* roles allow handcrafted software for the particular smart space to specify access policies. This kind of role is of interest to our framework only to the extent to which applications are accessed as services. Also useful is Gaia's notion [22] that the access rights conferred by a role change with the space mode. Several collaborative modes are proposed for an active space: *Individual*, *GroupSupervised*, and *Collaborative*. In Continuum, some of this work can be leveraged by service providers when they implement the authorisation policies for their services.

The assumptions made by smart spaces simplify things, but the authorisation solutions there do not necessarily scale across different smart spaces. Bussard *et. al* [3] takes a different approach to authorisation in smart spaces. When the user enters a smart space, he obtains a one-time capability from an Access Control Authority (ACA). This capability has similar mathematical properties to electronic cash. He presents it subsequently for a service access. Essentially, if a user misuses his access rights by reusing the capability, he loses a small amount of money previously deposited as a guarantee of fair use. The service provider has to prove to the bank that the access capability has been used multiple times, although this does not require that the ACA be online at the time of the transaction.

We extend the notion of assertion to serve additional purposes, such as hinting at service directories advertising services that might use the assertion for authorisation purposes. Finding service directories that are relevant for a user is a central issue in ubiquitous computing. Tightly coupled environments such as smart spaces have relied on a well-known directory service or a centralized repository of service advertisements (like the event heap in the IRoom project [13]), but when multiple administrative domains

are involved, such approaches are not satisfactory. Some service-discovery protocols specify mechanisms for finding service directories through some form of network broadcast or multicast (Jini [23]), which can also be used to find services directly (e.g., SSDP [11] in UPnP [18], or SLP [12]). However, "close" in networking terms does not always mean physically or logically close. Among discovery protocols, SLP integrates with DHCP [8], so a URL of, say, a directory agent (a service directory in SLP) can be obtained from a DHCP server, at the same time as the IP, gateway, and DNS information. This is not enough when multiple service directories are involved, each relevant to different users, especially since a DHCP server, at network login, cannot necessarily determine what is relevant to the user logging in.

3. Roles, Services, and Discovery

In our design, middleware at the edge of the network fulfills functions related to discovery and authorisation (among other tasks). It assists service providers by brokering service-discovery operations, and assists users in acquiring signed assertions, each of which asserts that the holder is acting in a particular role.

IssuedTo: Pseudonym

Role: *Guest@HotelName.com*

GrantedBy: Clerk123@HotelName.com
Valid: 12 days
Granted: Jan 23,2003,16:53:43GMT

ServiceDirectories:
slp://Services.HotelName.com,
jini://Services.HotelPartner.com

Digest: *Signature of Clerk123@HotelName.com*

Figure 1. Example of a "Hotel Guest" assertion

There are several prerequisites to enabling role assertions as the basis for service discovery, authentication, and authorisation.

(1) A service provider (SP) must arrange for its services to be listed in one or more service directories, and for role assertions to be issued that bind a user to a role. The assertions also list directories of services that may accept the role. The roles and their implied semantics are a convention of the services accepting them. The signer(s) of the assertions could be the service provider itself, or any business partner (or select user) whose signature is acceptable. Figure 1 is an example of an assertion issued and signed by a hotel clerk, confirming that a certain user is a hotel guest for a certain period.

(2) The SP must ensure that the middleware can publish end-user tools for its services, either directly (by providing and disseminating code) or indirectly (by adhering to standards). As a trivial example, a

general printing tool published by the proxy could provide the end-user interface and manage the interactions with a printer that follows a standard protocol and it is advertised in a standard service directory.

(3) The user acquires a role assertion for the service. The assertion may be acquired independently by some interaction with a signing authority, or through an interaction mediated by tools published by the proxy. (This assumes the user's proxy is the same as the signer's proxy, or that the two proxies co-operate.) The "signing authority" can be some service or another user.

(4) The user finds out about a service because a tool has been published. The tool execution invokes the service, supplying an acceptable role assertion along with other attributes of the user and parameters, such as the web object for which the tool was published. The service may consider the role assertion sufficient, or may engage in some type of validation of the assertion and attributes. Attributes may be acquired in a variety of ways, and cached in a soft session at the proxy.

The framework described above is flexible enough to support a variety of trust relationships and implementation choices.

Roles and associated service directories can be implemented independently by ISPs and/or SPs. If a particular role (or set of roles) has access to a number of different services, it makes sense to group them in one or more directories. A directory can be hosted by an individual service provider for private roles it creates. Alternatively, the SP could rely on some third party to host the directory, presumably as part of some larger grouping of roles and related services. The third party in question could in fact be the ISP of the user, which might provide advantages in that tools for the services could be published by default, without the need for extra mechanisms related to assertion exchange. When this is not the case, the middleware will find references to those service directories in the role assertions.

Service providers can allow others to sign role assertions, such as business partners, ISPs, or other SPs, and yet maintain control over authentication and authorisation. For an existing service, the service provider may wish to rely on some challenge-response mechanism with the user to verify the assertion. Alternatively, the tool vendor could implement some custom verification with the SP prior to invocation of the service. A third possibility is that the service interface implemented in the tool involves some standard authentication and authorisation negotiation with a purpose-written or standard-based service.

Users can acquire and manage role assertions independently of SPs and ISPs. This enables free exchanges of assertions with other users. Examples are users exchanging role assertions in electronic mail or beaming them between handhelds, storing them on some medium for future use with a different end-user device, and storing them in persistent storage provided by some service trusted by the user (such as the user's personal computer). Since assertions include information about service directories, they may

often be used without intrusive verification steps involving the user, especially if few services require non-trivial authentication.

ISPs can add value through service aggregation and tools for exchanging role assertions. Because of the multiplicity of communication media available in a chaotic ubiquitous-computing scenario, it may not be the case that the user's ISP is in fact physically close to or actually aware of the user's location. Even so, by querying the service directories listed in role assertions that the user has acquired independently, the proxy obtains enough context about the user to publish valuable tools. Of course, closer coupling between the proxy and a physical location may provide better functionality to the user, in the form of a more appropriate set of tools to access local services and acquire local roles. In fact, even just having the proxy know where certain roles are relevant could yield an improved set of tools. For instance, a role assertion could also indicate a physical area in which the role was valid.

4. Continuum

We have implemented a prototype called Continuum to test our ideas for middleware-assisted chaotic ubiquitous-computing systems. The prototype includes tools and a tool publication framework that decorates web objects with hotspots offering access to the published tools. It offers the ability to use external services (such as X-window displays or LPR/LPD printers), which are discovered through standard discovery protocols, and includes the general authentication, authorisation, and service-discovery mechanisms described in this paper. We have implemented a number of tools, including all those pictured in Figure 2, and tools for the exchange of role assertions. We are in the process of prototyping services that make use of role assertions. Tool applets run on a number of standard browsers, and require JavaScript support. We have exercised some care to ensure they are portable across several major browsers for mobile devices.

Figure 2 shows Continuum in operation. The left side of the figure was captured from the screen of a conventional laptop. Continuum's features are also available from PDAs, like the Sharp Zaurus in the right-hand photo where a user is performing the same task as on the left. (We had no tools to capture the PDA screen.) By default, Continuum transcodes images to small white GIFs, with the assumption that bandwidth is an important commodity and that the screen may be too small to display the image conveniently any- way. The user has clicked on the blank image, to show a menu of five tools. The menu (and the associated JavaScript to implement it) was added dynamically by Continuum as part of a process we call "decoration." The tools are a user-interface notion. We call the software abstraction behind each tool a frame, because it unites several functional elements, like the frame of an exhibit case. The frames of the figure provide access to printing on nearby printers and to large-screen displays near the user. The "home node" tool saves an image or other web object to the user's computer or home node. The

bookmark tool saves only a link to it. The transcoding tool controls the fidelity of the object on the screen of the mobile.

Frames are the core technical objects manipulated by Continuum. Each consists of an applet, or scripting code running in the client browser, and a servlet running inside Continuum. The applet is responsible for the user interface and for communication with the servlet. The servlet, in turn, mediates access to external services, or provides services to the user itself. The Continuum runtime environment provides mechanisms for discovering and using built-in and external services, and for constructing the frame user interface. Continuum itself is responsible for interacting with candidate frames to determine which of them should be published. Continuum publishes frames for each web object requested by a client browser, including the entire page (a small hotspot in the top left corner).

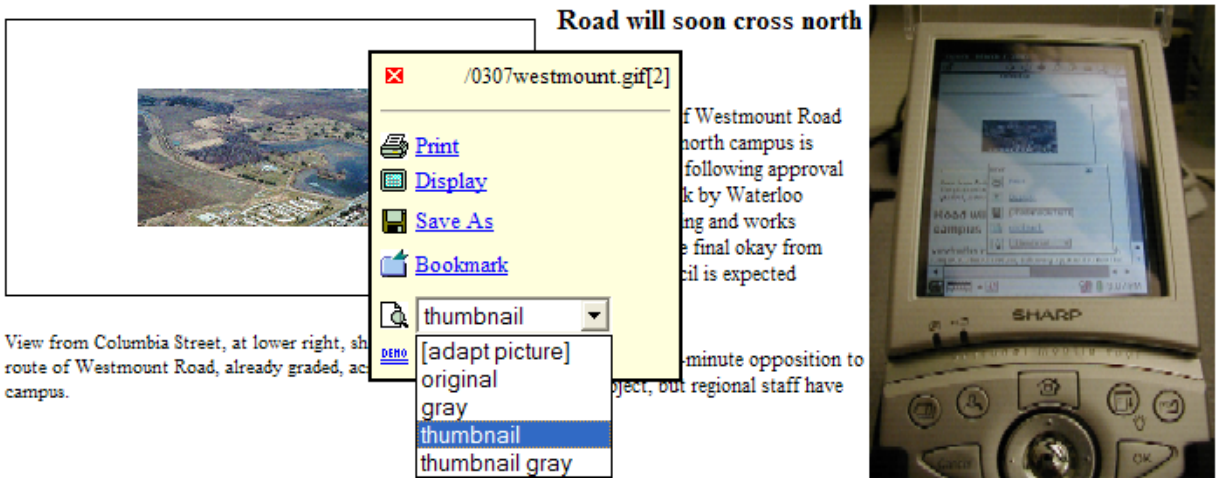

Figure 2. Using Continuum

For example, when the *Print* frame is invoked, the servlet uses Continuum facilities to discover (standard) print services available near the user. The applet presents a list of one or more actual printers to the user, whose choice is then returned to the servlet. The list was obtained from a combination of service directories that Continuum can access directly (if co-located with the user), and those indicated in role assertions provided by the user.

The servlet then invokes *Print* on a specific printer. The precise semantics of what is printed are chosen by the frame developer. For example, one might choose to print a high-quality version of an image from the Continuum cache, rather than the thumb- nail image presented by the user's browser. Another example of a choice of semantics involves the moment, When service discovery is executed. There is no guarantee that the user will ever invoke a published tool. At the time of invocation, the context might have

changed from that of the moment of publication. Continuum permits the frame developer to invoke discovery at either time.

Continuum views all the external entities with which it interacts as "services," whether they be services running on user devices (client services), legacy services wrapped by servlets (the *Print* frame wraps an LPR/LPD service), Continuum-aware services, or service directories. Services exchange "attributes" with semantics that can be opaque to Continuum. Examples of attributes might be location, user credentials, role assertions, device characteristics, and service characteristics. We assume users find it convenient to have a fixed "home" location where private information can be stored securely and permanently; this might be a server belonging to the user, or a service purchased from a third party. Our proposal for role-based access control relies on the ability of the user and Continuum to exploit the home node service.

In implementing our prototype, we have incorporated a number of common practices and technologies. Continuum operates as a conventional web proxy server. It could easily be modified to intercept web traffic transparently. We use open-source transcoding software to provide a number of standard services to frame developers (and hence to users). Since the user may not be using private, personal devices, we assume a secure "home" service where he or she can save information of interest, and use WebDAV [10] to manipulate it. Our prototype uses the Jini [23] and SLP [12] protocols for service discovery. Extending this to the web services framework (SOAP/WSDL/UDDI standards) poses no conceptual difficulties.

We have extended the Continuum prototype to include our approach to combined service and role discovery. In the next section, we illustrate this approach with several scenarios. Mostly, we ignore the normal web interactions users might have, to concentrate on those involving direct interaction with the proxy itself or with tools that it publishes. We assume Continuum has the ability to interpose itself as required, to take action before returning a web page. This is a standard feature of proxy servers. For instance, those in hotels require an interaction with the user to authorise a room charge before allowing user traffic onto the internet.

5. Scenarios

We present three scenarios, which can all be supported simultaneously by a single proxy server run by the same ISP.

The first concerns a small hotel in a larger building or convention center that wishes to provide a rich ubiquitous-computing environment to its guests. Basic wireless connectivity and a proxy are provided by a local ISP. Local services include a printer for hotel guests in the lobby, and perhaps others like a vending machine or a cash dispenser. An incoming guest first needs to establish a proxy session through some use of the network, which results in initial publication of tools relevant in the local context. The guest may or may not have business relationships with individual service providers or the specific ISP. As

the guest checks into the hotel, the registration clerk and the guest then arrange for an electronic role assertion to be acquired by the browser on the guest's PDA. (More precisely, the role-assertion attribute will be stored by a frame applet as part of a cookie, and may be stored at the user's home node.)

Recall the role assertion of Figure 1. In addition to the basic assertion, it also gives two service directories where the guest can find services accepting the role: an SLP directory maintained by the hotel and a Jini directory maintained by a local third party. Later, the proxy can detect the role assertions in outbound applet requests, and use the role and directory information to publish frames for services that accept the role. In this way, those services have no need to create or maintain a relationship with the guest. When the guest tries to access one of the services, the service need only accept the role granted by the hotel. A service may also apply whatever checks or validations it wishes to verify the authenticity of the user. We provide more detail in Section 6 on the protocol exchanges among the hotel clerk, the guest, the proxy, and services.

What does a service provider such as the hotel need to do? Since it wishes to create and sign its own role assertions, it must negotiate role names and their semantics with local service providers, possibly including one or more ISPs whose transmission facilities are available inside the hotel (e.g., 802.11b wireless, cellular wireless). If the hotel also wishes to make services available to holders of its roles, it would need to advertise those services in its own service directory, or in some public service directory maintained, say, by the ISP. If the hotel has some special services it wishes to make available, it may also wish to develop custom frames and have the code disseminated to ISPs, which in turn can publish them for the appropriate clients.

As a second scenario, consider the case of a guest visiting a university. In principle, members of the public are not given access to the university network. However, authorised university faculty members are allowed to create role assertions for their academic visitors. These assertions can be presented in response to a challenge for network or other services. In this case, the network itself cannot be used to transfer the role assertion from the professor to the visitor, and so some peer-to-peer communication between the professor's PDA and the visitor's is required. After using a Continuum frame to create the role assertion, the professor must transfer it to the visitor's PDA over, say, a Bluetooth network, or by direct transfer through a USB memory key. Other protocols to achieve the transfer can also be imagined, such as ones involving a Continuum server outside the university firewall.

A slightly different scenario involves a shopping mall, where basic connectivity is provided by one or several well-known ISPs (not necessarily affiliated with the mall), with whom many shoppers have a previous business and trust relationship. The various services provided by the mall merchants are registered by them in service directories maintained by the ISPs, the mall management, or other parties.

For example, to benefit from the trust relationship between the ISP and its subscribers, mall merchants would agree to accept role assertions signed by the ISP. The ISP might base these assertions on its technical information about the user's location.

In general, when non-standard services need to be deployed, the ISPs would not have the corresponding frames. In such a case, or when the particular role semantics are used in the publication process (to improve the user experience), the SPs and ISPs must come to more or less conventional business agreements related to providing proxy service in a locale. As the middleware provider, an ISP is expected to install, configure, and maintain the proxy service. Although service directories may be obtained from role assertions presented by clients, an ISP may also choose to configure external services and/or service directories to be used in default tool publication. Of its own accord, it may decide to provide a number of native services to users, including built-in ones. It may also implement a number of frames and services, potentially including role semantics and directories, or purchase frame software from independent vendors. Finally, an ISP will presumably take a lead role in negotiating the participation of third party SPs.

In general, service providers will have to determine what roles their services will accept, and whether there are existing frames available that can be used to access those services. They could decide to accept role assertions issued by other parties. They might also inform authoritative users of their abilities to issue role assertions. Continuum includes a number of frames, and we believe the frame concept provides a convenient development abstraction for enabling role-based services. In the next section, we provide some technical detail on how role assertion mechanisms have been implemented in Continuum.

6. Implementing Role Assertions

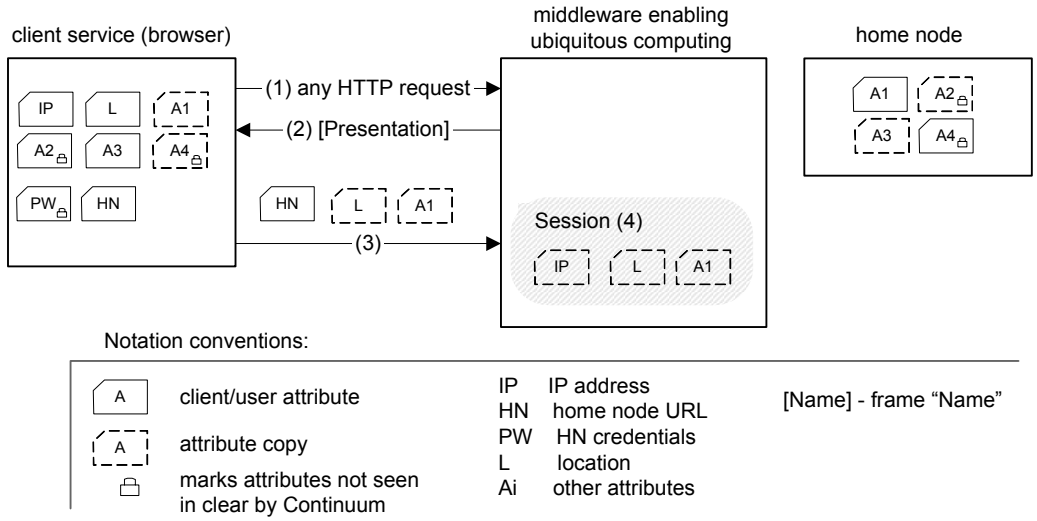

Figure 3. Establishing a Continuum session

Solving the challenge of service and role discovery with multiple administrative domains involves two key concepts. First, a particular role assertion must be created and communicated to an interested user. Subsequently, the user relies on the assertions and other attributes to secure access to a service. In this section, we use the hotel example to illustrate in more detail how the *Guest* role assertion is acquired and how it is used in subsequent services invocations.

Figure 3, on the previous page, sketches the initial login procedure. The proxy maintains a soft-state session. In the interaction (1), the published tool is noted with brackets over the interaction. The subsequent interactions occur when the user chooses the tool. Some attributes are not to be seen in clear by Continuum and are marked as such. Also, note the distinction between the main copy and the other copies of an attribute. We use the same conventions in subsequent figures.

In step (1), Continuum responds to any HTTP request by publishing a tool (the *Presentation* frame) that requests information from the user. A more sophisticated frame might obtain some of the attributes requested through a simple HTTP form. In a more sophisticated frame, some of the attributes requested can be obtained through introspective JavaScript code. The response (3) from the user establishes a session attribute cache, used by tool-publication heuristics. Attributes available from the client include its IP address, and perhaps others such as the user's public key or identity, the device location, and device type. We assume the user wishes to disclose the URL of the home node to the proxy, to enable the home-node service to interact with it. After session establishment (4), Continuum would normally transmit the result of the original HTTP request (1).

A frame for issuing or acquiring a role assertion appears in *Continuum Tools*, a set of frames published by default that implements common, object-independent services. See Figure 4, where the *AcquireRA* frame is published to the hotel guest in interaction (2). Similarly, the *IssueRA* frame is published to the hotel clerk in (4). In the interactions labelled (5), the hotel guest chooses to acquire a role, following a verbal indication from the clerk that he or she might wish to do so. In turn, this causes a dual interaction between Continuum and the clerk to create the assertion. The actual exchange of the assertion requires a three-way handshake, so that Continuum correctly associates the guest's session with that of the clerk. The role assertion (marked with an *R* in Figure 4) is returned to the guest, and may be stored at the home node through another three-way interaction hinted at in the figure. Continuum retains a cached copy of the role assertion to aid publication heuristics. For this example, we assume both clerk and guest use the same proxy, but the example can be extended to using different ones.

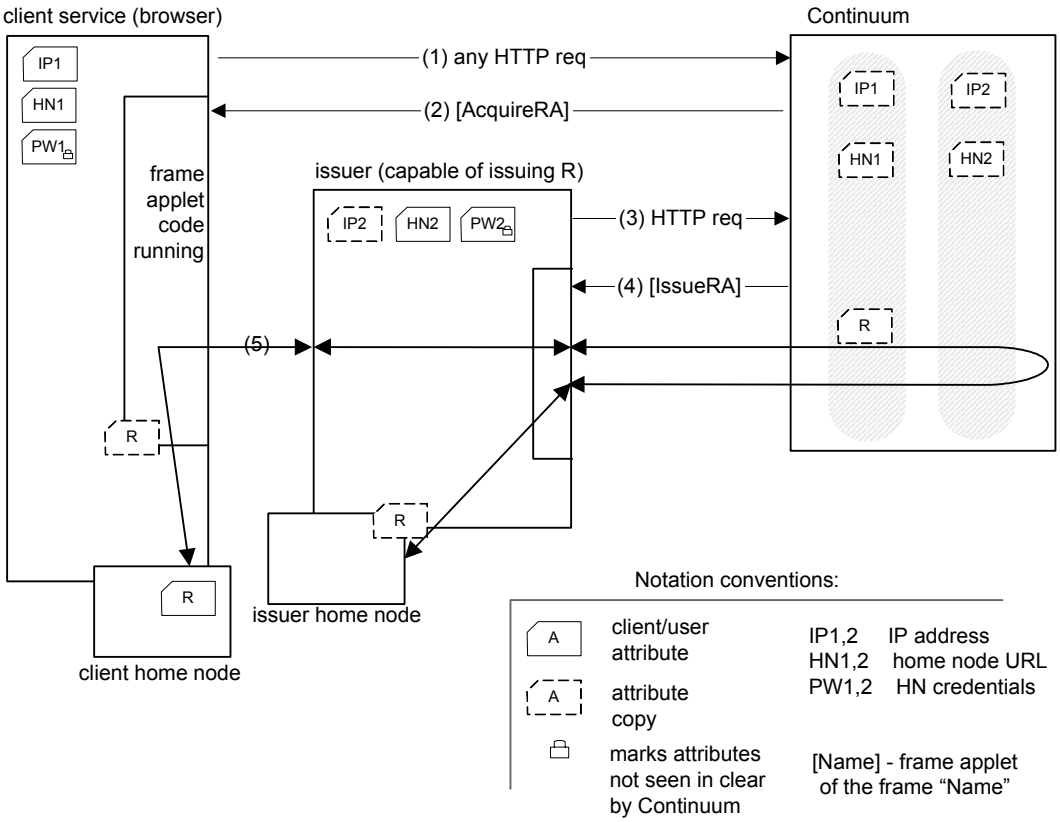

Figure 4. Acquiring a role assertion

Figure 5, on the next page, attempts to give a sense of what the guest and clerk see, in the current implementation, as they exchange the role assertion. The numbers beside each image indicate the sequence in which the two users' actions occur. The upper part of the figure refers to the issuer, the bottom to the recipient. They begin by choosing the "Continuum Tools" hotspot, and then the "Receive Role" or "Issue Role," the labels presented by the *AcquireRA* and *IssueRA* frame applets respectively. We assume each participant generates a simple password and communicates it to the other in (2)-(5). Continuum uses the pair of passwords to match the two users' sessions before their identities are revealed to each other. The *IssueRA* frame then uses the clerk's home node to generate the role assertion (assuming this is too onerous to accomplish on the clerk's PDA) in (6)-(8). Step (7) confirms the URL of the home with the clerk (obtaining the URL from the attribute pool). This might be followed immediately by a challenge for credentials, coming from the home node. Then, the assertion is received and perhaps stored at the guest's home node in (9)-(10), possibly after a challenge for credentials. The procedure ends after confirmations are received in (11)-(12).

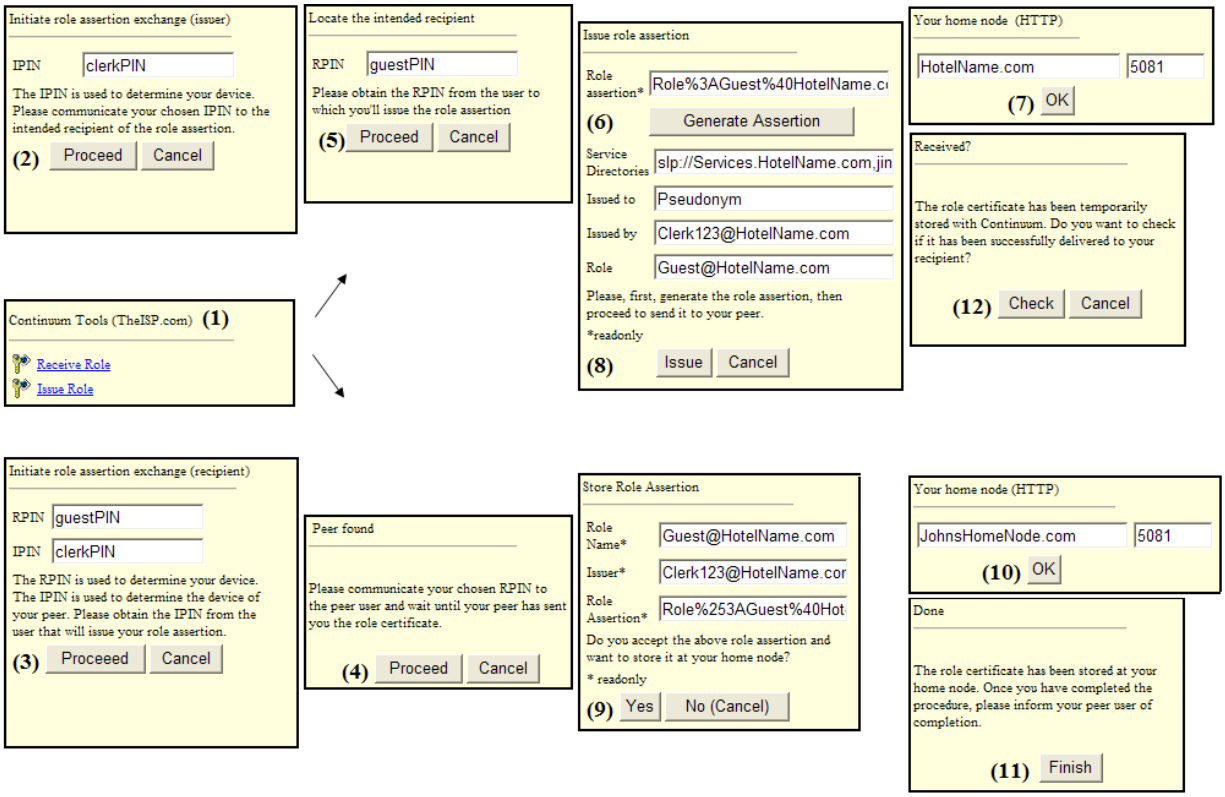

Figure 5. Exchanging a role assertion

Suppose now the user wishes to print a web object on the printer in the hotel lobby, and that the printer is owned and operated by a local copy center. Assume the hotel and copy center have arranged that the hotel guests may use the printer without hindrance, upon successful presentation of the *Guest* role. Whether the copy center charges the hotel for guest usage and whether the charge appears on the guest's bill is a separate issue, left to the discretion of the hotel. Figure 6, on the next page, describes the general steps in using a service.

The user chooses a print service in (3), providing attributes such as the hotel-guest role assertion, and a reference to the object to be printed. The service was discovered by the built-in *Print* frame, which interrogated the directories listed in the role assertion. (The assertions and URLs to all service directories that Continuum associates with a particular user can be cached in the user session.) The semantics of the frame are that it prints the best available version of an object, typically the original web object, rather than the transcoded one rendered by the user's browser. The frame servlet requests access to the printing service in (4), and is in turn challenged for attributes, in particular, for an appropriate role assertion. The service may determine that further attributes or validation are required, such as some assurance that the end user is indeed in possession of the private key corresponding to the pseudonym of the role-assertion

holder in steps (7-10). Once authorisation is obtained (11), the service is invoked on the desired version of the data object indicated by the user in (3). The object sent in (12) is converted to a format required by the printer, such as PostScript. Note the distinction between the references to the main copy of an object (or a copy identical to the main copy), in (3), and references to a transcoded copy (or a transcoded copy), in (12)

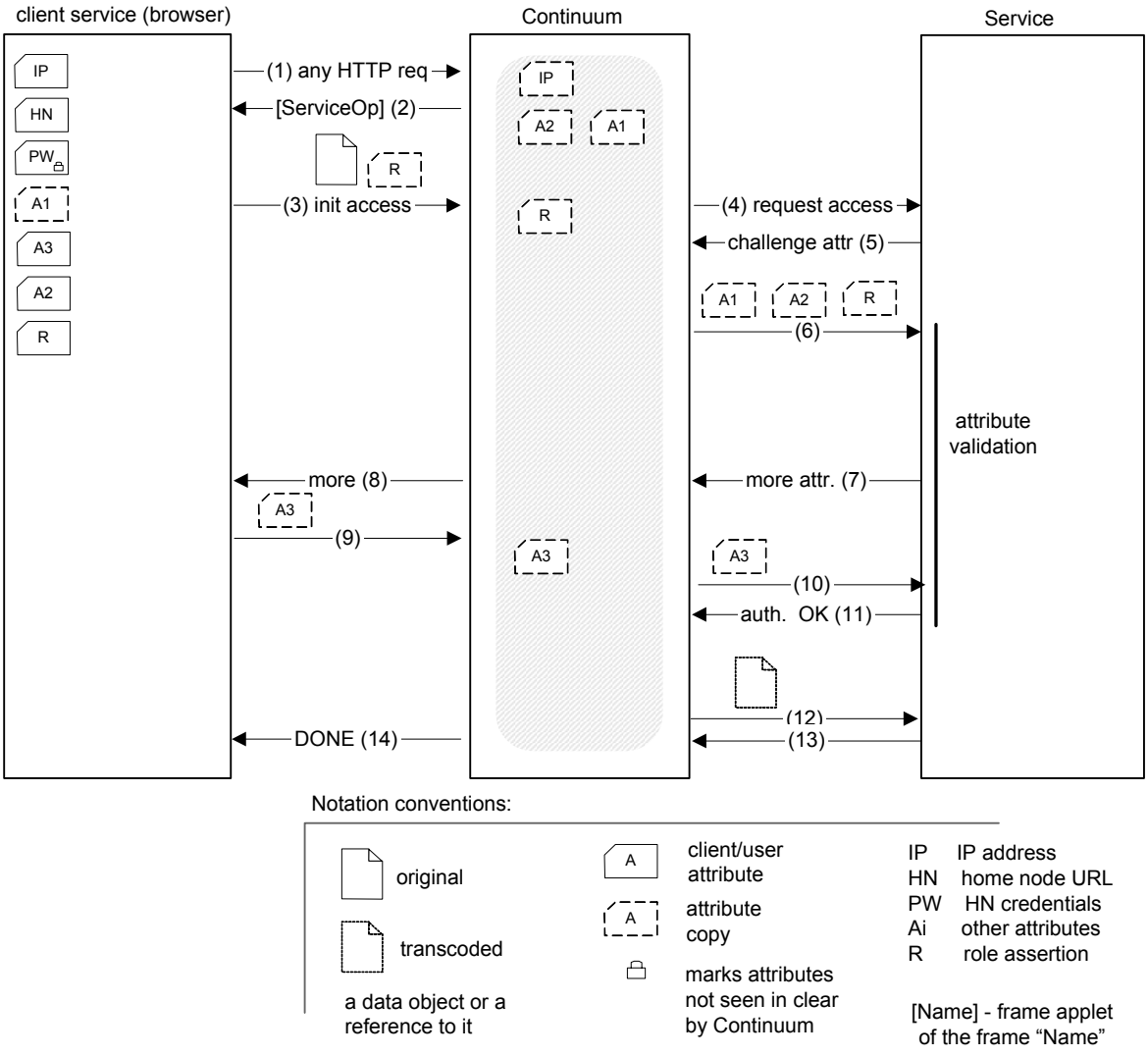

Figure 6. Accessing a service

In general, services may make their own decisions on the validity of attributes. For example, a vending machine might require some indication that a requesting user is physically adjacent before completing a transaction. The indication might involve some automatic location technology or physical interaction with the user. Other services might accept role assertions or other attributes at face value without any interactive verification. Continuum imposes no particular view on how services evaluate or validate

attributes. Currently, Continuum sees all challenges and responses for attribute validation. A delicate problem is the lack of user-interface mechanisms to convey easily to the user which server is the target of an HTTP request. The address bar does not show the queries issued by JavaScript or from HTML frames, while the status bar information is quickly overwritten. Consequently, the user cannot know reliably when he or she is interacting directly and in cipher text with a service, or when the interaction passes in the clear through Continuum. A first step in current browsers is the use of a visually different dialog box for password challenges, indicating the server.

7. Conclusions and Future Work

Our design has the desirable properties described in Sect. 1. It enables flexible authorisation and access control in chaotic ubiquitous-computing environments, while requiring no special-purpose client devices or software, and enabling an open "market" where all participants have a niche that they perceive as bringing value to them. SPs and ISPs can provide competing services for profit with limited trust and cooperation, and users have convenient tools to assist them in discovering and exploiting the unfamiliar (or familiar) environment in which they find themselves.

Specifically related to service access, we will explore attribute validation further, and in particular, location validation. We plan to investigate a service run by wireless ISPs that would sign role assertions based on a user's location. We would also like to prototype peer-to-peer, wireless exchange of role assertions, and assess how well this would work over common wireless technologies like Bluetooth. We plan to investigate the use of direct attribute challenges from services, bypassing the proxy. Additionally, we plan to explore heuristics for tool publication, how they can be tuned based on roles whose semantics the proxy might understand, and to experiment more with frames that search for external services proactively, either at publication time or at execution time.

We also plan to study further the relations between ISPs and SPs with respect to disseminating information about role semantics. This includes investigation of what dynamic mechanisms are needed to allow a proxy server to upgrade itself with new frames periodically, on demand, or driven by selected service providers.

In more general terms, we plan to prototype frames for additional services, including services that generate data objects (such as a digital camera), and tools with different user interfaces (voice, physical buttons on the device).

Our approach to role acquisition allows users to acquire them intuitively, and "by word of mouth," in the sense of interacting with local users and authorities who can sign role assertions. In turn, the roles facilitate the authorised use of services, and provide pointers to local directories of services likely to accept the roles. In general, a proxy needs no understanding of the semantics of particular roles. However,

conventions for naming roles might allow the proxy or service providers to further automate and enhance the user experience.

The key contribution of this paper is linking service discovery with role-based access control, and showing how the combination can be presented effectively to users of ubiquitous-computing environments, whether as signers or as recipients. We show how the approach allows proxy-type middleware to facilitate user interaction with the "chaotic" environment. Our design has been refined through the implementation of the Continuum prototype. We assume users are willing and able to acquire various roles actively, since they enable use of services in an unfamiliar environment. Moreover, we assume they will find the addition of hotspots to web pages an unobtrusive mechanism for accessing services relevant to the current data object and context. This mechanism provides value to service providers, internet service providers, and, of course, the end user.

8. References

[1] M. Blaze, J. Feigenbaum and J. Lacy, "Decentralized trust management", in *IEEE Symposium on Research in Security and Privacy*, pp. 164–173, Oakland, CA, USA, May 1996.

[2] C. Brooks, M.S. Mazer, S. Meeks and J. Miller, "Application-specific proxy servers as HTTP stream transducers", *World Wide Web Journal*, vol. 1, n. 1, 1996.

[3] L. Bussard and R. Molva, "One-time capabilities for authorizations without trust", in *2nd IEEE International Conference on Pervasive Computing and Communications (PERCOM '04)*, pp. 351–355, Orlando, Florida, USA, March 2004.

[4] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad and G. D. Abowd, "Securing context-aware applications using environment roles", in *6th ACM Symposium on Access control models and technologies*, pp. 10–20, Chantilly, Virginia, USA, 2001.

[5] A. K. Dey, *Providing architectural support for building context-aware applications*, PhD thesis, Georgia Institute of Technology, December 2000.

[6] T. Dierks, "The TLS protocol version 1.0", RFC2246, IETF, January 1999.

[7] O. A. Dragoi and J. P. Black, "Enabling chaotic ubiquitous computing", Technical Report CS-2004-35, School of Computer Science, University of Waterloo, August 2004.

[8] R. Droms, "Dynamic host configuration protocol", RFC2131, Network Working Group, IETF, March 1997.

[9] C. English, P. Nixon, S. Terzis, A. McGettrick and H. Lowe, "Dynamic trust models for ubiquitous computing", in *Workshop on Security in Ubiquitous Computing (with UBICOMP '02)*, Göteborg, Sweden, September-October 2002.

[10] Y. Goland, E. Whitehead, A. Faizi, S. Carter and D. Jensen, "HTTP extensions for distributed authoring: WebDAV", RFC2518, Network Working Group, IETF, February 1999.

[11] Y. Y. Goland, "Simple service discovery protocol/1.0, operating without an arbiter", Internet draft, work in progress, expires April 2000, IETF, October 1999.

[12] E. Guttman, C. Perkins, J. Veizades and M. Day, "Service location protocol, version 2", RFC2608, Network Working Group, IETF, June 1999.

[13] B. Johanson, A, Fox and T. Winograd, "The interactive workspaces project: Experiences with ubiquitous computing rooms", *IEEE Pervasive Computing Magazine*, vol. 1, n. 2, pp. 67–74, April-June 2002.

[14] L. Kagal, V. Korolev, H. Chen, A. Joshi and T. Finin, "Centaurus: A framework for intelligent services in a mobile environment", in *International Workshop on Smart Appliances and Wearable Computing*, Phoenix, AZ, April 2001.

[15] M. Langheinrich, "Privacy by design: Principles of privacy-aware ubiquitous systems", in *Ubicomp 2001*, pp. 273–291, Atlanta, GA, USA, September 2001.

[16] Liberty Alliance Project, "Introduction to the Liberty Alliance identity architecture, revision 1.0", White paper, http://www.projectliberty.org/, March 2003.

[17] Liberty Alliance Project, "Liberty Alliance and WS-Federation: A comparative overview", Technical white paper, http://www.projectliberty.org/, October 2003.

[18] Microsoft Corporation, "Understanding UPnP: A white paper", White paper, http://www.upnp.org/, June 2000.

[19] Microsoft Corporation International Business Machines Corporation, "Federation of identities in a Web Services world", White paper, http://www.ibm.com/, July 2003.

[20] J. Al-Muhtadi, R. H. Campbell, A. Kapadia, D. Mickunas and S. Yi, "Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments", in *International Conference of Distributed Computing Systems (ICDCS '02)*, pp. 65–74, Vienna, Austria, July 2002.

[21] R. Sandhu, E. Coyne, H. Feinstein and C. Youman, "Role-based access-control models", *IEEE Computer*, vol. 29, n. 2, pp. 38–47, February 1996.

[22] G. Sampemane, P. Naldurg and R. H. Campbell, "Access control for active spaces", in *Annual Computer Security Applications Conference (ACSA '02)*, pp. 343–352, Las Vegas, NV, USA, December 2002.

[23] Sun Microsystems, "JINI technology architectural overview", White paper, http://www.sun.com/, January 1999.

[24] S. M. Thayer and P. Steenkiste, "An Architecture for the Integration of Physical and Informational Spaces", *IEEE Personal and Ubiquitous Computing*, vol. 7, n. 2, pp. 82–90, July 2003.

[25] B. Zenel, *A proxy based filtering mechanism for the mobile environment*, PhD thesis, Department of Computer Science, Columbia University, February 1998.