Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments

Adrian Friday, Nigel Davies and Elaine Catterall

Distributed Multimedia Research Group,
Computing Department, Lancaster University,
Bailrigg, Lancaster,
LA1 4YR, UK
+44 1524 593807

{adrian,nigel,elaine}@comp.lancs.ac.uk

ABSTRACT

Future computing environments will consist of a wide range of network based appliances, applications and services interconnected using both wired and wireless networks. In order to encourage the development of applications in such environments and remove the need for complex administration and configuration tasks, researchers have recently developed a range of service discovery and interaction platforms. Examples of such platforms include SLP, HAVi, UPnP and Jini. While these platforms share a number of common attributes, they each have distinguishing features and hence future networked environments are likely to present developers with a heterogeneous environment composed of multiple specialised support platforms. However, careful analysis of these platforms reveals shortcomings that we believe will inhibit the development of applications that exploit service rich environments. In this paper we discuss these shortcomings and propose a new unifying architecture that brings together the advantages of current service discovery and interaction technologies and provides a new API that we consider to be better suited to the development of service based applications. This work is specifically targeted towards mobile environments, where applications will be required to interact with a wide range of services and devices with minimal user intervention.

Keywords

Mobile and ubiquitous computing, service discovery, service interaction, middleware.

1. INTRODUCTION

Current networked environments are populated with a diverse set of devices, services and computational entities. Enabling these components to work together harmoniously and allowing users and applications to interact with them without considerable administrative and configuration overhead poses a number of logistical and technical challenges. As a result, there has recently been considerable research into service location and device interaction technologies, including SLP [8], HAVi [9], UPnP [17] and Jini [19].

The key function of such service location and device interaction technologies is to allow users and applications to deploy, discover and interact with the services provided by devices and software components on the network. This interaction is required to occur without the users, the applications, or the service providers needing detailed knowledge of the local network configuration. While these technologies were originally developed for zero-configuration networks (e.g. those aimed at installation in the home environment) researchers have recently begun to investigate how they can be used to support aspects of mobile and ubiquitous computing (e.g. enabling a user in an unfamiliar network to discover nearby printers from their laptop) [1,10].

In the near future, as service technologies are deployed in mainstream operating systems and networked 'Internet' appliances, the number and heterogeneity of services and devices is set to expand rapidly. Furthermore, new personal mobile devices (such as PDAs and 3G telephones) and the emergence of near-ubiquitous communications infrastructures based on wired and wireless networks will offer yet richer ways of interacting with these environments, services and devices.

We believe that these future service environments will be characterised by a heterogeneous mix of services and technologies. In particular, devices, applications and users will need to interact with multiple, potentially specialised, service location and device interaction technologies. Based on an analysis of existing approaches to service discovery and interaction, we have identified a number of shortcomings that we believe will seriously impact on the scalability, efficiency, utility and usability of current techniques in emerging service environments.

In this paper we present an initial design for a new platform that combines the strengths of existing service discovery and interaction techniques, yet offers a unifying API for application developers and supports a number of additional data-engineering oriented services that provide a framework for addressing the shortcomings of existing approaches. The platform has been specifically designed to include support for the development of

MobiDE 2001 USA

ubiquitous computing applications operating on mobile devices.

In section 2 we provide a brief overview of the state-of-the-art in service location and device interaction technologies. Section 3 contains our analysis of these technologies and identifies requirements for an improved service location and interaction architecture. Section 4 presents the initial design for our platform and section 5 describes a number of scenarios that illustrate the advantages of our platform in mobile environments. Finally, section 6 contains our conclusions and plans for future work.

2. SURVEY OF EXISTING TECHNOLOGIES

The desirability of zero-configuration networks, whereby devices can join the network and "just work", has driven the development of significant numbers of device interaction technologies by both companies and commercial consortia, e.g. Jini [19], HAVi [9], SLP [8], UPnP [17], Salutation [18], Cooltown [14] and academic research projects, e.g. [10]. We begin by briefly describing the key architectural features of one of the most recently developed commercial systems, i.e. UPnP.

"Universal Plug and Play (UPnP) is an initiative to bring easy-to-use, flexible, standards-based connectivity to consumer networks, whether in the home, in a small business, or attached to the global Internet" [16]. From a technology perspective, UPnP is a suite of protocols and system services for device discovery and control in small to medium size IP networks. In home networking installations, where complex network management is not possible, the developers of UPnP anticipate that AutoIP [20] will be used to help configure hosts until such time as IPv6 is deployed on a wide scale.

In UPnP environments the Simple Service Discovery Protocol (SSDP) [6] runs on top of this base IP layer to provide a means of discovering devices on the network. SSDP allows devices to advertise services using datagrams sent using IP multicast. These advertisements contain a service type and a URL for the service being advertised. Clients interested in accessing services can either wait for announcements or can multicast a search request that forces all devices on the network to send service advertisements. Once a client has obtained a URL for a service it can retrieve from the device an XML description of the service being offered. This description will include, among a host of other pieces of information, an additional URL that can be used to access a web page representing the user interface for the device and a list of state variables associated with the service. In more detail, in UPnP devices are modelled as objects with an associated state table (*c.f.* properties). This state table provides an external representation of the object's internal state. Clients can make changes to this state table that causes the associated object to invoke operations to achieve a corresponding change in its internal state. For example, an object representing a camera might have a state table that included a variable `auto-focus` with possible values of `on` and `off`. Clients that wished to change the camera's auto-focus mode would simply need to change the value of the `auto-focus` variable. The architecture requires that objects generate events whenever their state changes and that clients register for these events in order to ensure all views on the device are consistent (for example, to ensure consistency between the camera's front panel and a remote application also controlling the camera).

It is worth highlighting at this point that UPnP compatibility is defined in terms of the on-wire format used for messages and the architecture is OS and language independent.

Many of the service discovery and interaction technologies operate in a similar way, and have a number of features in common with UPnP. For example, Jini, SLP and [10] all use IP multicast for device discovery. In addition, most use some form of directory service to optimise device discovery in large networks (including HAVi which, while not being based on IP, shares many architectural similarities with IP based solutions). However, despite the logical similarity of many of the technologies, the implementations are diverse, and therefore incompatible. Each technology has its own model for key features such as device and service description, scoping of service advertisements, interaction with devices and notification of events. One of the reasons for this heterogeneity is that the technologies were constructed with different domains in mind, and in the spirit of experimentation. For example, SLP is a very scalable discovery protocol, intended to serve enterprise networks; UPnP targets home and small office computing environments; while HAVi was designed to enable interoperability in home AV networks and supports multimedia streaming and service reservation. An in-depth comparison of these architectures can be found in [2] and [15].

3. ANALYSIS

Existing approaches to service location and device interaction provide powerful infrastructures on which to build distributed service based applications. However, with careful analysis it can be seen that as the usage of such technologies increases a number of issues arise. In particular, we assume that in future ubiquitous environments there will be sufficient specialisation to allow for the coexistence of multiple service location and interaction technologies simultaneously. This coexistence and rich heterogeneity of services and infrastructures poses problems in terms of unification, interoperability and consistency. In the classic case, where a client locates and interacts with a small number of services, many of the existing service discovery and interaction technologies work effectively. However, where a large number of clients interact with many services, issues of scalability and appropriate service selection become significant. Furthermore, we believe a crucial issue in terms of scalability, utility and performance will be the ability to reason about service selection and interaction on behalf of lightweight (potentially mobile) clients. In the following sections we consider these points in more detail.

3.1 Interoperability

Given the existence of multiple heterogeneous service discovery and interaction technologies one of the key issues for application developers will be how to deal with the diversity of device representations and interaction models. For example, UPnP defines XML schemas to represent devices, SLP uses defined URL syntax [7], HAVi uses device control module software elements and Jini uses serialised Java objects (placing stringent base requirements on client applications). In addition, UPnP's GENA, HAVi's Event Manager and the Jini distributed event model provide mechanisms for monitoring device state change notifications. SLP, in contrast, provides no such mechanism.

Clearly, creating an application that is to interact with more than one of these service location and interaction mechanisms simultaneously poses considerable challenges to the application developer. In subsequent sections we shall illustrate how these problems also impact service developers who must deploy services in heterogeneous environments. However, from an application developer's perspective it is, in our opinion, clear that a unified mechanism for locating, interacting with and representing services is required.

3.2 Scalability

As the number of clients and services in an environment increases, so the burden due to dynamic service discovery and interaction increases. The Windows Me UPnP client, for example, attempts to locate all the local network services upon initialisation. If the multicast search yields *n* service advertisements then *n* interrogations follow to gather the service description XML pages (the user would then initiate a further *n* interactions to get the presentation pages for each device). In the simple case of one root device with a single sub-device and service type we have found that, according to the UPnP specification, 18 packets are generated (6 packets each sent 3 times to avoid problems with packet loss) each refresh interval (30 minutes). As a client joins or roams into the network, it issues an ssdp:all m-search request (3 times for reliability). Every service must respond to each m-search by unicasting its service advertisements to the client (again 3 times per response). Hence, in this simple case, a total of 54 response datagrams are generated. It is evident that as the number of clients and services grow, low-bandwidth networks may become saturated with service announcement traffic.

Careful consideration of scalability issues is clearly important to the design of service location protocols (for example the aggregation facilities of SLP's directory agent or the refresh minimisation of Jini's distributed leasing scheme). However, we believe that great gains in bandwidth efficiency and improvements in scalability could be achieved by extending the power of the agencies in the network to reason on behalf of clients. In current systems, seeking an appropriate service is biased towards clients. More specifically, a client must enumerate through candidate devices by discovering and interrogating each service in turn if the state of the device is important in the selection process. A second client seeking the same service or entering the same domain will (using existing techniques) need to search the domain and interact with each service in turn as before. By using a network based intermediate agency the system could cache the results from client interactions and monitor for changes of state on behalf of clients, reducing consumption of network bandwidth and the time taken to locate services. This technique is already a well accepted approach to minimising traffic over low-bandwidth networks and has been employed in systems such as Rover [13].

3.3 Location-Based Services

Location-based applications (e.g. GUIDE [4]) represent a significant class of mobile computing applications. However, in current systems constructing location-based applications is complicated by two factors. Firstly, current service location and interaction techniques tend to offer simplistic scoping models, such as network domain based scoping or administrative multicast based scoping (SLP) and thus identifying a service that is applicable to a real world location (a room, building, area of a city etc.) that does not map onto such a topology is difficult. Secondly, the location of a service is not necessarily the field of that service's scope. For instance, a service that offers weather reports for the geographical location in which you are standing, may in fact be running on a server located in a remote city (or even another continent). Such a service clearly has two location attributes, a physical location relating to where the service is and a location that changes with the user's context that relates to the scope of the service. While research projects such as the Location Based Services (LBS) framework [12] are attempting to address these issues, current systems do not provide adequate solutions.

3.4 Time

Current service location and interaction frameworks concentrate solely on the 'present'; a client requires a service, finds the appropriate one and interacts with it. However, the temporal element to service interaction is often overlooked. For instance, a client may wish to locate the printer that they used last time, or the one they use most frequently. The trails and histories of device access are useful from both an application context and an administrative point of view (e.g. which clients have accessed a particular service and at what times, or what services were available during a particular time window). We believe that this temporal element to service location is particularly valuable in ubiquitous computing applications where the application context needs to 'live outside' one particular device or end-system (e.g. to facilitate enacting a task that has a lifetime longer than the use of just a single device).

3.5 State

The current state of a device or service in a service network is an important component of service location. For instance, a user seeking to print a document in a hurry is probably only interested in printers that have both paper and a short queue of jobs pending. Existing service location architectures provide facilities for the expression of state related information and the notification of change of state events (e.g. UPnP device descriptions and GENA). However, reasoning about dynamic state is not typically part of the service advertisement or discovery phases of the protocols. A client must thus enumerate through candidate services, utilising both time and network bandwidth, to identify the most appropriate service. Richer state based query semantics are therefore desirable, especially where partial network connectivity or low bandwidth is involved (e.g. supporting mobile access).

3.6 Security, Authentication and Access Control

Service frameworks are providing mechanisms by which users and applications can interact and control networked devices and services with unparalleled ease and convenience. Crucial to the successful adoption of such technologies however is security: can applications and users trust the services they find, and can the services trust the clients they serve? Authentication and certification mechanisms are required to ensure a suitable level of trust. Encryption and key distribution protocols are required to protect the exchange of sensitive information (these issues are being explored in the Ninja Service Discovery Service system

[5]).

Coupled with the notion of trust is control: the owner of a device might expect to be able to discover and control it and, furthermore, prevent others from maliciously tampering with the device. However, the same owner might also require that the system allows members of their family to control the device, perhaps only while they are present in the room or in a way that the owner approves of (e.g. not allowing a child to switch a television to an adult channel for instance). Although this level of social, role and contextual reasoning is almost certainly outside the scope of a service interaction protocol, such protocols should provide the core mechanisms to facilitate the construction of access control policies (e.g. access control lists, conflict resolution strategies and limitations on the adjustment of device and service state variables for particular clients).

3.7 Metadata

The underlying model of most service location and interaction strategies reinforces that of existing distributed network infrastructures: service providers or administrators establish conformant services within the network, which clients then access. The implication of this 'administrative' role is that the person or application that deploys the service controls the service's description and thus the attributes by which it can be discovered and used.

If we consider the way in which many services are used in real environments, we see that people 'personalise' their services, e.g. "I find this printer the fastest", "this projector belongs to my research group" and so on. These annotations are often personal or role based and may well change over time (e.g. a printer will always print on a certain size of paper which is part of its service description, but may well move between rooms or be replaced by a faster model). We believe that personalised, group and public metadata will be important to the utility of service location and interaction protocols. We do not intend that everyone should be able to modify the service description of a device, rather that service location platforms provide hooks for linking into meta-information databases. This would enable users and client applications to search for, for example, services that have been recommended by their colleagues.

3.8 Multiple Device and Service Reasoning

The focus of the service location and interaction technologies we have considered thus far in our analysis has been on locating a specific service or multiple services that offer a particular class of service (in fact, query support is currently lacking in some existing approaches [11]). We believe that by constructing agencies within the network that collate device and service advertisements and state information we can use higher level reasoning (queries) to gain further benefits in utility, timeliness and bandwidth reduction. For instance, a user in an active building might in a single query determine "All printers with paper to which I am allowed access that is nearby and not behind a locked door" by querying the printer, door lock and location tracker services.

3.9 Wireless Access

In general there is a poor match between current platforms' networking and end-system requirements and those typically found in mobile environments[1]. For example, many of the platforms use IP multicast for service announcements and this is likely to lead to significant levels of unwanted traffic between a mobile device and the fixed network. Moreover, many of the protocols used are extremely verbose (especially in the case of UPnP) and the platforms make significant demands on their client end-systems (e.g. support for Java in Jini).

Finally, none of the existing systems offer any support for intermittent connectivity, especially in the case of services hosted on a mobile device. For example, if a mobile device offers a service to devices on the fixed network there is currently no mechanism for maintaining state information for that device when it is operating in disconnected mode.

4. DESIGN AND IMPLEMENTATION ENVIRONMENT

The analysis presented in the preceding sections has highlighted a number of shortcomings that afflict many aspects of existing service location and interaction paradigms. In an effort to begin to address these shortcomings we propose a new middleware platform that aims to integrate existing service paradigms and offers an expressive API based on a modified form of the structured query language (SQL). Unlike existing approaches modelled on device databases (e.g. COUGAR [3]), our platform does not aim to replace existing service location and interaction platforms. Instead, we see these technologies as essential to underpin our architecture.

4.1 Overall Design

The overall design of our architecture is shown in Figure 1.

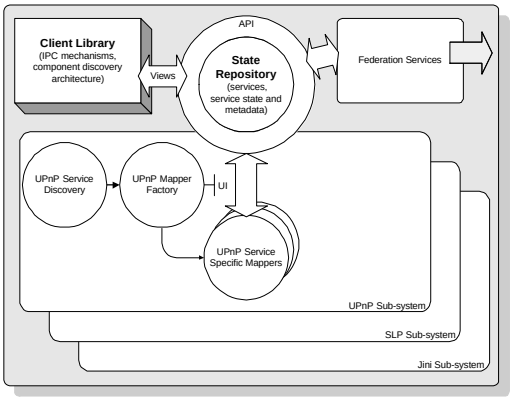

Figure 1 – Overall design of the platform architecture

The platform is structured as a lightweight set of components that may be distributed throughout the network as a set of services. The services may be replicated or federated within the network to improve availability and scalability or co-located to increase self-reliance. The platform as a whole functions by composing the platform services together as queries are executed. The state of the services and devices in the environment is thus the

[1] One notable exception is the work of Hodes et al. [10] on their Service Interaction Architecture, which focuses on service access for lightweight mobile clients. Note that mobile clients are not peer entities in this architecture and do not provide their own services to other clients.

composition of the information gathered by the platform components.

Client library

The smallest set of components that a client may instantiate is the client library, which consists of a lightweight component discovery mechanism, IPC mechanism and component instantiation mechanism. The client library presents the service query and interaction API (analogous to a temporal database) to applications (we anticipate that queries will be formulated in simplified ANSI SQL with some temporal extensions but we are considering a number of alternative APIs). In the minimal configuration, API queries are propagated to a remote query execution engine (API) component that interacts with the rest of the platform to enable service discovery. The query engine allows clients to provide rich service queries that can reason across multiple services, service types and, crucially, across service location mechanisms. Such a configuration would be ideally suited for execution on a resource-poor mobile device such as PDA or smart phone.

State management

The core unifying component of the platform architecture are the common models of service, state and device data. These models and their associated data are composed dynamically using service specific discovery functions (the UPnP discovery function is shown in Figure 1). It is worth noting that the state in our service database is composed almost entirely of soft state that is refreshed from the service discovery protocols. Such a unified information model requires dedicated service specific mapping functions for each location technology. While existing device systems typically focus on providing read-only state, we expect to allow applications to control devices by manipulating the values held in these tables.

Plug-in Service Specific Components

The service tables gather their information from dedicated service location discovery components. These components act as traditional clients to the supported service location technologies (e.g. control points in UPnP). Upon discovering a new service (or the change of state in an existing service), the component triggers the instantiation of a service specific mapping function from the appropriate mapper factory. The mapping function is responsible for transcoding the service advertisements and state change event notifications into a format suitable for the appropriate table. Since these platform components are themselves network based services, a mapping function can be written and deployed for a new service and the platform will dynamically discover the new function and reconfigure to include it.

Federation Services

We propose to link instances of the platform together, allowing bridging of service environments from one domain to another. Currently we anticipate developing a federation service that we can configure to link to other peer federation services. These services will bridge the central ‘table’ components together, allowing queries to be propagated to other service domains. One issue that we have yet to explore is whether these components federate the service name and state space between each other, whether information is synchronised between components such that federated table components become replicas or whether simply queries are distributed between components.

Transaction Support

One area we are particularly keen to investigate is the concept of a distributed service transaction. Existing service location and interaction technologies do not provide any support for regulating access (*c.f.* locking), checkpointing of service state or rollback behaviour. In the future many services will correspond to physical devices with tangible as well as networked ‘virtual’ user interfaces; we suspect therefore that a strong transaction may be impossible to achieve. However, the notion of being able to control sets of devices in an atomic and predictable fashion is certainly alluring and we aim to see what is possible using existing service location and interaction paradigms.

4.2 Implementation Environment

We are currently in the process of starting to prototype our platform. As a first stage we are attempting to prove the feasibility of mapping service types from heterogeneous service location technologies into a common format. As part of this initiative we are building a service testbed environment populated with both traditional networked devices and services (e.g. printers, information services and directories etc.) and prototype next generation appliances (such as networked door locks and location sensors).

The testbed currently includes a number of simple UPnP services including an SMS gateway, an ‘intelligent’ door lock and an X10 bridge. The X10 bridge controls a number of spotlights that may be controlled remotely in response to configurable stimuli (currently hits on a particular web page). Access to the testbed is via both wired and wireless networks and IPv6 is used as the network protocol.

5. APPLICATION SCENARIOS

To provide evidence of the applicability of our approach we now present a number of scenarios based on access to services from mobile clients.

5.1 Device Monitoring

Consider a scenario in which a user such as a security guard or operations manager wishes to discover devices in a particular state within their geographic area and have information about these devices displayed on their PDA. In the case of a security guard this might be doors that are currently unlocked or lights that are left on, while for the operations manager this might be devices that are due for repair work. Such an application requires an API that allows applications to query the availability of services based on both geographic location and the value of some state variable. In current systems this would be a very heavyweight operation, requiring the client to obtain references to all of the services in the network (possibly scoped by geography if the system supports static values in service advertisements) and then to poll each of these services for their state. Where the client is weakly connected this is clearly a poor strategy. In our system the polling of devices for state information can be managed in the fixed network, reducing the network traffic to the mobile device.

5.2 Discovering a Preferred Device

In this scenario a mobile user in an unfamiliar office environment wishes to discover and use a nearby printer. It would be ideal if the user could view and search by comments about printer

capabilities that have been added by colleagues in the office, e.g. "this printer jams frequently". In current systems this would require special support in every service – there is no mechanism for automatically integrating additional meta-data for services in a general way. In our architecture this can be achieved by the mapping components that can interrogate additional sources of information when creating representations of services. Hence no modifications are required to the underlying services or clients – metadata can be included in service advertisements as if it were generated by the service itself.

5.3 Interacting With Heterogeneous Platforms

In the final scenario we consider the design of a straightforward home theatre application that enables the user to start their favourite movie and adjust the blinds in their room by a single button press on their wireless IP-based PDA. In existing systems this is likely to be extremely difficult to achieve. The most fully featured system for controlling the AV devices is HAVi while the most likely candidate for controlling the blinds is X.10. Neither of these technologies are IP based and hence gateways will be required in both cases. In the case of HAVi this gateway may well expose the device's underlying HAVi API allowing sophisticated control of the device and the establishment of continuous media streams within the HAVi network, while in the case of the blinds these may be mapped onto a UPnP service. In this case the application on the PDA will be required to have bindings for the two distinct platforms, processing events and service announcements and requests from both domains. In our architecture these complexities are largely hidden from the programmer. Indeed, if in the future the user obtained a UPnP VCR rather than a HAVi VCR the application would require no changes – the system would simply instantiate an alternative mapping function for the device.

6. CONCLUDING REMARKS

In this paper we have argued that existing service discovery and interaction platforms have a number of significant shortcomings that, if not addressed, will seriously impact their utility, efficiency and scalability in future and emerging mobile ubiquitous service environments.

We have presented an initial design for a platform architecture that builds on existing service approaches and implementations to provide application developers with a simple yet powerful new API for discovering and interacting with distributed networked services. The platform is composed of multiple lightweight components that can be distributed throughout a network environment, making it particularly suitable for resource-poor mobile clients. We have demonstrated the usefulness of our approach by illustrating how it facilitates the development of a range of mobile applications. We are currently developing a prototype implementation of our platform that we will use to further refine the APIs and services we offer to the developers of future ubiquitous computing applications.

REFERENCES

[1] Bahl P. and V.N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System", In proceedings of IEEE INFOCOM 2000, Vol. 2, Tel-Aviv, Israel (March 2000), pages 775-784.

[2] Bettstetter, C. and C. Renner, "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol", In proceedings EUNICE 2000, September 2000.

[3] Bonnet, P., J.E. Gehrke and P. Seshardri, "Querying the Physical World", IEEE Personal Communications, Special Issue on Smart Space and Environments, October 2000.

[4] Cheverst, K., N. Davies, A. Friday and K. Mitchell. "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The Lancaster GUIDE Project", In proceedings ACM Mobicom'00, Boston, USA, August 2000.

[5] Czerwinski, S.E., B.Y. Zhao, T. Hodes, A.D. Joseph and R.H. Katz, "An Architecture for a Secure Service Discovery Service", In proceedings ACM Mobicom '99 Seattle, Washington, USA., 1999.

[6] Goland, Y., T. Cai, P. Leach, Y. Gu and S. Albright, "Simple Service Discovery Protocol/1/0", Work in progress, IETF Internet Draft draft-cai-ssdp-v1-02.txt, June 1999.

[7] Guttman, E., C. Perkins, J. Kempf, "Service Templates and , RFC 2609, 1999.

[8] Guttman, E., C. Perkins, J. Veizades, M. Day, "Service Location Protocol, Version 2", RFC 2608, 1999.

[9] HAVi Consortium. " HAVi Specification V1.0". 2000.

[10] Hodes, T.D., R.H. Katz, E. Servan-Schreiber and L.A. Rowe, "Composable ad-hoc Mobile Services for Universal Interaction", In proceedings ACM Mobicom'97, Budapest, Hungary. Pages 1 - 12. 1997.

[11] Hughes, E., D. McCormack, M. Barbeau and F. Bordeleau, "An Application for Discovery, Configuration, and Installation of SLP Services", MICON 2000.

[12] José, R. and N. Davies, "Scalable and Flexible Location-Based Services for Ubiquitous Information Access", In proceedings 1st International Symposium on Handheld and Ubiquitous Computing, HUC'99, Karlsruhe, Germany, 1999.

[13] Joseph, A.D., A.F. de Lespinasse, J.A. Tauber, D.K. Gifford and M.F. Kaashoek, "Rover: a toolkit for mobile information access", In proceedings 15th ACM Symposium on Operating Systems Principles, 1995, Pages 156 – 171.

[14] Kindberg, T., J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. serra and M. Spasojevic, "People, Places, Things: Web Presence for the Real World", In proceedings 3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000), Monterey CA. 7-8 December 2000, pages 19-30.

[15] McGrath, R.E., “Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing”, Technical Report UIUCDCS-R-99-2132, April 2000. http://choices.cs.uiuc.edu/2k/

[16] Microsoft Corporation, “Universal Plug and Play: Background”, Microsoft Corporation, 1999.

[17] Microsoft Corporation, “Universal Plug and Play Device Architecture Reference Specification, Version 1.0”, Microsoft Corporation, 2000.

[18] The Salutation Consortium, “Salutation Architecture Specification (Part 1)”, Version 2.0c, June 1999.

[19] Sun Microsystems Inc., “Jini Architectural Overview January 1999”, Sun White Paper, January 1999.

[20] Troll, R., “Automatically Choosing an IP Address in an Ad-Hoc Ipv4 Network”, Work in progress, IETF Internet Draft draft-ietf-dhc-ipv4-autoconfig-04.txt, April 1999.