ScreenSpot: Multidimensional Resource Discovery for Distributed Applications in Smart Spaces

Marko Jurmu
MediaTeam Oulu
Department of Electrical and Information Engineering
Univ. of Oulu, Finland
marko.jurmu@ee.oulu.fi

Sebastian Boring
Media Informatics Group
Department of Computer Science
University of Munich, Germany
sebastian.boring@ifi.lmu.de

Jukka Riekki
Intelligent Systems Group
Department of Electrical and Information Engineering
Univ. of Oulu, Finland
jukka.riekki@ee.oulu.fi

ABSTRACT
The big challenge related to the contemporary research on ubiquitous and pervasive computing is that of seamless integration. For the next generation of ubiquitous and distributed applications to emerge, disruptive functionality towards opportunistic and heterogeneous device ensembles is required on all levels of operation. In this paper, we present middleware-level resource management service for situated displays in public smart spaces, acting as a scheduler and an arbiter for mobile clients. From this service, we focus on multidimensional resource discovery, which facilitates mobile users in locating and deploying situated displays in public and semi-public smart spaces. Dimensions for discovery include dynamic availability of the displays in both spatial and temporal scales, user and role-based access control, as well as the support for intended service. We have implemented the discovery service and subjected it for alpha testing in an indoor setting. We report a proof-of-concept implementation of the ScreenSpot system and we demonstrate an approach of visualizing the discovery results to the user.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design.

C.2.4 [**Computer-Communication Networks**]: Distributed Systems.

I.3.6 [**Computer Graphics**]: Methodology and Techniques.

General Terms
Management, Design, Human Factors.

Keywords
Ubiquitous computing, leasing, dynamic QoS, publish/subscribe.

1. INTRODUCTION
Current research in the field of ubiquitous and pervasive computing [1, 2] is faced with a problem related to integration. The parts constituting the hardware side, i.e. small ultra-portable personal terminals and wearable sensors, as well as various sorts of ambient resources are already in their place due to the constantly decreasing costs of manufacturing and deployment. Same holds for networking, where Bluetooth is already a de-facto standard, WLAN emerges fast and 6LoWPAN sensor connectivity is paving its way to urban domains. The question then becomes how to opportunistically integrate these heterogeneous resources in order to realize the application models [3] the vision of ubicomp entails.

Within this problem domain, contemporary research has acknowledged the need for disruptive, cross-layer viewpoints on all levels of operation. The research on cognitive radios and dynamic spectrum access [4, 5] are examples on the PHY and MAC layers, while research on publish / subscribe systems [6, 7] propose disruptive mechanisms for data-centric routing. On the middleware layer, approaches towards interoperability are multitude, and the scope of deployment divides different approaches to different research areas such as large-scale grids [8], or smart spaces such as office environments [6, 9], homes [10, 11] or more generic public spaces [12, 13].

On the application level, research on integration and interoperability is enabled through various service discovery protocols [14]. Jointly with web service technologies, these protocols enable the construction of loosely coupled service-oriented architectures, or SOAs. The application level also features the research conducted in HCI towards the interaction mechanisms employed in communication between humans and ubiquitous device ensembles. Ballagas et al. [15] provide a thorough survey on this field.

A typical usage scenario in this research field involves a mobile user with a smart phone, PDA or other similar networked terminal, utilizing services from the ambient surroundings. The traditional view of service discovery and deployment through a federated service repository usually involves three distinctive steps: first, the user issues a discovery request including certain discrete keywords to the service repository, which in turn performs static matchmaking to return a set of matching service descriptions from the directory. According to some evaluation heuristics, a suitable service is selected from the candidate set, and a proxy for this service (either the service grounding or a physical software proxy) is downloaded into the mobile terminal. In the final step, the mobile client utilizes this proxy to control and exchange data with the remote service.

In the case of distributed and non-directory-based discovery protocol, two operational modes are possible. In the pull-based model, the client issues service requests as multicast messages to the network in order to discover suitable service candidates. In the push-based model, on the other hand, the service implementations

publish advertisements of their presence to the network, and clients can tap to this traffic to perform discovery.

The approaches depicted in the above sections present straightforward solutions to service discovery and deployment, but they contain certain shortcomings when applied to pervasive computing scenarios. First is the reliance on network topologies as the discovery range instead of physical location models of smart spaces, such as one presented in [16]. Second, the quality-of-service aspects related to the service are restricted to static capability declarations. In other words, the dynamic usage of the service and underlying resources are not evaluated. A direct implication of this is that load balancing between users is minimal, leading to hoarding and starvation situations. Finally, smart spaces should enforce access controls that allow different views to smart space service spectrum based on credentials and roles of the users [14].

One solution to the problems presented above is to extend the SOAs with context-aware features to better fit ubicomp scenarios [17]. This is however not feasible due to several reasons. First, ad-hoc extensions tend to create isolated domains of functionality where off-the-shelf discovery clients may not be aware of or able to interpret the enhancements. Second, the co-operation of these domains requires the installation of custom translation proxies on network edges, further increasing the deployment costs. In this paper, we suggest an alternative solution; a general resource management and scheduling service on the middleware level that allows the SOAs to focus on their intrinsic key functionality of exchanging controls and data between distributed service components.

We adopt a decentralized, infrastructure-centric view to discovering resources and services in the ambient environments, or smart spaces. Instead of a federated service repository, we view the infrastructure of the smart space in a more bottom-up fashion, as follows: first, smart spaces contain ambient resources that are managed through associated resource management (later: RM) components. These components are aware of the service binaries deployable on the associated resources. Through a pub/sub routing mechanism [7], these components form a loosely coupled peer-to-peer mesh overlay network, through which the components communicate with each other. In resource discovery, one of the components acts as a *seed* for the discovery request and uses the mesh overlay network to aggregate multidimensional availability information [18] regarding the other resources in the smart space. The seed component is responsible for aggregating the availability information and passing it to the mobile terminal for visualization to the user.

When the user has evaluated the results, the optimal resource is chosen and the negotiation between the mobile client and target resource regarding the service deployment starts. The negotiation ends with an agreement between the participants regarding the validity of the ownership transfer. We view this process as *context-aware leasing* [19], where the lease represents the transient transferring of the ownership, and the validity of the lease is based on spatiotemporal context elements.

The contributions of this paper are as follows: First, we report the design and implementation of ScreenSpot, a decentralized resource discovery framework for smart spaces. We focus especially on situated displays [15] as means for constructing multimodal user interfaces for mobile users. This framework is based on data-centric, publish/subscribe messaging semantics which allows loosely coupled resource networks to be constructed. Second, we demonstrate a multidimensional availability and thus quality-of-service structure for the resources, based on the dynamics of the usage in addition to the static properties. Third, we illustrate an approach for visualizing the discovery information for the mobile users. This approach of facilitating the high level user-based choice through context-awareness is influenced by the SpeakEasy project in Xerox PARC [20].

This paper is organized as follows: the system overview section defines the key terms and presents our design for the resource discovery service. It also features a short usage scenario as an example of utilizing the discovery service. The validation section presents a proof-of-concept implementation of the ScreenSpot system along with a visualization concept for showing temporal and spatial availability of displays. Finally, in the conclusions section we draw up the main findings of the publication and contemplate on various comparison points between the traditional discovery systems and our design. Also multiple points of consideration for future research on this topic are presented and discussed.

2. SYSTEM OVERVIEW

In this chapter, we present the design of the resource discovery system in detail. First, we illustrate the functionality of the resource discovery by an example usage scenario. Next, we proceed to define the central terms and concepts to be used throughout the design. Subsequently, the high-level architecture of the system is presented in detail. Finally, we illustrate the functionality of the system through a selection of sequence diagrams.

2.1 Usage Scenario

Alice is visiting a shopping mall with her friend. The mall premises acts as a smart space by containing a number of ambient hotspots, i.e. collections of ambient resources with associated management software and physical connectivity. These hotspots in conjunction with the personal mobile devices allow the deployment of distributed application structures that realize multimodal user interfaces towards the users.

Alice would like to deploy a personal media organization and sharing application jointly with her friend to view, organize and share pictures and media clips from a concert she visited couple of days ago. As she is in an environment not known to her a priori, she doesn't know what resources are open for deployment, or where they reside.

To discover deployable situated displays for the media application, she starts the ScreenSpot service on her mobile terminal, and initiates a discovery with certain parameters regarding the properties of the resources. Within seconds, she is presented with an integrated view of deployable situated displays, arranged into a radar structure based on the multidimensional availability information gathered from the environment.

The integrated view allows Alice to easily compare the different displays based on both static and dynamic properties. After selecting one display, she requests extra information of it. This extra information contains an image of the surroundings of the display, and Alice notices that the location of the screen is next to

the Starbucks of the first floor. She sets a lease for this display, closes the application and starts to head towards the Starbucks.

The scenario above illustrates how an integrated discovery view to a smart space can facilitate users in selecting deployable resources from the surrounding environment. In the case of situated displays, Alice does not have to explicitly know which resources she is able to utilize, since the discovery service automatically incorporates the information regarding the support for the intended application, as well as the notion of displays accessible by users acting in the role of *guest* within the smart space.

In the remainder of this article, we present the technical details of our proof-of-concept implementation, ScreenSpot, which is aimed for situations described above. We cover the networking topology and the situated hotspots with associated data structures, and explain the functional sequences that realize the discovery processes. We also present screenshots of the integrated discovery views seen by the end users, and explain the different aspects of this view.

2.2 Definitions

In this section, we define the terms that will be utilized throughout this paper, and form the core concepts of our research work. The concepts will be described bottom-up, starting from the routing level and ending on the level of individual service components.

2.2.1 Pub/Sub Client

Since the RM middleware is running on top of publish/subscribe routing system, the term pub/sub client in this context refers generically to any computational endpoint that interfaces the pub/sub network in order to exchange data with other pub/sub clients. All the RM instances realize the pub/sub client interface, as well as the client components of the middleware running on the mobile terminals.

2.2.2 Resource

Resources in this system are objects that are used in executing services, and they feature management interfaces decoupled from the associated services [18]. The usage of each resource is defined and constrained in its *usage policy*, which can be set on a resource independently of other resources. The execution of services on the resource is semi-static in nature, meaning that although the resource is hosting multiple service binaries, only one of them is deployed at a certain point of time. In a special case of non-exclusive resource, multiple service binaries can be running. Rules and constraints for this deployment are defined in the resource's usage policy.

2.2.3 Service

Services are software objects to which computational access is granted through well-defined interfaces and groundings. A service can also be involved in the construction of the user interface of the device ensemble towards the user. Each resource can host multiple services, depending on the purpose and capabilities of the resource. The capabilities of the services are described through static quality-of-service declarations. The RM instance on every resource is aware of the hosted service binaries and deploys them based on the scheduling and usage of the underlying resources. Each node in the smart space contains a set of different services where some of these services may be offered by multiple nodes. We define the set of all offered services as service spectrum of the smart space.

2.2.4 Lease

In our research, we view leasing as a process of transiently transferring the control of the leased entity to the client, along with some validity criteria and renewal options [19]. To utilize an ambient resource, the user must have an *active lease* to the resource. If the active lease cannot be set due to contention situation in the resource usage, the user can set a *pending lease* to the lease queue of the resource. When a pending lease reaches the head of the queue, it is promoted as the active lease and denotes the transferring of the resource ownership to the new user. Simultaneous usage of a resource (in collaborative applications) is enabled by allowing a single lease to encompass multiple owners.

2.2.5 Dynamic QoS

We define the concept of a dynamic quality-of-service as a hybrid description containing both the declarative capabilities of the service and the dynamic usage load of the underlying resource. To attain the dynamic QoS, certain heuristics are applied (that can be supplied to the system by a third party, for example) for integrating the degree of match between the service discovery request and the service capabilities (i.e. the static matchmaking) with the dynamic usage load of the underlying resource. We model the usage load on the resource as a FIFO queue containing the leases set by mobile users wishing to utilize the resource [19].

2.3 Architecture

This section presents the architecture of the RM middleware system. We first discuss the system on the networking level and introduce the relationships between the central entities. We then proceed to examine the structure of an individual RM instance, to gain an insight to the functionality that the management interface of each components contain, as well as what the core data structures are. The section ends with sequence diagrams of selected functionality which serve to highlight the communication and control between the components in realizing the higher level goals of the system.

2.3.1 Overview

Figure 1 illustrates an example setup of the entities comprising the RM middleware. Central to the figure is the Fuego pub/sub routing system, which is illustrated with a single router for simplicity reasons. The mobile terminals connect as pub/sub clients to the routing subsystem through the IEEE 802.11b panOulu [22] WLAN access network. Individual RM components in this figure are modeled as situated displays, since the work focuses on (large) public displays at this point. They contain terminal computation entities that act as containers for the respective RM instances and interface the pub/sub routers through IEEE 802.3 Ethernet links.

In addition to the aforementioned connectivity, the mobile terminals also engage ad-hoc communications with the ambient RM instances through Bluetooth. Through this short-range ad-hoc connectivity, we want to physically enforce the aspect of spatial proximity to the computation. The Bluetooth coverage area forms an ad-hoc connectivity hot-spot around the situated display, and we utilize this coverage area as a virtual watchdog for monitoring the proximity between user and the display. Through this functionality, the leasing for the display can be monitored in a way that user can terminate the leasing session merely by walking away from the display [19], thus we inhibit an implicit interaction session termination through the Bluetooth watchdog beacons.

Figure 2 depicts the structure of a single RM instance in the middleware. This instance resides within each situated display, and is responsible for the *allocation* and *scheduling* of the resource instance, as well as the execution of service instances residing on the resource. This design reflects the tight coupling of the resource with the associated services, which is essential in the case of stateful distributed applications. During the execution, it may be necessary to inject the application-related data from the client side to the respective ambient resource to reflect the state of the distributed application.

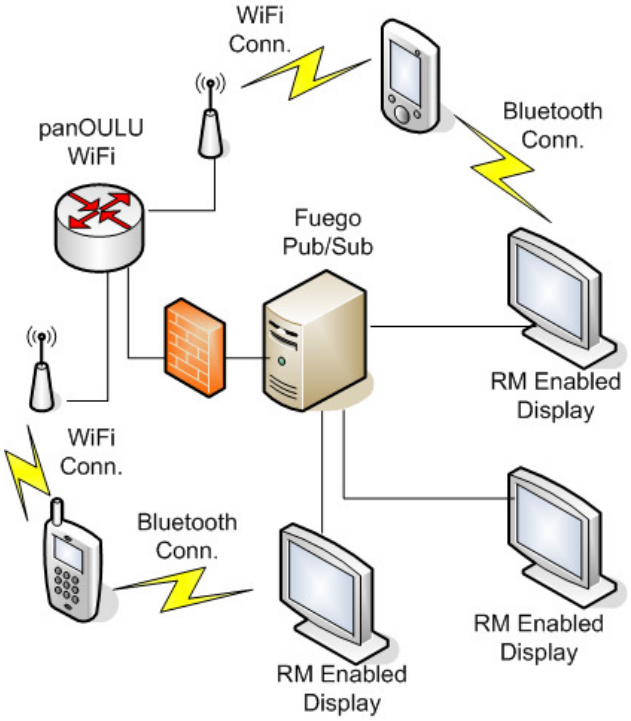

Figure 1. High-level architecture of ScreenSpot.

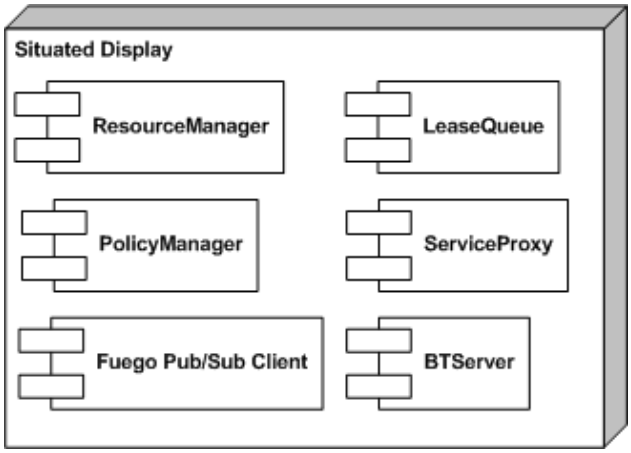

Figure 2. Component structure on a situated display.

The *ResourceManager* component forms the core of the RM instance and coordinates the other components. It realizes the discovery interface towards the mobile clients, and controls the execution of the discovery procedure. *PolicyManager* acts as a container and controller for the usage policies set for this resource. A single usage policy is an aggregation of resource utilization rules for a single user group. These include an access control list with usernames of the group, the renewal policies allowed for the group, as well as the role that the group has in the smart space, i.e. *employee* vs. *guest*.

Fuego pub/sub client realizes the networking interface towards the publish/subscribe routing system. The subscription semantics utilized in the discovery service are hybrid in nature. First, the routing system realizes a separate subscription channel for the discovery traffic. Secondly, the RM instance on each ambient resource subscribes only to the messages targeted to this group of instances (noted with an identifier). This decoupled routing scheme allows a flexible maintenance of the smart space, which is difficult and sometimes impossible to perform through a complete shutdown and restart. A discovery request sent to the network is only received by running resource managers that have an active subscription, and thus are ready for resource utilization and service deployment.

LeaseQueue maintains the queuing of the leases set for the resource instance by mobile clients. The basic queuing scheme is FIFO, but e.g. a priority-based queue associated with user roles is also possible. It should be noted that pre-emptive scheduling of users in this setting is highly counterproductive, especially when considering stateful applications. For this reason, we see the queuing and differing *lease renewal policies* more useful in enforcing soft scheduling processes.

ServiceProxy acts as a singleton interface towards the core, and allows the execution of the service binaries residing within this resource object. The service binaries are tightly coupled with the resource instance and the users issue utilization requests to the resource by setting leases to the associated lease queue. Hence, each lease must be associatively connected to a deployable service binary. In-depth illustration and evaluation of the service proxy component is out of scope for this publication.

Finally, *BTServer* implements the physical Bluetooth connectivity around the ambient hotspot. This spatial connectivity has two distinct roles in our system. First, users residing within the Bluetooth coverage area of a single hotspot can perform discovery requests to the rest of the system through the local hotspot. The benefit of this scheme is that users' relative locations in the smart space can be straightforwardly inferred based on the Bluetooth attachment point. This in turn facilitates presenting results to the users based on relative locations of the discovered resources.

Secondly, during the resource deployment, the Bluetooth coverage area acts as a *virtual spatial watchdog*. This watchdog enforces a spatial aura around the ambient resource. As the user leaves the proximity of the resource, the system can implicitly infer that the application session has ended, removing the need for the end user to manually close down the session.

2.4 Functionality

In this section, we highlight the functionality of the ScreenSpot system through a series of sequence diagrams. As stated in

previous sections, the main functionality of the system is to perform discovery requests within a physical smart space domain and return multidimensional resource information back to the user. The attachment points for end users in the system are Bluetooth coverage areas around the ambient hotspots that provide resources and deployable services.

In Figure 3 the roles of the different RM instances are illustrated within the context of a single discovery request. As the user is residing within an ambient hotspot when the discovery is initiated, the RM instance managing this hotspot becomes the *seed* for the discovery request. From this seed, other RM instances within the smart space domain appear on different virtual spatial ranges, denoting relative physical distances between the hotspots.

Range zero consists of the local hotspot, and the successive ranges are dependent on the location model configured to the system. Range 1 can for example denote hotspots within the same floor as the seed instance, while range 2 denotes instances within the next floor up. Outer ranges denote other areas within the smart space physical domain.

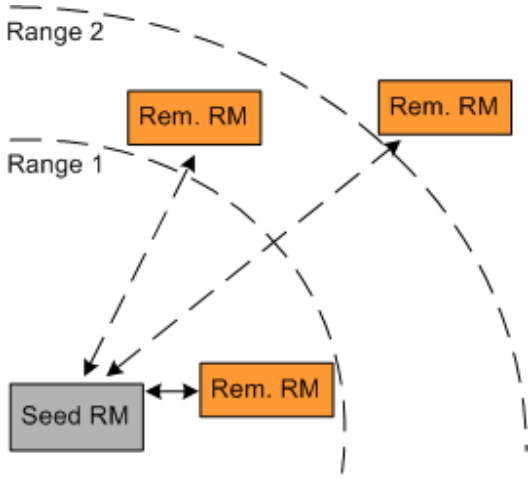

Figure 3. Virtual spatial ranges from the seed instance.

In Figure 4, the sequence of operations between the mobile terminal and the seed instance of the system is illustrated. The sequence starts by the mobile terminal scanning for the Bluetooth service of the seed instance. This can be initiated by the start-up of a certain application in the terminal, or by an explicit discovery request by the end user.

When the service is discovered, the mobile terminal passes the discovery parameters to the seed instance. Here, we assume that Bluetooth pairing between the nodes has already occurred. Parameters include the Bluetooth MAC address of the terminal, in order to distinguish between multiple concurrent discovery requests from the common seed. Other parameters include username of the end user, the identifier of the intended application, the spatial range for the discovery and the type of the ambient resource to be deployed.

In the case where the end user does not have a designated username to the smart space domain in question, she is assumed as a guest, and subsequently the usage policy for users with the role *guest* is utilized. To obtain the unique identifier of the application, such as a UUID, we assume the usage of a name resolution system such as INS [22], where an early binding query can be used to obtain the identifiers. Integrating a name resolution system such as INS is recognized as a future work in this research.

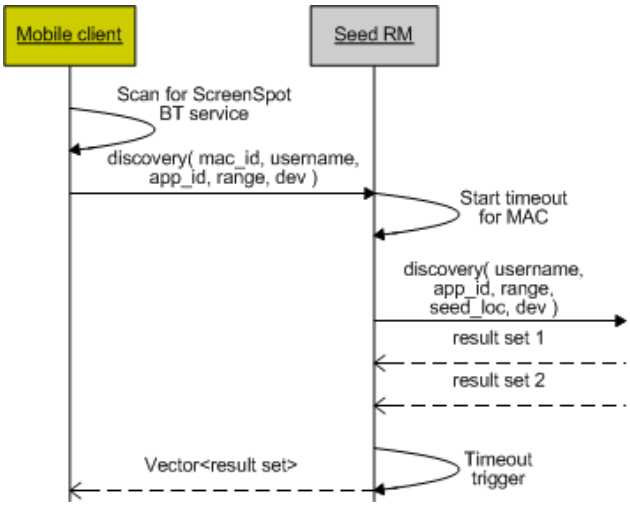

Figure 4. High-level resource discovery process.

When the seed instance receives the discovery request, it starts a discovery timer with the MAC identifier of the terminal. Subsequently the discovery request is forwarded to the publish/subscribe routing for other instances to receive. Each instance residing within the range defined in the request answers the seed instance with a result set containing its multidimensional availability information. The discovery sequence ends when the discovery timer for the designated end user triggers a timeout. After this, the received result sets are aggregated into a vector which is passed back to the mobile terminal for visualization.

Figure 5 illustrates the process of resolving the availabilities of remote resources in more detail. The discovery message shown in detail in Figure 4 is published to the discovery channel, and thus received by all active RM instances. First calculation on the remote system is the relative distance to the seed. This relative distance is represented with a single integer, and we refer to it as a *spatial coefficient*. The discovery request is relevant to the remote subsystem, if

$$coeff_{spatial} \leq range_{disc} \quad (1)$$

i.e., if the remote instance resides within the relative range defined in the request.

If equation (1) applies, the remote ServiceProxy resolves the support for the intended application in the remote node. If the application is available (i.e. the corresponding binaries are existent), the remote node proceeds to authenticate the user. This is done by the PolicyManager, based on the usage policy set for the role of the user. Finally the temporal availability is calculated by the LeaseQueue according to following equation:

$$avail_{temp} = t_{al} + \sum t_{pl} \quad (2)$$

The temporal availability consists of the currently remaining time of the *active lease*, as well as the cumulated negotiated durations of the *pending leases*. The latter can be different for individual leases, as the durations are always associated with the usage policies the users are authenticated with.

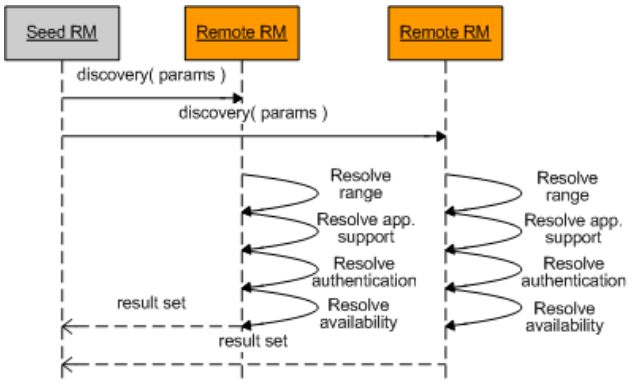

Figure 5. Resolution of availability information.

After all the availability dimensions for the remote subsystem are calculated, they are grouped together as a result event, and published through the pub/sub network directly to the seed RM. In the case of successful authentication of the user, as well as recognized support for the target application, the overall availability is denoted with the spatial coefficient and the temporal availability. If the user cannot be authenticated, or the target application is not supported, the availability of the resource is set to infinite and presented as *unavailable* to the user.

3. VALIDATION

This section presents the proof-of-concept implementation of the ScreenSpot system. First we will describe the visualization concept used to give feedback which allows a fast and simple selection of a screen. Second, we will show how the user can interact with the described user interface concept including screenshots of the mobile application. Finally, we present a theoretical use case of integrating UPnP control points and services on top of our resource discovery framework.

3.1 Representation to Users

In this section we highlight the representation of results to the user while using the ScreenSpot system. The visualization will be described from the user's point of view mostly concerning the representation of displays with their static and dynamic properties.

3.1.1 Visualization Concept

As mentioned beforehand, the ScreenSpot system tells users about the spatial as well as temporal availability of a resource (e.g. a display). Hence, the user interface on the mobile client needs to visualize both parameters in a compact way as screen space is limited and thus expensive. Both time and space already have well defined and widely used graphical representations. Temporal coherences can be depicted with a clock-based metaphor, whereas distance is often illustrated by user-centric radar-style interfaces.

In order to allow users a fast decision based on the results, the system needs to take the advantages of both visualizations and merge them into one view. Radar views give two parameters (i.e. distance and direction) where as a clock only gives the time (as direction). With the ScreenSpot system, the direction cannot be detected as it does not track the user's orientation. Hence, we can integrate the radar's distance parameter as well as the clock's time parameter into a new two-dimensional representation. Thus, the representation shows a clock-based view with additionally showing the distance between the user and the screen. Figure 6 sketches our approach.

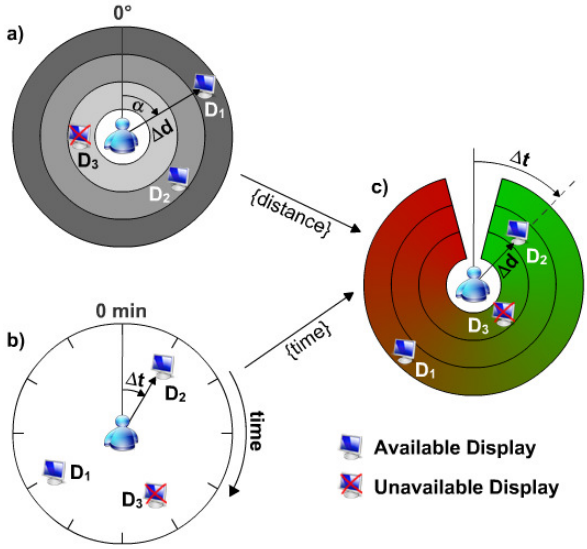

Figure 6. Visualization of the results. a) shows a standard radar view with orientation and distance as parameters. b) depicts a clock-based view with the time as radial parameter. c) illustrates ScreenSpot's view comprising the distance (radar) as well as the time (clock).

The remaining parameter (i.e.: *"are the application binaries available at the display?"*) can also be visualized with ScreenSpot. As seen in Ffigure 6, some displays might be crossed out. This indicates a display in the environment with its spatial and temporal availability that is not able to execute the application. Even though the user is not able to run the desired application on the display, s/he can later use the display for other purposes.

3.1.2 Requesting Additional Information

With the mobile screens' limited resolution and transmission bandwidth (Bluetooth), information about a display needs to be placed step-by-step. Hence, the user only gets spatial and temporal availability information about the displays in the environment. Before a user finally makes a selection, s/he might want to gather more information about the display itself. The information includes further parameters and attributes of the currently selected screen, such as the room it is in, its resolution or its physical size.

In addition, a picture of the display can be requested by the mobile client. This picture can then be taken by a nearby webcam to give the user a real-time view of the screen. In our prototype, the pictures are pre-captured and stored for each instance.

3.2 Interaction with the UI

Most today's mobile phones are equipped with a joystick or arrow keys. The two independent parameters (time and distance) can thus be controlled by those input devices. For example, the user can utilize the left and right keys (and joystick respectively) in order to select a different level of distance corresponding to the spatial coefficient (e.g. *corridor*, *floor* or *building*). With the up and down keys (and joystick respectively) the user is able to switch between the displays contained in this distance level. In addition to the three distance level, we introduced a level showing all displays (e.g. in all distance levels). This virtual level can also be accessed by either using the left and right keys or the joystick.

A selection of a display is indicated by both a white circle beneath it as well as a tooltip showing the exact time until the display is available. Once the user has selected a display, s/he can either request more information about it or immediately select it. The selection of a display is also possible in the information screen. Hence the user does not have to traverse back to the main selection screen which in the end reduces the number of clicks needed to finally select a display through ScreenSpot. Figure 7 shows screenshots of the possible interaction path a user takes.

If the user has selected a display for use, the system creates and sends a lease request to the associated RM instance. This results either in a success screen informing the user about a successfully placed lease or an error screen containing information about the error occurred. These errors might include a lost connection or a lock placed on the screen in the meantime by another user.

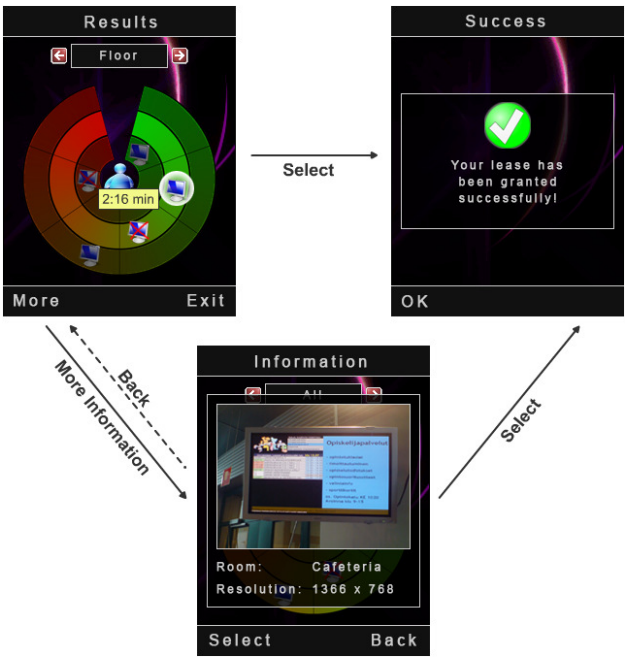

Figure 7. Screenshots of the mobile UI. Both paths are shown: either the user requests more info, or s/he immediately selects the desired display.

3.3 Case Study: Integration with UPnP

This section presents a theoretical use case for integrating universal plug and play (UPnP) control points and services on top of ScreenSpot. UPnP is paving its way as a service-oriented architecture in smart spaces, but the network-dependent nature of the simple service discovery protocol (SSDP) utilized in UPnP constraints its use in the construction of situated user interfaces. The discussion is divided into two sections: First one deals with the control points, whereas the second section focuses on the services.

3.3.1 Control Points

The pull-based service discovery in UPnP is realized with the control point broadcasting a service discovery request within the network domain. This broadcast message is answered by services attached to the same domain. Although straightforward, this solution lacks any information about the associated resource states in the smart space.

Through ScreenSpot, the pull-based discovery can be enhanced to include the necessary resource information. To resolve the potential service candidates, the broadcast message on the local device can be captured and fed to a name resolution framework such as one implemented by INS. By utilizing distributed hash tables, this resolution returns a list of universally unique identifiers (UUIDs) used in UPnP for identifying services. These UUIDs in turn can be utilized as the service identifiers in the ScreenSpot framework to resolve service support on dedicated hotspots.

3.3.2 Services

In contrast to pull, the push-based service discovery in UPnP features the services broadcasting their presence in the network for any control points attached to the same domain. In this approach, the utilization of INS name resolving and ScreenSpot resource discovery can also be utilized.

When a service is introduced into a situated hotspot, it tries to announce its presence to the network. On the local hotspot, this message can be captured and fed to the ScreenSpot framework. This presence message can be utilized in two ways: The INS system in the smart space can use this message to perform soft-state management of the service, as well as extract the service description to be utilized in name resolving queries. The resource discovery side in addition can capture the UUIDs of the local services in order to provide the multidimensional discovery results explained in this article.

3.3.3 Outcome

The previous sections described the integration of UPnP control points and services on top of a ScreenSpot / INS framework on a theoretical level. What is notable in this discussion is the fact that the actual UPnP components in this setting do not require any additions or extensions in order to operate. Only aspect requiring explicit integration is the UPnP service to adhere to the bootstrapping protocol on the hotspot side. Through this short discussion, we have demonstrated the power of introducing a dedicated resource discovery framework for smart spaces and how it frees the developers of SOA components from the design of ad-hoc extensions.

4. CONCLUSIONS

In this paper, we presented the design and implementation of ScreenSpot, a middleware level resource discovery system for smart spaces. ScreenSpot incorporates a tightly coupled model between ambient resources and associated services, and executes

on top of an asynchronous publish/subscribe networking scheme. This allows flexible maintenance of the smart space in relation to both resources and the associated services.

After motivating the need for a middleware level, general-purpose resource allocation and scheduling system, we presented a distributed design that tackles the challenges posed by service discovery systems in smart spaces, as indicated also by Zhu et al. in [14]. These challenges are primarily as follows: utilization of physical smart space boundaries instead of IP networking topologies as the discovery range, which in our system is tackled by the virtual ranges between the ambient resources and the calculated spatial coefficients. Second challenge is role-based view of the smart space, which in our system is handled by managing different usage policies for different roles that the end users are representing.

The proof-of-concept implementation presented in this article realizes all the components illustrated in Figure 2. The client side of the discovery system is also implemented. These realizations give us a good basis for conducting further research in this area.

Through the implementation, it is clear that the Bluetooth device scanning amounts for the majority of the latency experienced by the end users. To alleviate this, we see the utilization of RFID tags as a viable solution in smart spaces [23]. These RFID tags can be configured to provide the mobile terminal with the necessary service parameters, thus greatly reducing the scanning time as indicated in [24].

We acknowledge the need for security and privacy aspects in this work. Levels of security we are considering in the future development include the encryption of the Bluetooth traffic, as well as the encryption of the pub/sub traffic with a PKI scheme offered by the Fuego routing system [7].

Finally, we recognize multiple tracks of future work in this research area. First, we are in progress of preparing first distributed applications on top of this system, and will be later publishing results from user tests based on the combination of the discovery system and the application usage. Secondly, we are researching the possibility for physically distributed application sessions involving distributed leases. Such scenarios allow applications such as video calling by utilizing ambient displays and webcams, or physically separated collaborative brainstorming. Finally, we are looking into the stateful bootstrapping of distributed applications, and the effects that the state injection and extraction has to the scheduling of the resources.

5. ACKNOWLEDGMENTS

This work has been conducted within the UbiLife project framework. The authors would like to thank TEKES and the participating companies for financial support. First author would also like to thank the GETA graduate school for financing this research. This research has also been partly funded by "Deutsche Forschungsgemeinschaft" (DFG).

6. REFERENCES

[1] Weiser M. (1991) The Computer for the 21st Century. *Scientific American*, September 1991.

[2] Satyanarayanan M. (2001) Pervasive computing: visions and challenges. *IEEE Personal Communications 8(4)*, pp. 10-17.

[3] Banavar G., Beck J., Gluzberg E., Munson J., Sussman J. & Zukowski D. (2000) Challenges: An Application Model for Pervasive Computing. *Proc. 6th Annual International Conference on Mobile Computing and Networking (MOBICOM 2000), Boston, MA, USA*, pp. 266 – 274.

[4] Mitola J. III & Maguire G.Q. Jr. (1999) Cognitive Radio: Making Software Radios More Personal. *IEEE Personal Commnunications 6(4)*, pp. 13 – 18.

[5] Akyildiz I.F., Lee W.Y., Vuran M.C. & Mohanty S. (2006) NeXT Generation / Dynamic Spectrum Access / Cognitive Radio Wireless Networks: A Survey. *International Journal of Computer Networks 50(13)*, pp. 2127 – 2159.

[6] Johanson B., Fox A. & Winograd T. (2002) The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing 1(2)*, pp. 67 – 74.

[7] Tarkoma S., Kangasharju J., Lindholm T. & Raatikainen K. (2006) Fuego: Experiences with Mobile Data Communication and Synchronization. *Proc. 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, *Helsinki, Finland*, pp. 1 – 5.

[8] Foster I. (2002) The Grid: A New Infrastructure for 21st Century Science (book chapter). *Grid Computing: Making the Global Infrastructure a Reality*, J. Wiley & Sons. ISBN: 9780470853191.

[9] Tripathi A.R., Kulkarni D. & Ahmed T. (2005) A specification model for context-based collaborative applications. *Elsevier Pervasive and Mobile Computing Journal 1(1)*, pp. 21 – 42.

[10] Helal S., Mann W., El-Zabadani H., King J., Kaddoura Y. & Jansen E. (2005) The Gator Tech Smart House: A Programmable Pervasive Space. *IEEE Computer 38(3)*, pp. 50 – 60.

[11] Georgantas N. et al. (2005) The Amigo Service Architecture for the Open Networked Home Environment. *Proc. 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, pp. 295 – 296.

[12] Sousa J.P., Poladian V., Garlan D., Schmerl B. & Shaw M. (2006) Task-based adaptation for ubiquitous computing. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 36(3)*, pp. 328-340.

[13] Roman M., Hess C., Cerqueira R., Ranganathan A., Campbell R.H. & Nahrstedt K. (2002) A middleware infrastructure for active spaces. *IEEE Pervasive Computing 1(4)*, pp. 74-83.

[14] Zhu F., Mutka M.W. & Ni L.M. (2005) Service Discovery in Pervasive Computing Environments. *IEEE Pervasive Computing 4(4)*, pp. 81 – 90.

[15] Ballagas R., Borchers J., Rohs M. & Sheridan J.G. (2006) The Smart Phone: A Ubiquitous Input Device. *IEEE Pervasive Computing (5)1*, pp. 70 – 77.

[16] Satoh I. (2007) A location model for smart environments. *Elsevier Pervasive and Mobile Computing Journal 3(2)*, pp. 158-179.

[17] Lee C. & Helal S. (2003) Context attributes: an approach to enable context-awareness for service discovery. *Proc. 2003*

Symposium on Applications and the Internet (SAINT), Orlando, Florida, USA, pp. 22 – 30.

[18] Perttunen M., Jurmu M. & Riekki J. (2007) A QoS model for task-based service composition. *Proc. 4th International Workshop on Managing Ubiquitous Communications and Services (MUCS'07), Munich, Germany*, pp. 11 – 30.

[19] Jurmu M., Perttunen M. & Riekki J. (2007) Lease-based resource management in smart spaces. *Proc. Workshops of the 5th International IEEE Conference on Pervasive Computing and Communications, White Plains, NY*, pp. 622-625.

[20] Edwards W.K., Newman M.W., Sedivy J., Smith T. & Izadi S. (2002) Challenge: recombinant computing and the speakeasy approach. *Proc. 8th Annual International Conference on Mobile Computing and Networking (MobiCom), Atlanta, Georgia, USA*, pp. 279 – 286.

[21] panOulu Public Access Network, URL: http://www.panoulu.net/ (Accessed: Mar. 14th, 2008).

[22] Adjie-Winoto W., Schwartz E., Balakrishnan H. & Lilley J. (1999) The design and implementation of an intentional naming system. *Proc. 17th ACM Symposium on Operating System Principles, Charleston, South Carolina, USA*, pp. 186 – 201.

[23] Riekki J., Salminen T. & Alakärppä I. (2006) Requesting pervasive services by touching RFID tags. *IEEE Pervasive Computing 5(1)*, pp. 40 – 46.

[24] Salminen T., Hosio S. & Riekki J. (2006) Enhancing bluetooth connectivity with RFID. *Proc. 4th International Conference on Pervasive Computing and Communications (PerCom), Pisa, Italy*, pp. 36 – 41.