Practical Web-based Smart Spaces

C. Prehofer, J. van Gurp, V. Stirbu, S. Sathish, P. Liimatainen, C. di Flora, S. Tarkoma

Abstract

Mobile devices are evolving into hubs of content and context information. There have been many research projects showing the potential for new applications for pervasive computing. We aim to support pervasive applications on a wide variety of devices using Web and resource-based smart spaces. We address several issues for adapting a resource-based style of HTTP (REST) for pervasive services to enable easy mashup of applications in this environment. First, for security and access control in heterogeneous, dynamic environments we introduce a flexible access control mechanism on top of OpenID and OAuth. Additionally to support finding resources we use a search engine that can collaborate with existing service and network discovery mechanisms. We also outline how an emerging W3C standard, DCCI, can be used to share information within a device in an interoperable fashion.

Keywords: Pervasive computing, Smart Space, Security, Internet

1 Introduction

Mobile devices are evolving into hubs of content and context information. With this premise, we focus on pervasive applications in smart spaces that use locally available connectivity and discovery of devices. For example, this allows sharing content and offering services locally with direct connections between devices. While there have been many research projects showing the potential of such applications, results from this research have not yet become widely deployed. An important reason for this is interoperability --- a fundamental requirement for practical smart space applications. Smart applications are straightforward to demonstrate in a research lab environment with a small number of limited devices, but rather challenging in practice with hundreds of different devices with a large number of different operating systems and run time environments. This is a well-known challenge for ubiquitous and pervasive computing. For example, the survey [1] features 29 different pervasive computing platforms up to 2004.

The aim of this paper is to present our approach for addressing practical smart space deployment using Web technology. Web technology is becoming an integral part of mobile devices as consumers demand Web access. Moreover, the Web has already proven to be a highly interoperable software platform. In our proposed approach, devices offer and consume services using Web technology. While not every device will be able to offer services, they will be able to consume services and provide content and context information for these services. Furthermore, we see Web technology as an essential enabler for combining locally provided services with other Internet services. For instance, users may share geo-tagged content locally and load maps and other content from third party Internet services. The current success of application mashup technologies in Internet applications can be extended

to pervasive services in smart spaces. Currently, application mashup is typically done using HTTP in REpresentational State Transfer (REST) style, combined with RSS or Atom feeds.

Another important aspect is that the skills needed to create Web-based applications are now widely developed and there also exists an abundance of tools and developer and vendor support for these. Technologies, such as Web servers and content management systems, are currently becoming available on mobile devices as well.

Solutions for hosting Web-based services on mobile devices have been emerging recently, such as the Nokia Mobile Web server [2]. Additionally, recent popularity of push email and chat on mobiles shows the feasibility of semi-real time delivery of messages to devices. However, fully integrated and optimized solutions for mobile devices are still being developed. In addition, we have to take into account limitations of mobile devices, such as more limited memory, energy, and wireless transmission resources. A number of additional key requirements come from the mobile device and personal context, which is different from PC-centric devices. In this paper, we outline solutions for the following challenges in smart space applications:

- Web technologies need to be integrated with local smart spaces and made available on more mobile devices. While many approaches have used Internet technology, we use "Web 2.0" HTTP and REST style Web services as a middleware to mashup services, both between devices as well as on one device. Mashups also need to include local smart space artifacts like user context, content and devices.

- Resources in a smart space need to be able to find each other. The heterogeneous environment makes this resource discovery and subsequent communication between resources challenging. In our context, this needs to be integrated with the above.

- Privacy and security are major challenges for services and application mashups in smart spaces. Currently, application mashups are often poorly secured [4]. In mobile devices, we typically have sensitive information like location and other personal content. Devices in the smart space engage in many complex interactions with services on other devices and in the Internet that need to be properly secured. Widely used Internet practices, which often require manual entry of passwords and authorization of services, are not an adequate solution here.

- The Web browser model of consuming services does not allow one to access local context and content on the device. Fortunately, there are a number of upcoming standards that permit such context access from the web browser.

We have implemented a smart space shopping mall setup, which is open for public at the Nokia Showroom in the city of Oulu, Finland. This scenario demonstrates a number of services and integrates solutions for the issues outlined above.

Earlier approaches such as CoolTown [5] have applied basic Web architectures to ubiquitous applications based on PDAs. We now see the opportunity to use the traditional Web as well as "Web 2.0" technologies as a platform for providing smart space services. REST has been recently proposed for pervasive applications due to its simplicity and interoperability [7][8]. Additionally, Semantic Web, ontologies and SOAP-based Web services, and agent systems have been proposed as enablers in this environment [1][3]. Our approach advocates lightweight solutions that use established standards. We do not use SOAP-based Web services for reasons of complexity and lacking suitability for mashup applications. Many consumer Internet services omit SOAP support for this reason as well. A preliminary version of this work has been presented in [9].

2 Web and Resource based Smart Spaces

A smart space is a heterogeneous, local environment with many different devices with varying capabilities. A schematic picture is shown in Figure 1. Being Web-based, a smart space network includes devices capable of accessing, and optionally providing, HTTP based Web services. In practice this means most devices with a network connection (WIFI or telephone network) and TCP/IP support. For low-end devices without this capability, we utilize a proxy solution in which high-end devices, such as phones or access-points, proxy functionality of more limited devices such as for example sensors, Bluetooth peripherals, etc. We argue that with this approach we can target a very large portion of the current mobile and portable devices market.

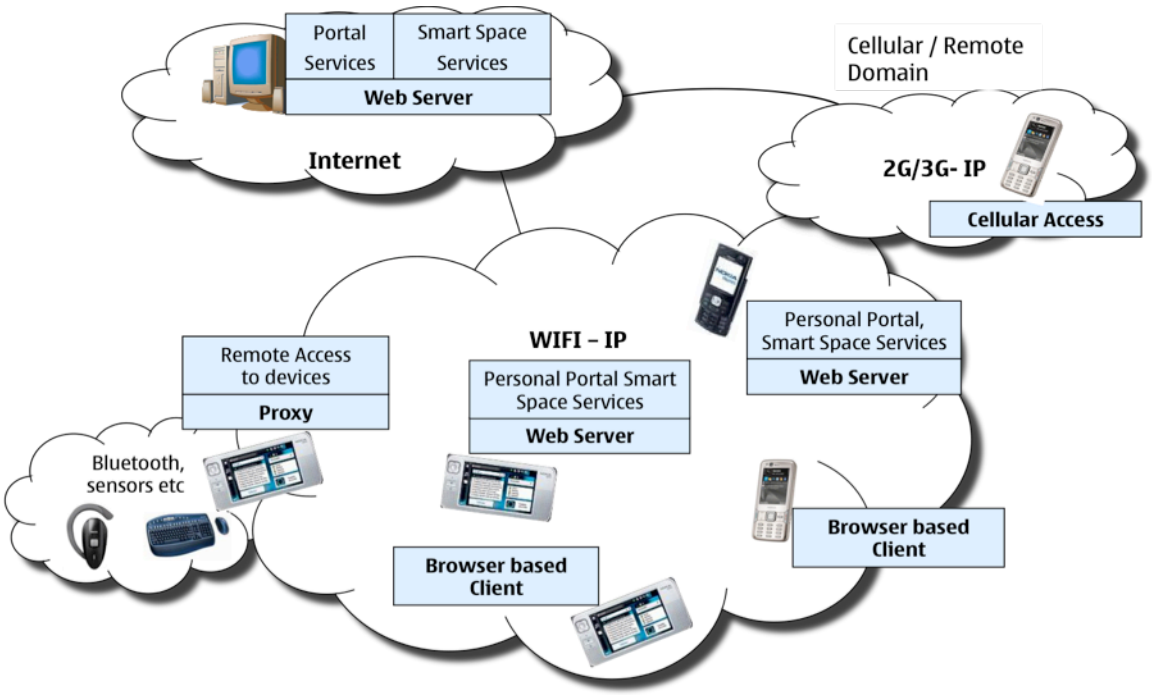


Figure 1. Smart Space Network Architecture

We distinguish between devices that host Web-based services and clients that only consume services. Clients are either Web browsers accessing a HTML/XHTML based Web application in the Smart Space or on-devices applications that access Smart Space services through their API. The key advantage of using HTML based UIs is that these can work well across devices with a sufficiently capable Web browser. Additionally, deployment is easier since software installation is not required. A native application may provide better usability, which comes at the cost of sacrificing portability and the need to install the software before it can be used.

Web services and Web applications are dependent on Web servers. The distinction is that a Web service provides an API whereas a Web application provides a browsable application UI (usually HTML based). Servers can be embedded in, e.g., a mobile phone or hosted in the Internet in a hosting environment. One advantage of running an embedded Web server in a mobile device is that such servers are capable of accessing local device functionality and resources. For example, the Nokia Web Server [2] comes with a portal that can access the camera and application data such as contacts and calendar. This Web server can be reached even if firewalls and NATs are present (e.g. in mobile operator networks) by employing a permanent connection to a proxy server which passes any incoming traffic to the device over the open connection. This is conceptually similar to how Javascript techniques such as Ajax can turn a passive web application in the browser into a message processing server. The proxy also provides a URL for the device.

Our software architecture for a mobile device hosting a Web portal is shown in Figure 2. The figure shows the Web platform and middleware services which are

discussed in this section. Distributed operation is realized by querying a search engine for services and accessing these services.

Smart Space services in our architecture provide REST APIs. This means that entities such as people, content, devices, services are resources with a URI that can be manipulated using simple operations provided by the HTTP protocol such as GET, POST, PUT, and DELETE. The main advantage is that it is suitably lightweight for implementation on limited devices and that it aligns well with existing APIs. Being Web and REST based has allowed us to reuse existing web components such as feed aggregators, content management software, Web application frameworks, databases etc in our shopping mall demo. Another key benefit of REST style Web services is that they can easily be supported in devices in our target market of mobile and portable devices.

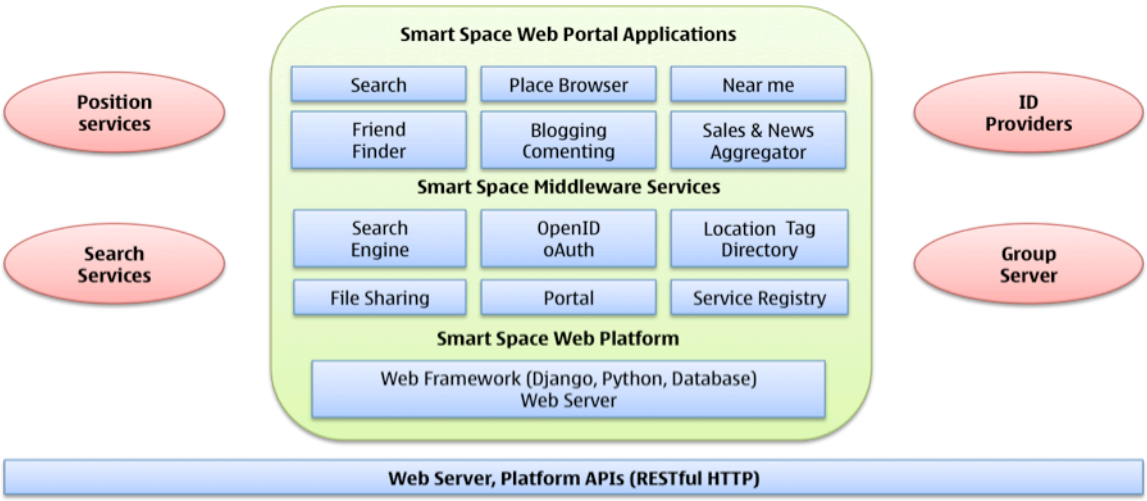


Figure 2. Software Architecture Overview

Locating Resources

Smart Space services are similar to location-based services (LBS) in the Internet in the sense that they are typically scoped to a particular physical space such as a room, a moving vehicle, a building, etc. However they are different in the sense that such spaces are not easily identified by simple attributes such as GPS coordinates.

In our setting, people, devices, and resources may come and go and they need to constantly adjust to this dynamic nature of the smart space. In order to be able to locate entities, indoor maps, points-of-interest, and indoor positioning infrastructure are required. In order to foster and support easy mashup of indoor LBSs, we have adopted solutions that are simple to integrate with existing Web technologies and applications, such as making services and their own geo-spatial data available through REST APIs and widely used formats such as ATOM feeds and GEO RSS. Our system was developed by integrating commodity Open Source GIS software and novel research prototypes realized within our labs, including a WIFI-based indoor positioning system [13].

As is common for indoor positioning systems, this positioning system represents locations using spatial relations between buildings, floors, sections and rooms. We represent these locations using a URL schema. For example, http://nokia.com/locations/helsinki/ruoholahti/4/A/401 references room 401 in one of the Nokia premises.

.Referencing resources using a URL is a concept that is native to the Web and an important aspect of the REST architectural style. It is also an important concept in Web 2.0 where content is often tagged with keywords. These keywords are typically part of a name space that is represented by a URL. The URL with /<keyword> appended is then used as a tag URL to refer to the tag (e.g. http://delicious.com/tags/smartspace). Resources can be indoor geo tagged with location URLs similar to this. For example, we indoor geo tag items in an Atom feed by simply including link elements with the rel="tag" attribute set.

The tag URLs refer resources in a location tag directory that provides meta information about the locations, such as GPS coordinates, name and description, etc. This directory can be manipulated using the IETF Atom Publishing protocol (RFC 5023), which is a REST based protocol for performing create, read, update and delete operations on resources that is complementary to the IETF RFC for Atom feeds (RFC 4287).

Finding Resources

A key issue for interacting with resources in the smart space is finding out their URL. There has been a lot of research on service registries and service discovery. While various solutions have been proposed today's Web and Web services function mostly without such solutions (aside from DNS, which is used for looking up domain names). However, a smart space is much more dynamic than the current Web and a service registry or discovery mechanism is needed here to be able to create mashups that use locally available resources.

In our system we use a combination of both and decouple the problem of finding resources from that of registering or discovering them. To find resources in the Smart Space, we use a search engine based on Apache Lucene (http://lucene.apache.org) that indexes resources, including their tags that encapsulate information about their type, location, ownership, etc. Discovering resources then becomes a matter of specifying the right tags in a search query and using the resulting resource URLs. The search service can be accessed through the internet or discovered in a local network via MDNS.

The search engine can be populated with resources in various ways: it can crawl networks for resources, use traditional discovery mechanisms such as UPNP or MDNS to discover resources in local networks, or it can be used as a traditional service registry (local or Internet based). In our prototype system we use a combination of these approaches. For example, we use discovery to add UPNP media server advertised content to the search service. Other services register

directly with the search service and consumers of resources look them up with simple queries to the search service.

Combined with the location tag directory, this solution provides a flexible system for tagging resources such as content objects, service end points and other information, with location URLs and retrieving by querying the search engine. We also use tagging for associating other semantics with resources. We use service type tags, content type tags, group membership tags, etc. Consequently, the technique is also suitable for non-localized, virtual smart spaces (e.g. clusters of devices and services belonging to a person, company or group of friends). Location is just one of many possible search criteria.

Protecting Resources

Ubiquitous context information, based on location and sensing of the surrounding environment, is typically privacy sensitive. Similarly, there is a need to restrict the ability to act upon the physical world and the entities within it. Consequently there is a need to authorize access to resources. There is considerable work on security and access control for pervasive systems and discovery protocols [14]. Our goal is to use existing, web based application mashup techniques, which also combine easily with existing Internet technologies and services. Howevr, current mash-ups atr unsecure [4] and do not support our setting with many distributed resources sufficiently well, as we discuss below.

Resources are accessed by different services on behalf of users. This includes resources owned by the user, other users, and resources provided by various services. A typical example is giving access to your current location, which is provided by some location service, to other services. This becomes more complex for a friends finder application that needs to access the locations of your friends on different location services. The application needs to authenticate with each location service and each of these needs to check authorization rules (i.e. are you allowed to see your friend's location).

We need a solution for this that is lightweight, simple and that does not rely on a centrally owned and provisioned identity solution. Additionally, it needs to integrate well with our resource based architecture, it has to scale to large groups of resources and to loose associations, and the solution needs to be easy to support across a wide range of existing devices.

There is considerable work in this area of security including for example, the Liberty Alliance Identity Federation Framework and several trust management systems such as SPKI/SDSI that are distributed in the sense that they permit separate, local name spaces. Access control is possible over several kinds of relations such as "kids of friends of Dad", where the local names are maintained locally. However, these solutions do not fully meet our requirements of being lightweight and mashup friendly. On the web, two key protocols have emerged recently, OpenID [11] and OAuth [12] that rely on a novel and more flexible, decentralized approach to

authentication and authorization and thus can provide the foundation for a solution that meets our requirements.

OpenID is a lightweight, single sign on, identity federation protocol that can be used by relying parties to authenticate users with an identity provider. To sign in, a relying party verifies whether the client that the user is using (usually a browser) has authenticated the user with the identity provider that the user specifies. An important aspect of OpenID is that it uses identity URLs to identify users. OAuth is a similar protocol for authentication that is used to allow services to authenticate to each other on behalf of a user. The result of a successful authentication with OAuth is a token that can be used to access the service. OAuth is especially popular for implementing mashup applications involving services from different service providers. Important is that OpenID and OAuth rely on URLs to specify trusted entities and hence nicely fit with our REST based architecture.

We extend these protocols to cover scenarios with large amounts of resources, people and services. To do so, resources are organized in groups, which themselves are resources. A group server is technically similar to the location tag directory discussed earlier and a group is simply a list of URLs. Each group has a public key and a private key. The central idea is to authorize access to resources based on group membership. The group server can issue a signed proof of group membership. This membership assertion is then included in requests to resources that can simply verify the signature.

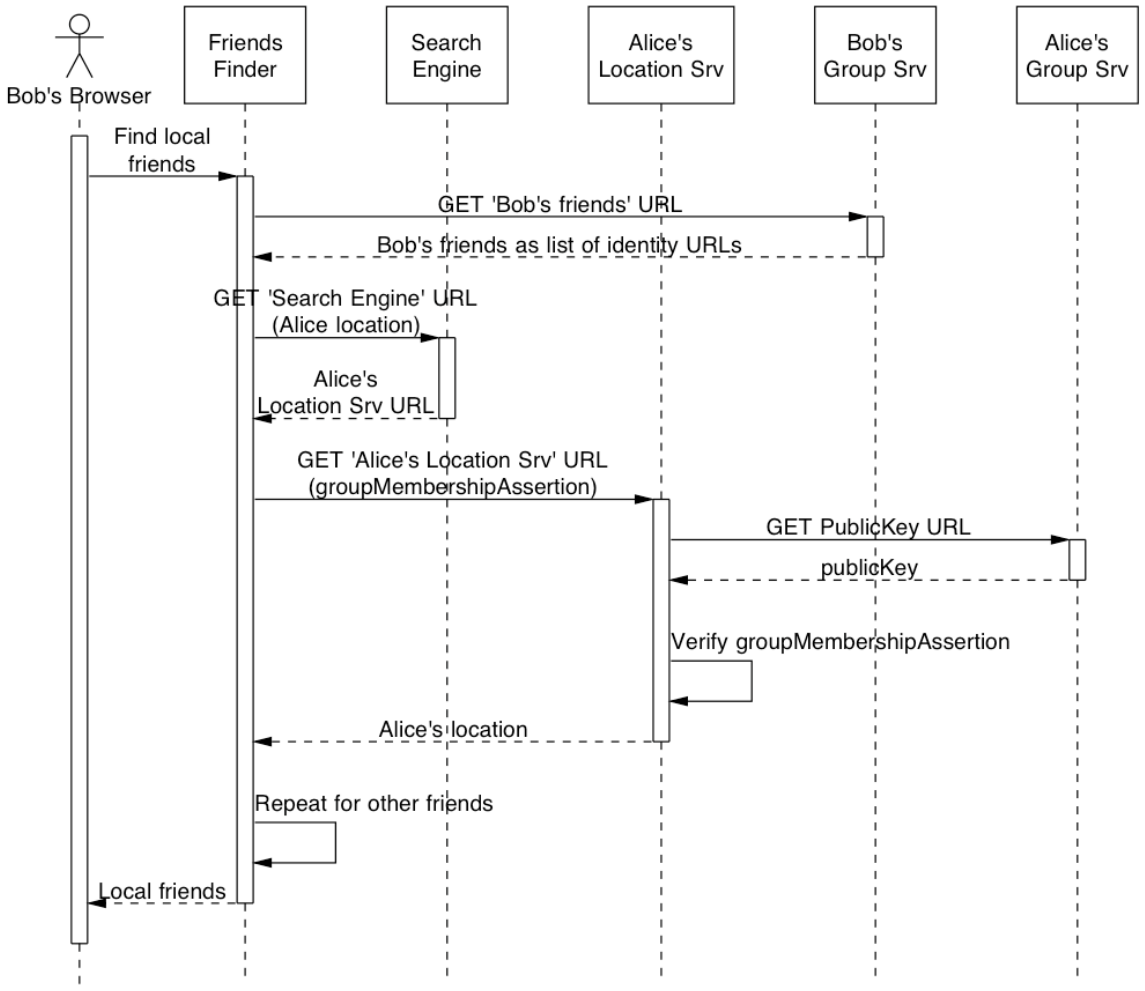


Figure 3 Sample Interaction for Friend Finder with multiple location and group servers

We use the group server mechanism in combination with the location tagging mechanism and search architecture outlined above to implement a system where it is possible to use groups and membership assertions to find and access resources that one is authorized to use. A modified OpenID provider keeps track of the user's group memberships and an OAuth like mechanism is used between services to pass on membership proofs. Group members can be people, represented by their OpenID URL, services represented by their service endpoint URL, or any other type of resource.

Figure 3 illustrates our approach with the friend finder example. The friend finder uses Alice's identity URL to query the search engine for her location service provider. Then it uses the membership token that proves that Bob is a friend of Alice to authenticate for the location service, which returns Alice's location after verifying the signature. It is assumed that the friend finder obtains membership tokens in advance in a secure way. A nonce is used to prevent replay attacks. Any public keys

fetched can of course be cached and, additionally, HTTPS certificates may be used to verify ownership of public keys that are being exchanged.

A key benefit of this approach is that it can vastly reduce the number of HTTP redirects in the browser that would otherwise complicate use of either OpenID or OAuth with more than a handful of resources. With OAuth, the browser would have to redirect to each resource individually, just to verify authentication. Therefore, using verifiable and cacheable assertions significantly reduces network overhead for clients, which is essential in a smart space.

The approach is decentralized since it decouples service providers from identity providers and group servers. This means that they can be flexibly reused with each other in a smart space. For example, a location service can discover the user's list of friends and their preferred identity provider and enforce a rule that friends of the user are allowed to access their location when another service is doing so on their behalf. Furthermore, it is very easy to adopt this protocol for service providers since it stays close to how OAuth and OpenID work. Finally, it reduces the number of user interactions needed (e.g. permissions and confirmations) by relying on assertions instead.

Appropriate levels of security and trust are achieved in a similar manner to other public/private key based security mechanisms that have been proposed in the ubiquitous computing research community. Trust is decentralized in our system and very much relies on choices made by, e.g., identity providers and users regarding which groups to trust, which services to trust, etc. Given that a smart space is too dynamic and heterogeneous to orchestrate this centrally, we claim that this is a suitable way to help the smart space participants self organize along alliances, friend relations, etc.

Context Information for Browser Based Applications

While not all devices will be able to provide services, most will be able to consume services. Adaptive web applications used on such devices need access to local device and user context information. In order to be able to support this local data sharing, we have experimented with W3C DCCI (Delivery Context Client Interfaces) [10]. DCCI is based on W3C DOM (Document Object Model), which is an API with an underlying tree representation for accessing and manipulating HTML/XML documents. DCCI extends the standard DOM tree and API to provide a context tree where each context source has its own node representation. A directed event propagation model based on DOM events is provided to allow remote capture of specific events and change notifications from other devices. Additionally, static and dynamic properties are supported (local or remote). DCCI provides only the access API for consumer applications. It is up to the implementation to provide additional modules within the framework that addresses functionalities such as property management, security, and connectivity for context providers to the model. The main benefit of using DCCI for pervasive smart space applications is that it provides

easy navigation, search and query for static and dynamic properties and can support publish subscribe mechanisms for local and remote context changes.

We have implemented DCCI as an extension to the Mozilla based MicroBrowser on the Maemo platform for Nokia Linux Tablet N810 that can easily be added to other Mozilla based browsers. This extension provides dynamic location information and static information such as screen size and width. The "Location" parent node has two child nodes "Location-GPS" and "Location-Indoor" that are linked with GPS and indoor location respectively. The "property change" event is used to send notifications whenever the location changes. By using a single event listener for parent "Location" node, an application (script) can monitor all location nodes below the parent "Location" node.

The extension also supports remote DCCI devices. This can be used to create an adhoc smart space over Bluetooth. When combined with the proxies illustrated in Figure 1, we can expose local context to non-DCCI capable devices in the smart space. The hierarchical nature of DCCI trees is utilized to construct a composite tree structure that includes local copies of DCCI trees of remote devices that are synchronized using events. Web applications see the composite smart space context reflected in the local DCCI tree. In practice, the DCCI context tree of each device may be different as a result of e.g. different access rights or different needs of locally running applications.

Our analysis of the implementation has revealed a few drawbacks with the DCCI approach:

- Since DCCI is derived from the DOM API, implementations have to support all DOM Element (and thereby Node) methods, including those intended for document manipulation. Consequently only a subset of the API is relevant for context representation models. However, for compatibility reasons an implementation must still support the full API.

- A simplified event model would suffice as opposed to the full DOM event support. The DOM Event mandates the implementation of event capture, target and bubble phases.

- DCCI provides an API to read context information provided by various context services; however, applications currently lack standardized means to pass information to these services (e.g. for configuring update frequency or other service specific parameters).

- There is no standard ontology that DCCI context trees conform to. Such agreed semantics would be needed to ensure compatibility of DCCI context between applications. The W3C UWA working group is developing an ontology to resolve this issue.

- The lack of a security model for DCCI is inhibiting factor for wide spread adoption. However, DCCI is intended to be used with additional frameworks

that can address this and other issues. For example, the resource access control mechanism discussed in this paper can be used for this. The DCCI model would have to be annotated with access rights and could also be used to store membership assertions for the user.

Despite these current limitations, DCCI is a forward looking approach with its support for dynamic properties, access to data providers through a unified API, dynamic topologies and representation that separates syntax and semantics (through use of ontology) that provides many advantages for future smart spaces.

3 Concept Evaluation and Experiences

In order to gain first-hand experience on the applicability of the Web-based smart space approach in a practical setting, we developed a shopping mall smart space demo. This has been running non-stop for several months in a Nokia Showroom in Oulu, Finland.

The user interface (Figure 4) presents the users with a Web portal served from a mobile web server that integrates a set of service portlets. The portlets represent distinct services in the system that are found using our resource search solution. The integration of a Web based UI with local service discovery ensures that each user is presented with a dynamic, up-to-date representation of the contents and services from devices participating in the smart space. The main services include a shop directory component, per shop sales feeds, aggregated sales and news, portal finder, friend finder, commenting, media sharing and smart space search. Users are able to come and go and their presence in the smart place is reflected by their device portals showing up on the list of visible portals in the main screen of the portal running on their device.

Creating additional services is similar to creating ordinary web applications. Security is integrated into the application framework (in our case python Django) and resource discovery for easy mash-ups are supported with a python library.

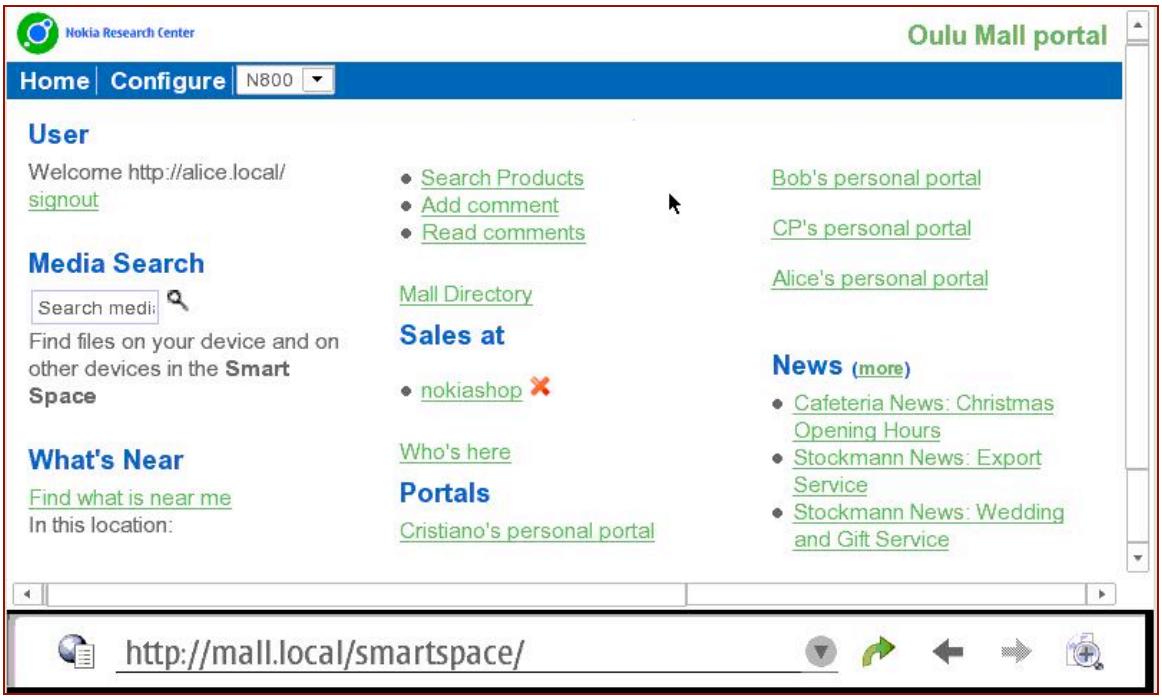


Figure 4 Composition of discovered services on a web based portal UI

Our main findings and feedback we got from users are:

- Location-aware maps and location-based services are good concepts that users would like to see more of.
- People were generally able to accomplish simple tasks since using the system is similar to using a normal browser based Web service.
- Responsiveness of the UI is important, especially for blocking operations involving e.g. network discovery. This suggests that we should handle the discovery asynchronously, which with our search engine approach is possible.

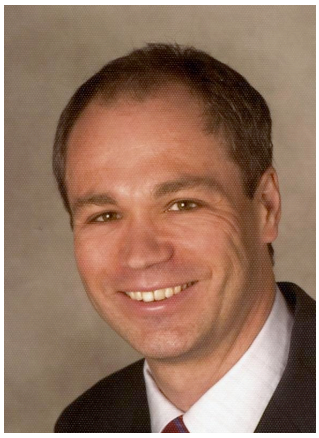
4 Conclusions

The main goal of this work has been to establish Web technologies as a middleware platform for pervasive services in smart devices and utilizing a number of open source components. We have outlined the key challenges for practical smart spaces and provided solutions for these in the form of a REST-based framework that includes support for location models, finding relevant resources, authorizing access to resources with group based security, and sharing context information locally through W3C's DCCI. Furthermore, we have demonstrated the feasibility of providing pervasive smart space services by implementing a prototype smart space around. The user study we have done has confirmed that this approach can be acceptable to users, despite some usability issues.

Our central claim is that the presented solutions may help to achieve the vision of mass market pervasive services. The REST based approach advocated in this paper can be applied easily in the context of existing Web based architects and it can be used on existing browsers on mobile devices. The key remaining challenges involve resolving usability issues such as those encountered in our study, and establishing a widely used security and resource finding solution as proposed in this article as well as spreading the use of technologies such as DCCI on the client side.

Authors

Christian Prehofer



Nokia Research Center

Itämerenkatu 11-13

00180 Helsinki

Finland

+358-504869882

christian.prehofer@nokia.com

Christian Prehofer is Senior Manager in Nokia Research. In the last ten years, he held different management and research positions in the mobile communication industry. He obtained his Ph.D. and his habilitation in computer science from the TU Munich in 1995 and 2000. He is author of more than 80 publications and 20 granted patents, and he also serves in several program comitees for conferences on networking and software technolgoy. He has been lecturing about software technology for communication systems at TU München.

Jilles van Gurp

Nokia Gate5

Invalidenstrasse 117

10115 Berlin

Germany

+4915155155757

jilles.vangurp@nokia.com

Dr. Jilles van Gurp received a licentiate degree from the Blekinge Institute of Technology (2001) in Sweden and a Ph. D. Degree from the University of Groningen in the Netherlands (2003). Since 2005 he has worked in Nokia first as a industrial researcher in the Nokia Research Center in Helsinki Finland and currently as a software architect in Nokia Gate5 in Berlin. During his research career he has published over 30 peer-reviewed articles in journals, conference & workshop proceedings on topics including object oriented frameworks, software architecture, software product lines and pervasive computing.

Vlad Stirbu

Nokia Corporation

Visiokatu 1

33720 Tampere

Finland

+358-503860572

vlad.stirbu@nokia.com

Vlad Stirbu is a member of Research Staff at the Nokia Research Center in Tampere.

Sailesh Sathish

Nokia Corporation

Visiokatu 1

33720 Tampere

Finland

+358504835679

sailesh.sathish@nokia.com

Sailesh Sathish is a Senior Researcher at Nokia Research Center, Finland. His main interests are context models and architectures, especially their relations to mobile web. His other interests include adaptive frameworks, user interest modeling and web technologies. He also represents Nokia within Ubiquitous Web Architecture group of W3C and has editor status. He holds a Licentiate of Philosophy degree from Tampere University, Finland, Master of Science from University of Bristol, UK and Electronics Engineering Degree from University of Cochin, India.

Pasi P. Liimatainen

Nokia Corporation

Visiokatu 1

33720 Tampere

Finland

+358504860077

pasi.p.liimatainen@nokia.com

Pasi P. Liimatainen works as a Principal Member of Engineering Staff at Nokia Research Center. He has a 15+ year career in various software R&D positions at Nokia and Patria, and holds an M.Sc. degree from the Tampere University of Technology.

Cristiano di Flora

Nokia Corporation

Hatanpäänkatu 1,

Tampere (FINLAND)

+358 (0)50 4872919

cristiano.di-flora@nokia.com

Dr. Cristiano di flora is a Software Architect in Nokia Devices R&D, Maemo Software, Tampere (Finland). He holds a Ph.D. Degree in Computer Engineering from the "Federico II" University of Napoli (Italy).

Sasu Tarkoma

University of Helsinki

Department of Computer Science

Gustaf Hällströmin katu 2 b (PL 68)

00014 Helsinki

Finland

+358-503841517

sasu.tarkoma@cs.helsinki.fi

Sasu Tarkoma received his M.Sc. and Ph.D. degrees in Computer Science from the University of Helsinki, Department of Computer Science. He has been recently appointed as full professor at University of Helsinki, Department of Computer Science; he is also currently professor at Helsinki University of Technology, Department of Computer Science and Engineering. He has managed and participated in national and international research projects at the University of Helsinki, Helsinki University of Technology, and Helsinki Institute for Information Technology (HIIT). He has worked in the IT industry as a consultant and chief system architect, and he is principal member of research staff at Nokia Research Center. He has over 100 publications, and has also authored two recent books.

References

[1] C. Endres, A. Butz, A. MacWilliams, A survey of software infrastructures and frameworks for ubiquitous computing, January 2005, Mobile Information Systems, Volume 1 Issue 1, 2005.

[2] Nokia Mobile Web server, http://mymobilesite.net/. Accessed April 2009.

[3] X. Wang, J. Song Dong, C. Chin, SankaRavipriya Hettiarachchi, Daqing Zhang, "Semantic Space: An Infrastructure for Smart Spaces," IEEE Pervasive Computing, vol. 3, no. 3, pp. 32-39, July-September, 2004.

[4] G. Lawton, "Web 2.0 Creates Security Challenges," IEEE Computer, vol.40, no.10, pp.13-16, Oct. 2007.

[5] P. Debaty, D. Caswell, Uniform Web presence architecture for people, places, and things, IEEE Personal Communications, Vol.8, Iss.4, Aug 2001, Pages:46-51, 2001.

[6] S. Sathish, A. Brodt, D. Nicklas, B. Mitschang, "Design and Implementation of a Platform for Location-aware Mashups on Mobile Devices". In proceedings of the Ninth International Conference on Web Information Systems Engineering (WISE 2008), Auckland, New Zealand, September, 2008.

[7] D. Coffman, S. McFaddin, C. Narayanaswami, D. Soroker, J. H. Han, H. K. Jang, J. H. Kim, J. K. Lee, M. C. Lee, Y. S. Moon, Y. S. Paik, J. W. Park, "Modeling and Managing Pervasive Computing Spaces using RESTful Data Services", IBM technical report, RC24344, 2007.

[8] W. Drytkiewicz,, I. Radusch, S. Arbanowski, R. Popescu-Zeletin, "pREST: a REST-based protocol for pervasive systems," Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on , vol., no., pp. 340-348, 25-27 Oct. 2004.

[9] C. Prehofer, J. van Gurp, C. di Flora Towards the Web as a Platform for Ubiquitous Applications in Smart Spaces, Second Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI), at UBICOMB 2007, Innsbruck, 16-19 September, 2007.

[10] K. Waters, R. A. Hosn, D. Raggett, S. Sathish, M. Wome, Delivery Context: Client Interfaces (DCCI) 1.0, W3C Candidate Recommendation, Dec, 2007, http://www.w3.org/TR/DPF/.

[11] B. Ferg et al., OpenID Authentication, http://openid.net/specs/openid-authentication-2_0.html, 2007 .

[12] M. Atwood, et al., OAuth Core 1.0, December 2007. http://oauth.net/core/1.0/ .

[13] C. di Flora, C. Prehofer, Leveraging GIS Technologies for Web-based Smart Places Services, 6th IFIP Workshop on Software Technologies for Future Embedded & Ubiquitous Systems, 2008.

[14] F. Zhu, M. W. Mutka, L. M. Ni, Service discovery in pervasive computing environments, Pervasive Computing, IEEE, Volume: 4, Issue: 4, 2005.