Experiences with Realizing Smart Space Web Service Applications

Jilles van Gurp, Christian Prehofer, Cristiano di Flora
Nokia Research Center
Email: [jilles.vangurp|christian.prehofer|cristiano.di-flora@nokia.com]

***Abstract*— This paper presents our approach for building an internet based middleware platform for smart spaces as well as a number of services and applications that we have developed on top of it. We outline the architecture for the smart space middleware and discuss several applications and services that we have so far realized with this middleware. The presented material highlights key concepts in our middleware vision: services are HTTP based and restful; applications are accessed through a browser so that they are available on a wide variety of devices; and we demonstrate the concept of bridging non internet enabled smart space devices to our IP and HTTP centric smart space network.**

I. Introduction

This paper focuses on middleware for ubiquitous applications in — what we call – smart spaces. A smart space is a multi-user, multi-device, dynamic interaction environment that enhances a physical space by virtual services [1]. These services enable the participants to interact with each other and other objects in a P2P way in the smart space. The research in the area of ubiquitous and pervasive computing has led to many interesting research demos and usage experiences. Building on widely spread wireless devices such as phones, PDAs and other special purpose devices, there is an enormous potential to create new smart space services and applications.

However, despite many efforts in industrial and academic research [1], [2], [3], [4], [5], the mass market uptake has been very sluggish. We outlined several reasons for this in our earlier paper on this topic [14], which include: use of disruptive/experimental technology; lack of maturity of research prototypes; lack of proper development environments; and the fact that most of these systems tend to be one of a kind verticals optimized for a particular use case or research problem. The purpose of this paper is to outline our approach to address this issue: we want to build smart space software systems that have a good perspective for mass market adoption.

While mass market adoption of ubicomp technology has so far been weak, there has been an astounding growth of web-based, Internet applications during the last 15 years. Most recently this has occured in the areas of web services and Web 2.0 [9]. What has enabled this progress is not just the web browser and the agreement on some key (de-facto) standards, but also a wealth of development tools, content management systems and other internet centric tooling. For instance, even end-users can now easily create content on many internet sites.

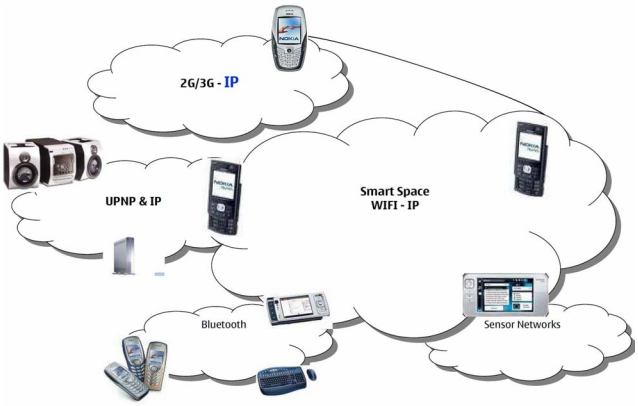

Fig. 1. Local IP network with smart devices acting as proxies

Because this is so easy, there is now a wealth of user generated content on the internet.

Consequently, our approach builds on these technologies. By reusing the technologies that enable the type of collaborative and social services associated with the Web 2.0 phenomenon, we enable applications and services in the smart space. This means that mobile devices can host web services and other devices can connect directly, thus enabling a P2P network. The contribution of the paper is to present such an architecture and middleware on mobile devices, and to show how it can be realized with current devices and open source software.

Earlier approaches like CoolTown [**?**] have applied basic web architectures to ubiquitous applications based on PDAs. We now see the opportunity to use these traditional as well as new Web 2.0 technologies as a platform for providing services to ubiquitous devices [6], [7]. Recent advances in device technologies allow us to bring both the key web standard software and development tools to widely deployed personal devices (mobile phones, PDAs). For instance, many mobile devices can not only surf the Internet, but can also provide web services based on mobile web servers.

II. Architecture and Key Design Decisions

This section summarizes our Smart Space architecture from [14]. The Smart Space Network (see Figure 1) is based on a decentralized, local, IP-based network which includes devices and services provided by the smart space owner. These devices

1-4244-1457-1/08/$25.00 © IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

include PC-class devices and high-end mobile devices that can connect to a WLAN, have a browser, and have the ability to run a web application server. The web server is needed for hosting smart space services. The device base can also span other devices such as low end phones without the mentioned capabilities or non personal networked devices such as bluetooth accessories, UPnP appliances and sensors.

While all these devices can be part of the smart space network directly or indirectly, only the high end devices in the classification above have the ability to run custom service software. Consequently, we consider these devices to be the backbone of our smart space architecture. Sensor/actuator networks are often connected to high end devices not by IP networking but through technologies such as Wibree or Bluetooth that may also be used for connecting to consumer electronics. To integrate such devices in the smart space, their services are published by and accessed through technology-specific proxies running on a high-end device. This means that a high-end device accesses the sensor using the sensor-specific interconnection technology and exposes its features to the rest of the smart space.

Devices joining the smart space network become part of the smart space. However, to access services and web applications in the smart space network, a device needs to discover their existence and their network addresses. In the local smart space we use the Zeroconf mDNS mechanism [10], [11], which is similar to DNS and integrates into the hostname resolution at the OS level. This also supports service registration and discovery. This design choice is further motivated in our earlier work [14].

By using mDNS, devices can advertise their name within a locally scoped namespace, e.g., device A is available at deviceA.local address. Such a locally scoped symbolic hostname may then be transparently resolved by other devices in the smart space. Indeed, since mDNS integrates into the OS in the same way as DNS does, existing software such as web browsers or web service clients can use the .local hostnames without any modification. Additionally, devices can make services available through such locally scoped web server URLs. For example, a photo website offered by the mydevice.local device could be accessed at `http://mydevice.local/photos`.

Figure 2 provides an overview of the system architecture of our web based infrastructure for smart spaces. The bottom layer contains a IP based network protocol stack and the Zeroconf based discovery mechanism. The base platform and communication layer on top of that contains components that we see as necessary to realize a full web platform in the smart space.

Most of these components are based on existing software already available in the market such as web servers and databases. The components in this layer are used to run web services, web applications and for hosting multimedia content. Using off-the-shelf web components allows us to bring many features to the smart space such as, for example, user management and security solutions, instant messaging, and

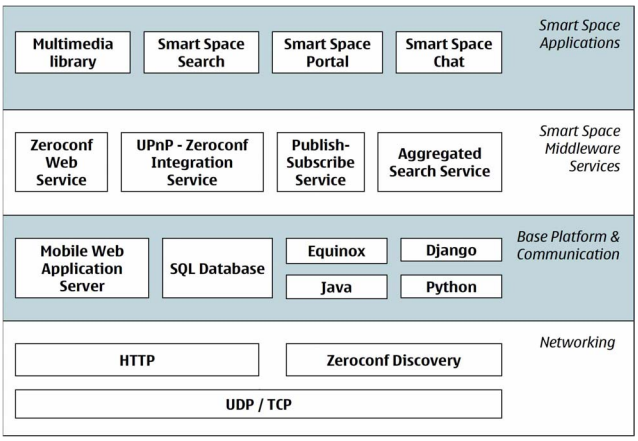

Fig. 2. System Architecture

asynchronous communication infrastructures. A key difference with a normal web server is that services and applications running on a mobile device (e.g. a python script or OSGI Java components) can access the Zeroconf service discovery mechanism to integrate other services in the smart space as well as native device features such as the phone camera, contacts, bluetooth, etc.

On top of these two layers we have realized several smart space services and web applications. These demonstrator applications, and the services they depend upon (third layer), are discussed in more detail in the next section. Compared to other work, the key design decisions that characterize our approach are as follows:

- **Bridging/proxying non IP devices.** As we focus on IP infrastructure, we also run technology-specific proxy components on smart devices that connect to non IP devices to access the features and services of these devices and offer them to the rest of the smart space.
- **Zeroconf naming and service discovery.** The infrastructure cannot rely on central facilities (such as DNS) to address naming of devices in the smart space. Zeroconf mDNS is designed to solve this issue and integrates into the operating systems hosting it, so as to allow existing software to use it without requiring any modification.
- **HTTP.** HTTP and REST-like web services are used as the primary means for integrating software across devices in the smart space, similarly to what currently happens on the Internet, where HTTP forms the cross platform glue that allows mash-ups across the extremely heterogeneous network of the Internet.
- **Reuse of existing web technology.** As mentioned in Section I, a key problem with existing solutions in the ubiquitous and pervasive computing research community is that they are rarely reusable. By relying on existing web technology, the infrastructure opens up to a large number of devices already available in the current market.
- **Multiple software run-times.** After years where mobile development platform choice was limited to J2ME and

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

C/C++, high end mobile devices are now offering devices a much wider range of technologies, including the already mentioned support for Python and other scripting languages. In this way, platform developers can use several run times. This means that many existing components used on the web can be used in a smart space context as well.

III. Smart Space Web Applications

As a proof of concept for our smart space concepts, we have developed middleware services and a few demo applications for the Nokia N800 internet tablet. The N800 is a Linux based device equipped with a touch screen, a WLAN device and various end user applications, including a web browser. The device runs Nokia's Maemo operating system, which is a Debian Linux derived system for which many open source software packages are available. We have realized an implementation of the architecture on the N800 by integrating open source packages. The most important of these packages include:

- **Avahi** This is an implementation of mDNS for linux that integrates seamlessly into Linux. With Avahi installed, applications are able to resolve Zeroconf .local host names. Additionally, avahi includes commandline tools for discovering and publishing services.
- **Lighthttpd + FastCGI enabled scripting languages** Lighthttpd is a lightweight webserver that can run on the N800. Using FastCGI, scripting environments such as Python and PHP can integrate into the webserver. The web applications discussed below are implemented in Python on top of the Django framework.
- **SQLLite** This is a so-called embedded SQL database that, unlike non-embedded databases, does not require running a separate database server but can be accessed in-process using a library, which reduces overhead significantly.
- **OSGi service container** In addition to the web server, a separate Java based service container is used. This service container is based on the Eclipse Equinox OSGi framework which runs in a Java CDC virtual machine.

As can be seen from this small summary, there is some overlap in functionality and room for optimization by removing packages. The criteria for package selection was merely availability of software components on the N800 to realize the applications and services discussed below and not to provide a fully integrated, one size fits all smart space platform. We envision that future smart spaces, similar to the internet, will use a much wider variety of software on many devices in many different compositions and configurations. We are already planning to move beyond the N800 and are considering components for Nokia Series 60 phones for example where we have a port of apache and python available [8] as well as many interesting native features that are integrated into S60 Python (e.g. access to phone camera; contacts and calendar).

Fig. 3. Web portal to smart space

A. Zeroconf Web service

As discussed in Section II, the smart space concept depends on Zeroconf for publishing and discovering services. However since Zeroconf is not a very web like technology and since Zeroconf is also not available on all devices, we have chosen to implement a web service wrapper around the Zeroconf functionality provided by Avahi. The API of this service is based on REST (Representative State Transfer) [13]. A list of known hosts and services can be obtained by a simple HTTP GET on `/hosts` and `/services`. This will return an XML formatted list of hosts or services known on that device. Using parameters such as type and host, the list of services may be filtered. Given a serviceid, a description of the service can be accessed by doing a HTTP GET of the `/services/$<$serviceid$>$` url. In order to post services, one simply posts the details to `/services`.

The web based publishing and discovery is used by the other services in Figure 2. Consequently, our services and applications are not dependent on Zeroconf directly. This makes it possible to access and run them on devices that do not have Zeroconf simply by accessing Zeroconf through a webservice on a device in the network that does have Zeroconf support. Additionally, by replacing the service implementation, we can remove the dependency on Zeroconf entirely.

B. Smart Space Portal

To the user, the smart space appears to be just a set of browsable web sites. Like normal web applications, these sites are hosted on web servers. The only difference is that in addition to regular web servers on the internet, there are also web servers running on devices in the smart space. Like on the internet, a key problem is finding interesting web pages and 'starting' points for browsing. This problem is solved using the concept of a smart space portal. The portal concept is illustrated with a screenshot of the N800 browser in Figure 3. The browser is displaying a web page on *foobar.local*, which is the Zeroconf name of a N800 device in our lab. The owner of this device may visit this page at any time by redirecting the browser to it (e.g. using a bookmark). The web page is dynamically created by a web application running on python

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

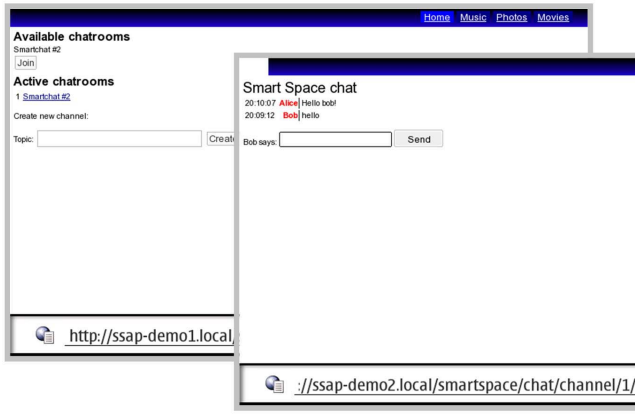

Fig. 4. AJAX Chat Application

Fig. 5. UPnP Browser

and the Django application framework. The implementation makes use of the Zeroconf Web service to discover other portal websites.

On the page, several links to similar portals on other devices are listed as well as links to music, photo and video content on the local device that the user can choose to share. Additionally, the portal integrates a search feature. As can be seen in the screenshot, there is an option to search content locally as well as in the smart space. When searching locally, the search service accesses a search engine that we have developed for the N800. This search engine, which is based on Apache's Lucene project, indexes content on the device. This search service is published through Zeroconf web service. When using the second button, the search service works by discovering all the search services in the network and aggregating their results.

Finally the portal has a few links for latest movies, photos and music. These links use a predefined query (e.g. mimetype: audio/mpeg) to search the smart space. The results are ordered by timestamp. A nice use case for this is to browse for example the latest photos taken by people in the smart space at an event.

C. Chat

A second application we have realized is a simple, IRC like web based chat client. This client uses a light weight REST based publish-subscribe mechanism that we have implemented on top of Zeroconf. Similar to the Zeroconf Web Service, this service is also based on REST principles. Channels can be created by sending an HTTP Post message to `/smartspace/chat/channel/1/`. This results in the creation of a message feed which may then be accessed by doing an HTTP GET on the feed url, which returns an Atom feed with the latest messages in a feed with `id=1`. In addition to the id, a uuid and a type are advertised for the feed as well. The uuid is a required Atom field. Posting a text message to a feed url results in the addition of the message to the feed.

Clients can advertise their interest in channels by publishing a notification service. The service advertisement includes an attribute to specify the feed uuid. When a message is posted, the feed service discovers all advertised notification services with its uuid and posts a notification message to them (which includes the feed url).

The screenshot in Figure 4 demonstrates a simple AJAX chat client based on these services. Users can create a channel and subsscribe to existing channels. In the screenshot, a user has created a channel and posted a message. The same mechanism can be used to implement asynchronous application messaging.

D. UPnP Media Integration

A third application we have created demonstrates the concept of bridging discussed in Section II. Unlike the other two applications, this application is currently not fully integrated into the portal.

Figure 6, presents an overview of how it works:

- A control device with UPnP and Zeroconf capability acts as a bridge between the smart space network and UPnP devices in the network.
- The device discovers UPnP devices and exposes them to a REST-ful API. Using this API, XML descriptions of device and service descriptions may be accessed.
- A bridge component uses the UPnP REST API to discover UPnP devices and advertises their REST service end points using the Zeroconf Web Services
- Finally applications discover the advertised services and consume them.

In Figure 5 a screenshot of a AJAX UPnP browser is displayed. The application discovers advertised UPnP devices and allows the user to browse their description and services. In the screenshot two UPnP devices have been found and the description for one of them has been expanded (a UPnP Media server running on a Nokia N95).

IV. Conclusions

This paper presents early results from currently ongoing work at Nokia Research Center to create a web based platform for smart space applications and services. In an earlier article [14], we already presented requirements and the key design decisions we summarize in Section II. Our vision is that smart spaces will extend the current internet onto people's devices

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.

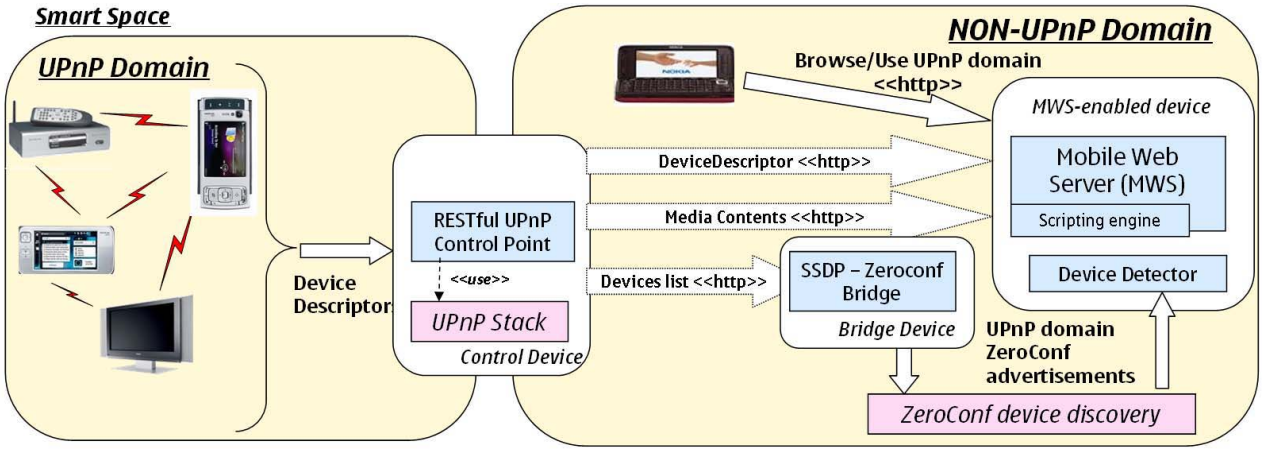

Fig. 6. UPnP Zeroconf Integration

where smart space services will integrate the local space into the internet, thus truely creating an internet of things. To enable such an internet of things, it is critical to align well with internet technology. Consequently, we make use of already available technologies such as web servers, search engines, discovery mechanisms, etc. We aim to minimize the amount of new technology needed since it is our vision that while disruptive conceptually, it can be realized without requiring users to update devices and software. Currently, mobile phone vendors are already selling WIFI enabled devices with advanced networking and browsing capabilities.

We have presented several smart space applications and services that we have so far implemented on top of our Smart Space Middleware Service platform. The software illustrates key concepts from our vision. The applications and services integrate over HTTP, which we see as the best way to span as much mobile devices as possible. Our services implement REST-ful APIs, which makes them light weight and easy to consume from applications, services and even in browser side Javascript (e.g. the chat and UPnP browser). Finally our applications can be accessed through a (mobile) web browser thus making them accessible on a wide range of devices.

This paper covers a lot of material and is very much ongoing work. We are currently layering many interesting features on top of the outlined technologies. Additionally, we are trialling in various use cases and expanding our middleware to support this. For example, we are currently working on a authentication and authorization that like what we present in this paper, builds on internet technology.

ACKNOWLEDGMENT

The authors are grateful to the project team at Nokia Research Center for many discussions and insights leading to this approach. We would particularly like to mention our colleagues Jaakko Kyro, Kari Ahvanainen, Heikki Matila and Pasi Liimatainen who have been involved with both conceptual work and implementation of software in this project.

REFERENCES

[1] Wang, X., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., Zhang, D., Semantic Space: an infrastructure for smart spaces, IEEE Pervasive Computing, 3(3) pp. 32-39, 2004.
[2] Abowd, G. D. Mynatt, E. D., Designing for the human experience in smart environments. In Cook, D. J. and Das, S. K., eds., Smart Environments: Technology, Protocols, and Applications, pp. 153-174, Wiley, 2005.
[3] Kaasinen, E., Niemela, M., Tuomisto, T., Valkkynen, P., Ermolov, V., Identifying User Requirements for a Mobile Terminal Centric Ubiquitous Computing Architecture, Proc. of International Workshop on System Support for Future Mobile Computing Applications, pp. 9-16, IEEE Computer Society, 2006.
[4] Coen, M., Phillips, B., Warshawsky, N., Weisman, L., Peters, S., Finin, P., Meeting the computational needs of intelligent environments: The Metaglue system. In Proceedings of MANSE'99, Dublin, Ireland, 1999.
[5] Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S, EasyLiving: Technologies for intelligent environments. In Proceedings of Second International Symposium on Handheld and Ubiquitous Computing (HUC 2000), Bristol, UK. Springer, LNCS1927, 2000.
[6] Schroth, C., Janner, T., Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services, IT Professional, 9(3), pp. 36-41, 2007.
[7] Yamakami, T., MobileWeb 2.0: Lessons from Web 2.0 and Past Mobile Internet Development, in Proceedings of International Conference on Multimedia and Ubiquitous Engineering, 2007.
[8] Mobile Web Server, Raccoon, `http://wiki.opensource.nokia.com/projects/Mobile\_Web\_Server`, 2007.
[9] O'Reilly, T., Web 2.0: Compact Definition, `http://radar.oreilly.com/archives/2005/10/web\_20\_compact\_definition.html`, 2005.
[10] Guttman, E., Microsyst, S., Autoconfiguration for IP Networking: Enabling Local Communication, IEEE Internet Computing 5 (3), pp. 81-86, 2001.
[11] Engelstad, P., Van Thanh, D., Jonvik, T.E., Name resolution in mobile ad-hoc networks, in Proceedings of the 10th International Conference on Telecommunications, ICT 2003., IEEE Computer Society, 2003.
[12] UpnP Forum, UpnP Device Architecture 1.0, July 2006, `http://www.upnp.org/resources/documents.asp`, 2006.
[13] Fielding, R.T., Architectural Styles and the Design of Network-based Software Architectures, University of California, 2000.
[14] Christian Prehofer, Jilles van Gurp, Cristiano 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 Sebtember, 2007.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2008 proceedings.