Christian Prehofer · Jilles van Gurp · Cristiano di Flora

Towards the Web as a Platform for Ubiquitous Applications in Smart Spaces

Abstract We introduce our web based middleware for smart spaces, which strongly relies on technologies used in Internet services. Following the key requirements and technologies, we present our architecture for ubiquitous applications in smart spaces. It exploits and leverages many of the key web-technologies as well as "Web 2.0" collaborative and social Internet services, including browsers, web servers, development tools and content management systems. In this way, we aim to make many of the highly disruptive ubiquitous applications less disruptive from a technology point of view. Furthermore, we discuss a number of new challenges for applying these technologies in ubiquitous applications. These include the areas of discovery/delivery of services, security, content management, and networking.

1 Introduction

This paper focuses on software infrastructure 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 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. Despite many efforts in industrial and academic research [1–5], the mass market uptake has however been very sluggish. At the same time, we have witnessed an astounding growth of web-based, Internet applications in the last 15 years. Most recently this has been around the areas of web services and Web 2.0 [21]. 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 and content management systems. For instance, even end-users can now easily create content on many Internet sites.

Our approach aims at smart space applications by building on the above technologies developed for the web based Internet applications. We are reusing the technologies that are enabling the type of collaborative and social services associated with the mentioned Web 2.0 phenomenon. What we are creating in the smart space domain is similar in the sense that it involves mash-ups of services in the local space, thus allowing users to derive value from interacting locally with people, devices and services.

Earlier approaches like CoolTown [13] have applied basic web architectures to ubiquitous applications based on PDAs. We now see the opportunity to use traditional as well as Web 2.0 technologies as a platform for providing services to ubiquitous devices [14, 15]. 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. An example is the open source Nokia Raccoon server [18], which is a port of the Apache web server to Nokia S60 smartphones that uses Python as a scripting language. This enables us to leverage new web technologies for content hosting and management. In addition, we can also include other smart devices, such as home entertainment devices and dedicated sensor devices.

Following this approach, we also have to understand the specific requirements and settings of ubiquitous computing research. For instance, ubiquitous research has often focused on novel hardware or software technologies to implement and evaluate a few specific ubiquitous

Christian Prehofer
Nokia Research Center

Jilles Van Gurp
Nokia Research Center

Cristiano di Flora
Nokia Research Center
P.O. Box 407, FI-00045 NOKIA GROUP, Finland
E-mail: {christian.prehofer, jilles.vangurp, cristiano.di-flora}@nokia.com

computing techniques (e.g. location-sensing techniques, context management). Consequently, this has resulted in green field approachs where researchers decided to implement components from scratch, tailored to the requirements of a specific setting. While there are exceptions, a greenfield approach strongly obstructs reuse of software and has not led to any viable platform.

In summary, the underlying problem for mass market commercial success of smart space applications is that they are highly disruptive to existing consumer devices, which are typically high-volume and inexpensive. To view this from a business perspective, we would like to quote from the well known book "The innovators solution" [10]: "Disruptive innovations usually do not entail technological breakthroughs. Rather, they package available technologies in a disruptive business model." This basic insight has been seen in history in many cases, even such prominent, highly disruptive examples like electricity or cars. These took many years or decades to establish themselves. Thus our claim is that many smart space applications present not too little but too many innovations for a disruptive new market.

Following the above, this paper presents our approach to build smart space applications by using web technologies as a platform in ubiquitous devices. In the remainder of this paper we first discuss scenarios and existing technologies and then present our web-based architecture. We then discuss related work and discuss open research issues stemming from our approach.

2 Scenarios, Requirements, and Existing Technologies

The type of scenarios we envision involves users bringing their devices into a space in order to interact with other people's devices and other objects in the space. Such smart spaces can be associated with a community of end-users who share a common goal or interest around a specific location. A space can be any place where people come together: e.g. a home, an office, a restaurant, a bar, a hotel, a concert, a cinema, a bus or a park. Smart spaces can augment local events by providing attendees with end-user applications to enhance interaction between end-users themselves, as well as to improve experience of individual users attending the event. In all these cases, the smart space must provide interoperable and configurable mechanisms to communicate with people, devices and services in local environment. Additionally, services in a smart space may be intermingled with external Internet services.

As an example, consider public events such as for example people watching a jazz performance in a bar. Generally, such events have many people in them doing things like taking pictures with their cameras and camera phones; shooting video and expressing opinions via phone or text messages. This information is however not easily shared with the other people in the local space.

2.1 Main Requirements and Stakeholders

In the following, we give a short overview of some of the key requirements. We start first by requirements from the key stakeholders, and then structure the remainder by technology areas.

End users and usability. For end users, the experience of connecting to and interacting with the smart space needs to be effortless and seamless. In other words, it is crucial that the system is easy to configure and use. Ideally, users should be able to connect to the local space without any configuration and connect to it by simply walking into it. A further key requirement is that the platform must work with the widest possible range of devices.

Smart space customization and service creation. A second group of stakeholders consists of people owning, deploying and customizing services in the space. This includes the owner of the space (e.g. the bar owner) and also some visitors may choose to host services on their own devices (e.g. a photo blog). In neither case these people are likely to be technical experts. Installing, configuring and customizing the services should therefore be straightforward.

Connectivity. A basic requirement for devices in the smart space is to be able to connect to an IP network locally. Such a network may be provided by smart space owners in the form of a wireless access point. Alternatively, devices can connect directly in a peer-to-peer fashion. Non-IP devices can be attached by service specific gateways or proxies. Additionally, there may be remote users wanting to participate in activities taking place in a smart space.

Security. An important aspect of managing the local smart space is dealing with trust and security. Unlike the Internet, it is quite difficult to rely on centralized facilities to orchestrate trust and security strategies in a smart space. Consequently, a decentralized solution is preferable. On the other hand, being connected locally makes it easier to establish trust relations in some cases. For example, in the previously mentioned scenario, the bar keeper could grant default access to an audio system from devices known to be local.

Interoperability. A key requirement for software development in this space is interoperability. Interoperability with other software components is important because the smart space is very likely to be populated by many different devices and software components. Web technologies can be applied in the local smart space since they are now also well supported on mobile devices, but limitations and specific characteristics of mobile and embedded devices must be taken into account.

Engaging the development community. Development should be easy enough to engage a wide community of developers. On the Internet, the wide deployment of scripting languages such as PHP has opened up web

application and service development to millions of professional and amateur developers.

2.2 Existing Technologies

In the following we survey and classify existing technologies according to different device categories.

PC-class devices and high-end mobile devices. High end mobile devices can connect to a WLAN and have the ability to run web application server software needed for hosting services in the smart spaces.

Personal mobile devices with WLAN and a browser. This growing category overlaps with the previous category, but includes phones that may not able to host services, e.g. due to lack of software or memory.

2G/3G Phones with a browser. While many phones still lack the ability to connect to a WLAN, quite many have a browser and can connect to the Internet. It is possible to integrate them into the smart space via normal Internet by a gateway, although e.g. NAT translation and firewalls may pose problems.

Low end phones without a browser. While the before mentioned devices represent the future, low end devices without a browser represent still a considerable group of devices which cannot connect to the smart space. For example, while limited, SMS messaging has been successfully used for large public events and could also be used for smart spaces.

Non personal, networked devices. There is an increasing amount of non personal devices with a network connection. This includes multimedia home appliances such as Universal Plug'n'Play (UPnP) [24] media servers and renderers, Bluetooth accessories. home automation systems, and sensor/actuator networks. Some of these devices can connect directly to an IP network; others connect via alternative network technologies.

Independently of a specific device category, software development for the smart space needs to build on, and integrate with existing mobile software development platforms. These platforms currently include web browsers and mobile Java, which are both widely supported across devices. While suitable for running client applications with a graphical user interface, neither browsers nor mobile Java platforms are suitable for developing server applications. For that purpose other platforms need to be used. Fortunately, many high end devices support both application servers and scripting languages such as for example Python. This enables the porting and use of a wide range of technologies currently used on web application servers. This trend will progress rapidly and many devices will be capable of running and hosting web servers and other service technologies. This makes it possible to build the software infrastructure for smart spaces on this technology. Mobile servers can host familiar web applications such as photo albums, blogs, content management systems, while their clients can run sophisticated AJAX user interfaces to provide users with a nice user experience. In addition they can host a wide variety of networked services with a web service APIs.

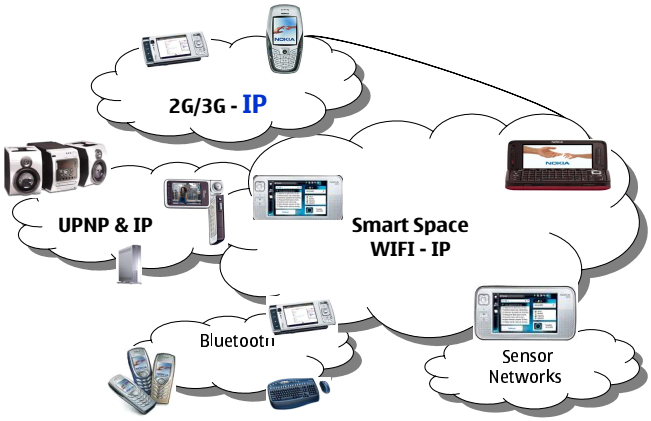

Fig. 1 Local IP network with smart devices acting as proxies

3 Architecture and System Design Approach

In the following, we first introduce our network architecture and then present an overview of our system architecture.

3.1 Smart Space Network

Our 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. While all devices discussed above can be part of the smart space directly or indirectly, only the top two categories of the classification in the last section have the ability to run custom service software. Consequently, we consider these devices to be the backbone of the smart space. In addition, sensor/actuator networks, where power usage is a great concern, are often connected to high end devices not by IP networking but through technologies such as Wibree or Bluetooth. To integrate such devices in the smart space, their services must be 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 [22,23], which is similar to DNS and integrates into the hostname resolution at the OS level. This also supports service registration and discovery. By using mDNS, devices can advertise their name within a locally scoped namespace, e.g., device A is available at de-

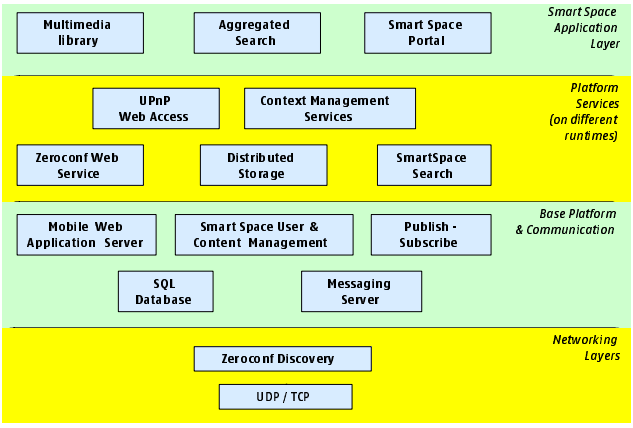

Fig. 2 System Architecture

viceA.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.

3.2 System Architecture and Design Decisions

Figure 2 provides a bird's eye view on the architecture of our web based infrastructure for smart spaces. A detailed discussion of this architecture is beyond the scope of this article. In this section we will focus on the key design decisions underlying this architecture.

The bottom layer contains the mechanisms discussed previously, including a network protocol stack and Zeroconf device discovery layer. 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. For example, we use the Nokia S60 version of Apache httpd [18] and the S60 port of Python as well as the Java OSGI Http Service on top of the J2ME CDC environment. These two components can both be used as the mobile web application server in Figure 2. They are used to run web services, web applications and for hosting multimedia content. The difference with a normal web server is that services and applications running on it (e.g. a python script or OSGI Java components) can access the Zeroconf service discovery mechanism to integrate other services in the smart space. Other run-times, for example a PHP interpreter, can be integrated alternatively. 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 asynchronous communication infrastructures.

Smart space services can be created by using the technologies in the second layer. A few typical smart space specific services are shown in the third layer. Applications that use these services are listed in the top layer. These may be web applications, which can be accessed using a browser and which can be hosted on the mobile web application server. Alternatively, they can be written using any of the device specific development kits that access the services.

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 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 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.

4 Related Work and Research Challenges

There has been abundant research in the area of ubiquitous computing and we cannot list all challenges here. We focus on a few key issues which arise from applying existing software and technologies to our setting.

Content-management. Content management system (CMS) software for e.g. blogging, web-forms, or In-

ternet services like MySpace.com, have made it increasingly more simple for users to create content on the Internet.

There are thousands of CMS products available that are optimized for various use cases (see e.g. the CMS Matrix [20]). These products provide a wide range of features related to content syndication, versioning, scalability, security, web site design, etc. In recent years the phenomenon of users adopting CMS tools and using them to create massive amounts of content and to remix content and services on a large scale, has inspired the term Web 2.0 [21].

In the smart space, Web 2.0 technology may enable local participants and service providers to create content in the smart space. However, there are several challenges related to the ad-hoc and local character of smart spaces. For example, the smart space CMS content would ideally be robust against users and services joining and leaving the smart space.

A clear example of such issues is provided by the work in [11], which shows how a well architected modular design can address such challenges in location-based systems. However, it does not discuss the issues related to other types of smart space systems. The work in [12] by Storz et al. outlines several lessons learnt on content management that have general applicability to many types of public ubiquitous computing deployments for situated displays. The work makes it clear that content management requirements of pervasive environments cannot be satisfied by existing content management systems.

Additionally, in typical content management systems, there are clear roles and responsibilities for the content. In the smart space this has to be managed in a more dynamic way e.g. taking into account that some services may be less trusted as they are user created. Finally, links to other services have to be more dynamic because they can start/stop working as devices come and go. Related to this is the need to ensure availability of important content by, for example, replicating it on multiple devices.

Web technology for smart space services. The idea of using web technologies as a smart space middleware and of considering the web itself as a smart space runtime environment has been already investigated by several state-of-the-art research works. The Cooltown system [13] provides one of the clearest examples of such a research effort. Our work and much earlier work on Cooltown share the same goal to enable people to build a Web presence for real-world entities without requiring advanced programming expertise. As discussed above, we now exploit a much larger part of the tools for Internet services.

Universal Plug'n'Play (UPnP) [24] and the more recent Device Profile for Web Services (DPWS) [16] are two clear examples of off-the-shelf service discovery and delivery infrastructures built on top of existing web technologies. UPnP is designed to support zero-configuration, invisible networking, and automatic discovery for a breadth of device categories. The UPnP technology has very strong roots in standard Web technologies. It is based on IP, HTTP, Web browsing, XML and SOAP. UPnP's service discovery and event notification frameworks are strongly based on HTTP and rely on several IETF standards. DPWS is aligned with Web Services standards, such as WSDL (Web Services Description Language), XML Schemas, SOAP, and other web-services protocols concerning addressing, security, and meta-data exchange. The main limitation of Cooltown as well as of UPnP, DPWS, and other related approaches is that they rely on traditional web technologies without taking into account the latest advances in technologies and user-interaction models related to the web. On the contrary, our work takes explicitly into account migration of web-technologies and related user communities towards before mentioned Web 2.0 technologies. We are aiming at exploiting these technologies as the basic enabling platform for our smart spaces.

Security. A pervasive challenge is of course security. As we have several server devices distributed in the network, users certainly do not want to log in separately to each of them. The challenge is to manage user credentials in such a way that the user can rely on the trust provider. Additional challenges come from privacy requirements, which can vary widely between different smarts space scenarios.

Network issues. Web platform technologies have advanced significantly in reliability by many techniques for redundancy and scalability for the Internet. In our smart space setting, challenges come more from the limitations of devices and dynamic network scenarios. So far, we have evaluated infrastructure based WLAN, but more dynamic ad-hoc connectivity with multiple wireless hops is even more challenging. For instance, users moving in a building may not always be best connected to a WLAN access point and suffer from interruptions and/or high packet loss. This can lead poor user experience, e.g. due to session termination.

As we aim to run services mostly on mobile devices, and some of them joining dynamically, energy efficiency and energy awareness are crucial. For instance, in larger scenarios, services may have to be replicated and load balancing is needed not just for network load but also for power reasons.

Similarly, monitoring the smart space and ensuring a consistent user experience is difficult for several reasons. First, the network capacity is difficult to plan and monitor in WLANs. For instance, the owner of a smart space in a restaurant may want to ensure that all users have acceptable service experience. In this case, it may be better to limit the number of streaming sessions before the quality for all users decreases. Secondly, disruptions such as intentional or unexpected disconnections and variable

network performance can degrade the service quality and are difficult to monitor.

5 Conclusions

We have presented a web based middleware for smart spaces, which strongly relies on technologies used in Internet services. In this way we aim at simplifying the creation of ubiquitous services and making many of the highly disruptive ubiquitous applications less disruptive from a technology point of view. Based on an analysis of requirements and technologies, we have presented an architecture for a web based middleware designed for smart spaces. We have shown that many of the key technologies for "Web 2.0" Internet services can be exploited. This includes not just browsers and basic web servers, but also development tools, scripting languages and content management. While many technologies can be reused from Internet applications, we also face a number of additional challenges in this new setting. These have been discussed with respect to several key issues such as discovery/delivery of services, security, content management, and networking. We are currently implementing the presented architecture and already have good experience running several popular software packages and applications on the Nokia N800 Internet tablet on top of Python and OSGI Java components. Furthermore, many of the key Internet technologies are available as open source projects, which makes them more accessible and adaptable to our platform.

Acknowledgements The authors are grateful to the project team at Nokia Research Center for many discussions and insights leading to this approach. Further, Pasi Liimatainen has given valuable feedback to this paper.

References

1. Wang, X., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., Zhang, D., Semantic Space: an infrastructure for smart spaces, Pervasive Computing, IEEE , 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. Martin, D., Cheyer, A., Moran, D., The Open Agent Architecture: A framework for building distributed software systems, in Applied Artificial Intelligence: An International Journal, 13(1-2), pp. 91-128, 1999.
7. Fox, A., Johanson, B., Hanrahan, P., Winograd, T., Integrating Information Appliances into an Interactive Workspace, in IEEE Computer Graphics and Applications, 20(3), pp. 54-65, 2000.
8. Xie, W., Shi, Y., Xu, G., Mao, Y., Smart Platform - a software infrastructure for Smart Space (SISS), proceedings of the Fourth IEEE International Conference on Multimodal Interfaces, 2002.
9. Uribarren, A., Parra, J., Uribe, J.P., Makibar, K., Olalde, I., Herrasti, N., Service Oriented Pervasive Applications Based On Interoperable Middleware, Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures, Dublin, Ireland, 2006.
10. Christensen, C. M., Raynor, M. E., The Innovator's Solution: Creating and Sustaining Successful Growth, Boston: Harvard Business School Press, 2003.
11. Tummala, H., Jones, J. Developing spatially-aware content management systems for dynamic, location-specific information in mobile environments. 3rd ACM international Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (Cologne, Germany). WMASH '05. ACM Press, New York, NY, 2005.
12. Storz, O., Friday, A., Davies, N., Finney, J. Sas, C., Sheridan, J., Public Ubiquitous Computing Systems: Lessons from the e-Campus Display Deployments, IEEE Pervasive Computing, 5(3), pp. 40-47, 2006.
13. Debaty, P., Caswell, D., Uniform Web presence architecture for people, places, and things, IEEE Personal Communications, 8(4), pp. 46-51, 2001.
14. Schroth, C., Janner, T., Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services, IT Professional, 9(3), pp. 36-41, 2007.
15. 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.
16. Chan, S., Kaler, C., Kuehnel, T., Regnier, A., Roe, B., Sather, D., Schlimmer, J., Sekine, H., Walter, D., Weast, J. and others., Devices Profile for Web Services, May 2005, Microsoft Developers Network Library, http://specs.xmlsoap.org/ws/2005/05/devprof/devicesprofile.pdf, 2005
17. Jammes, F., Mensch, A., Smit, H. Service-oriented device communications using the devices profile for web services, 3rd international Workshop on Middleware For Pervasive and Ad-Hoc Computing, Grenoble, France. MPAC '05, vol. 115. ACM Press, New York, 2005.
18. Mobile Web Server, Raccoon, http://wiki.opensource.nokia.com/projects/Mobile_Web_Server, 2007.
19. Kindberg, T., et al., People, Places, Things: Web Presence for the Real World, Third IEEE Workshop on Mobile Computing Systems and Applications, 2000.
20. CMS Matrix, http://cmsmatrix.org. 2007.
21. O'Reilly, T., Web 2.0: Compact Definition, http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html, 2005.
22. Guttman, E., Microsyst, S., Autoconfiguration for IP Networking: Enabling Local Communication, IEEE Internet Computing 5 (3), pp. 81-86, 2001.
23. 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.
24. UpnP Forum, UpnP Device Architecture 1.0, July 2006, http://www.upnp.org/resources/documents.asp, 2006.