Together we are strong– Towards Ad-Hoc Smart Spaces

Andreas Brodt*, Sailesh Sathish‡
* IPVS Universität Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany, andreas.brodt@ipvs.uni-stuttgart.de
‡ Nokia Research Center, Visiokatu 1, 33720 Tampere, Finland, sailesh.sathish@nokia.com

***Abstract*—Today's mobile devices feature many resources, sensors and data, which make it possible for them to adapt to their environment. However, they would have a lot more resources at hand, and thus could better adapt to their context, if they used their communication capabilities to share their resources with others near them. In this paper we propose the concept of ad-hoc smart spaces, which enables mobile devices to utilize resources owned by other devices in physical proximity. This imposes a number of challenges as cooperation partners can join and leave any time and as devices are highly heterogeneous. We propose a system architecture for ad-hoc smart spaces and present a prototypic implementation.**

I. Introduction

Today's mobile devices (smartphones, PDAs, etc.) have come a long way from single purpose tools to mobile general purpose computers. They feature many capabilities, including GPS receivers, cameras, or accelerometers. Moreover, mobile devices carry lots of data, such as calendar data, or maps. This allows applications to adapt to the user's *context*, which is defined as any information that can be used to characterize the situation of an entity [5]. However, not all aspects of context are always available. Devices largely differ in their capabilities and not every sensor is always functional (e. g., GPS indoors). What would humans do in such a situation? Most people would probably ask somebody near them for help. Although today's mobile devices possess numerous means to communicate, they don't do that.

If mobile devices could combine their own context models with (parts of) context models from other devices nearby, a much richer view of the world would result and context-aware applications would become a lot more useful. To illustrate this vision, let us consider Anna, who is using a pedestrian navigation application on her smart phone. Anna does not posses a GPS receiver, so her phone estimates her position from the cellular network and only shows a map of the area. A few meters behind her, Bert is also navigating, but with a GPS-enabled phone. As soon as the two devices discover each other, Anna's device receives much more accurate position data from Bert's and gives Anna exact walking directions. On the other hand, Bert can now see local sights on his map, which Anna had stored on her device.

Later, Anna is in a meeting with Charles, Denise and Edward. To find an appropriate date for a follow up-meeting, Anna asks her calendar application to search for a two-hour slot in about three weeks. All devices have been together in the room long enough to discover each other and everybody allows Anna to read their free time slots. So Anna's calendar can easily find a date which is fine for everybody.

Before she leaves the meeting, Anna uses a printer from her mobile phone connected to Edward's laptop.

In this paper we propose the concept of *ad-hoc smart spaces* to address Anna's scenario. We identify major challenges that ad-hoc smart spaces impose, present a system architecture and a prototypic implementation. The remainder of this paper is organized as follows: In Section II we define ad-hoc smart spaces. Section III lists the most important challenges imposed by ad-hoc smart spaces and Section IV proposes a system architecture to address them. We present our prototypic implementation in Section V and discuss related work in Section VI. We conclude the paper in Section VII

II. Ad-Hoc Smart Spaces

To put Anna's scenario of spontaneously cooperating co-located devices into practice, we propose the concept of *ad-hoc smart spaces*. They do not require any dedicated infrastructure, but are self-organizing *device societies* which solely rely on the mobile devices themselves. A device society is formed when devices are brought into proximity. The devices discover each other, exchange their resource information, and share their resources between applications running on the respective devices. As devices move, device societies grow and shrink, split up and merge, so capabilities appear and disappear very dynamically. Consequently, applications for ad-hoc smart spaces must constantly adapt to the capabilities that are currently available.

Ad-hoc smart spaces do not attempt to replace existing context-aware applications or infrastructure-based smart space environments. On the contrary, they improve existing context-aware applications and can be used in addition to infrastructure-based approaches, as they attempt to provide context information where other approaches have already given up (e. g., when navigating without GPS).

III. Challenges

Ad-hoc smart spaces impose a number of challenges. None of them is new within the context of smart spaces, but the ad-hoc characteristics add quite a bit of new flavor.

Ad-hoc device discovery. First of all, devices have to find each other before any interaction is possible. Wireless device discovery itself is well understood, the question is merely when to do it. Ad-hoc smart spaces address a scenario with

constantly changing device neighborhoods and thus cannot make any assumptions on available interaction partners. Devices constantly attempting to find others that are not there will waste unnecessary battery power. Devices not searching sufficiently often may not find useful cooperation partners. Discovering devices and configuring wireless communication between them is a major challenge.

Fault tolerance. In ad-hoc smart spaces it is the rule, rather than the exception, that any member of a device society may leave at any time. Thus, fault tolerance must be designed into any implementation from the very beginning. This in turn influences the way devices interact. No guarantees can be given that a certain message will ever reach its destination or whether the recipient will be seen ever again, i. e., reliable communication or distributed transactions are utterly difficult in ad-hoc smart spaces.

Directory service. In addition to finding cooperating devices, it is also a challenge to discover resources of interest offered by the device society, since no central entity keeps track of what devices offer. A distributed index, e. g., as provided by Chord [12], must be highly redundant to be sufficiently fault tolerant. However, considering the highly volatile nature of ad-hoc smart spaces, the device society must not be drowned in index management and recovery. Additionally, it is desirable that applications be notified as new resources appear.

Interaction paradigms. As seen from Anna's scenario in Section I, devices may interact in several ways. Anna using Bert's GPS is a typical example of a data stream accessed via *publish/subscribe*. Map data or calendars are static data which are used in a *query/response* scheme. Finally, services, such a printer, require interface descriptions and invocation messages. While it seems easy to offer one specific interaction paradigm, we consider it a challenge to offer all of them in a single system.

Context modeling. Context-aware applications need a data model to represent context. In an ad-hoc smart space, applications have no direct control over context data sources. Position information originating from a remote device, for instance, does not reflect the exact position of the user and, depending on the requirement of a particular application, might or might not be accurate enough. Only the application can decide whether remote context data is useful. Thus, it is essential that the context model provides detailed *metadata* and a notion of *data quality*.

Moreover, ad-hoc smart spaces bring together utterly different devices. Ideally, these devices communicate seamlessly using the same context model. But devices come up with new capabilities and different vendors are likely to use different models. To be realistic, ad-hoc smart spaces will have to cope with *heterogeneous context models*.

Access control. Ad-hoc smart spaces provide device resources for everybody in physical proximity, so it is self-evident access control is needed. Within such spaces private data, such as the user's calendar, must be protected. Yet, people who are physically co-located have overlapping contexts, so a part of their context model is deemed shareable. Moreover, ad-hoc smart spaces will only work if there is *some* (non-private) data available. So the question is merely to find a simple but powerful way for the user to determine what can be shared with whom.

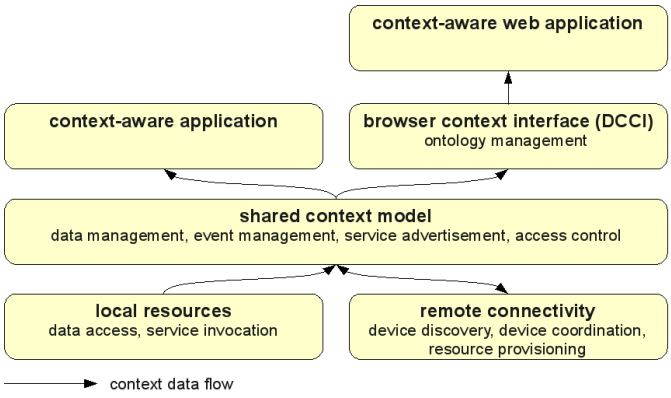

Fig. 1. Proposed system architecture for ad-hoc smart spaces

Platform independent applications. It is entirely possible to build native applications for every device participating in ad-hoc smart spaces. For certain application domains this might be the best choice. However, since applications for ad-hoc smart spaces must constantly adapt to the available resources, they are not always simple to build. Thus, it would be desirable if they could be reused for the entire heterogeneity of devices participating in ad-hoc smart spaces. For this reason, the resources of an ad-hoc smart space should be made available to *web applications*.

IV. Architecture

To put the vision of ad-hoc smart spaces into practice and address the above challenges, we propose a middleware layer for mobile devices we call the *shared context model*. The shared context model contains local resources of the mobile device and shares them with other devices nearby. Mobile applications built on top of the shared context model access a unified view of both local and remote resources and are disburdened from device discovery, data exchange, access control, and error management. Applications do not have to know as to where the resource is being accessed from i.e. whether the resource is remote or local to the device. However, applications can get this information by checking the metadata for that particular resource.

Figure 1 illustrates our proposed system architecture. Every device hosts its own shared context model, which runs as a daemon process in the background. The shared context model is the central context model for everything a device knows about its current situation and contains data and metadata as well as service descriptions for all known resources. Being the component where everything is integrated, the shared context model is also the right place to perform access control.

The shared context model features a *provisioning interface*, via which resources are entered. Local resources are registered at the shared context model, but given the heterogeneity of

resources, they cannot be accessed in a unified way. Static data, such as device capabilities, is read directly from storage. Dynamic data, data streams, and services, e. g., battery state, position information, or the camera, respectively, must be accessed via appropriate wrappers.

In addition to local resources, one or more components providing remote connectivity are connected to the provisioning interface of the shared context model. The remote connectivity components provide what makes ad-hoc smart spaces unique. Based on the networking capabilities of the device's operating system, they search for nearby devices, exchange resource descriptions and update the shared context model accordingly. The remote connectivity components coordinate the device society, i. e., this is where self-organization of ad-hoc smart spaces happens. Upon request from the shared context model, the remote connectivity components fetch required resources from remote devices. Finally, they remove resources from the shared context model which are no longer available.

For applications utilizing ad-hoc smart spaces, the shared context model provides the *client interface*. The client interface provides a browsable view of the resources currently available in the shared context model. In addition, the client interface features an event system to notify applications of changes in the shared context model, so that the applications can adapt accordingly. Naturally, the client interface lets applications access the resources of the shared context model, i. e., the client interface provides all supported interaction paradigms of ad-hoc smart spaces.

To support context-aware web applications utilizing ad-hoc smart spaces, the shared context model must be made available in the web browser. This is done by extending the web browser of the mobile device providing a context model suited for web applications. In order to achieve wide adoption of this model, it is important to use standardized interfaces. For this purpose we propose the W3C Delivery Context Client Interfaces (DCCI) [13]. The DCCI specification defines a tree containing *properties* to expose context data to web applications. DCCI specifies only the interface and makes no statement on the underlying implementation or where the context data is obtained from. Moreover, DCCI supports the dynamic addition and removal of properties and features events for web applications to adapt accordingly. Thus, DCCI fits the scenario of ad-hoc smart spaces very well. As DCCI mandates an object-based context model, an implementation of DCCI for ad-hoc smart spaces likely has to convert between the context model used in the shared context model and the one presented to the web applications. Moreover, opening the context model to web pages even increases the need for access control. This is taken care of by the shared context model, but has to work hand in hand with the DCCI implementation.

V. Prototype Implementation

We built a prototypic implementation to study the advantages of ad-hoc smart spaces and obtain first insights on how the challenges of ad-hoc smart spaces can be addressed best. The mobile devices we used are the Nokia N810 and N800 Internet Tablets running maemo linux, as they provide a nearly full desktop linux environment which facilitates development considerably. In addition, we used a laptop running ubuntu linux. The Nokia N810 features a built-in GPS receiver whereas the N800 does not, so we focused on the GPS sharing scenario. To support this scenario, we implemented a shared context model in the Python programming language which addresses most of the challenges elaborated in Section III in a simple way: Remote connectivity is based on Bluetooth networking and we use Bluetooth service discovery to find co-located devices running a shared context model service. Fault tolerance is implemented by removing all knowledge about a remote device as soon as communication errors occur. To provide directory service functionality, knowledge of other devices' resources is currently replicated on all devices of a device society. For the GPS sharing scenario, we implemented the publish/subscribe interaction paradigm on a context model based on *properties*. A property is identified by a namespace and a name, to cater for heterogeneous context model schemas, and carries a value, on which the shared context model does not impose any restrictions. As our prototype is merely for studying, it does not address access control. For platform-independent applications, we extended the mobile web browser of the Internet Tablets. We used the Telar DCCI implementation [1] developed for Nokia Internet Tablets in our previous work [2] and added an additional context data provider as a bridge between the shared context model and the DCCI property tree.

The shared context model is implemented as a dbus service and as such automatically started when an application first accesses it, e. g., when the DCCI browser extension is brought up. Local resources become available immediately. Also, the remote connectivity component of the shared context model starts to perform a Bluetooth scan every 30 seconds to find other devices running a shared context model service. At the same time a Bluetooth server socket is kept open for other devices to connect. As soon as one device discovers another one, they exchange the (namespace, name)-pairs of all their properties. This includes properties of other devices they know about, i. e., device A might receive from device B knowledge about a property owned by device C. If after this "handshake" a shared context model has gained knowledge about new properties, a dbus signal is sent to inform applications (and the DCCI bridge) about the new properties. Applications interested in the new properties may subsequently subscribe to the new properties, which causes the shared context model to subscribe at the remote device. From that point on, the remote device will send notifications whenever the value of a subscribed property changes. If the subscribed property is not owned by the remote device, subscriptions are made along the entire path to the original property owner and notifications are routed back along this path. A subscription ends if the requesting application explicitly unsubscribes or if communication errors occur, in which case the shared context model removes all knowledge about the respective property. The removal of properties is broadcasted to applications as a dbus signal.

Using our prototype we were able to build and use location-aware web applications on a Nokia N800 which utilize position data gained from the GPS device of a co-located Nokia N810 or simulated by a laptop. This demonstrates the concept of ad-hoc smart spaces very nicely.

VI. Related Work

Since Mark Weiser's vision of ubiquitous computing [14] numerous frameworks, platforms, and systems were created to put this vision into practice. Smart environment systems [3], [8] create intelligent living or meeting rooms from cooperating devices with some dynamic aspects. These systems are strongly infrastructure-based and usually rely on a central coordinator, such as the EventHeap in [8]. Moreover, the smart environments in the various research labs are often built from devices whose functionality is known to the developers beforehand.

Sodapop [7] builds smart environments without a central coordinating infrastructure. To achieve a certain goal, a suitable chain of stateless *channels* and stateful *transducers* has to be created in a device society. The Sodapop model is event-based and is not designed for the highly volatile scenario of ad-hoc smart spaces.

The Nexus platform [10] is an example of a distributed context platform and provides a unified world-scale context model composed by numerous context servers. Nexus features a complex and extensible schema and metadata model. In Nexus, a client has to include some knowledge about its context (e. g., its location) in a query to get appropriate results. Using ad-hoc smart spaces, the context servers would be in proximity, so the client's context would be implicitly clear.

AmbientDB [6] is a self-organizing P2P database management system intended for an ad-hoc mobile environment. It makes use of Chord [12] to share data among a potentially large collection of nodes in a robust way. AmbientDB is used in a query/response manner but again aims at a less dynamic scenario than ad-hoc smart spaces do.

The Context Broker [4] provides a centralized context model using the Web Ontology Language (OWL) for for modeling ontologies of context and supporting context reasoning. This is interesting for ad-hoc smart spaces too, as it provides a possibility to deal with heterogeneity and fault tolerance.

Mäntyjärvi et. al. presented a method to make context recognition more reliable by combining uncertain context data from multiple devices [9]. This addresses the challenges of context modeling, metadata, and data quality.

Contory [11] is a Java-based context provisioning middleware for mobile devices which incorporates multiple provisioning strategies, including ad-hoc networks.

VII. Conclusion

Today's mobile devices could be a lot more useful if they utilized their manifold means of communication to work together and share their resources. In this paper we proposed the concept of ad-hoc smart spaces, which enables mobile devices to benefit from resources owned by other devices in physical proximity. We identified major challenges mainly caused by the highly dynamic scenario of ad-hoc smart spaces and the heterogeneity of devices. We discussed an architecture to address these challenges and presented a first prototypic implementation.

Our work is at an early stage and we are yet to fully address challenges such as access control, varied interaction paradigms and a more powerful context model supporting heterogeneous data models. We will be integrating more devices with support for varied communication platforms and service discovery. We aim to conduct subjective and objective evaluation of the system through user trials incorporating more applications, different browsers, and devices from different manufacturers. We are encouraged by the potential of ad-hoc smart spaces and intend to provide a more comprehensive report soon.

Acknowledgments

We thank Christian Prehofer, Leonid Zolotarev, Nazario Cipriani, Michael Trunner and our fellow colleagues at Nokia Research Center, W3C, and University of Stuttgart for all their valuable feedback and support with the overall framework.

References

[1] A. Brodt. Telar DCCI website, 2008. http://telardcci.garage.maemo.org.

[2] A. Brodt, D. Nicklas, S. Sathish, and B. Mitschang. Context-aware mashups for mobile devices. In *Web Information Systems Engineering - WISE 2008, 9th International Conference*, volume 5175 of *LNCS*, 2008.

[3] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easyliving: Technologies for intelligent environments. *Handheld and Ubiquitious Computing*, 2000.

[4] H. Chen, T. Finin, and A. Joshi. Semantic web in the context broker architecture. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, Washington, DC, USA, 2004. IEEE Computer Society.

[5] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1), 2001.

[6] W. Fontijn and P. Boncz. Ambientdb: P2P data management middleware for ambient intelligence. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. IEEE Computer Society, 2004.

[7] M. Hellenschmidt and T. Kirste. Sodapop: a software infrastructure supporting self-organization in intelligent environments. In *2nd IEEE International Conference on Industrial Informatics (INDIN 04)*, 2004.

[8] B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: experiences with ubiquitous computing rooms. *Pervasive Computing, IEEE*, 1(2), 2002.

[9] J. Mäntyjärvi, J. Himberg, and P. Huuskonen. Collaborative context recognition for handheld devices. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, Washington, DC, USA, 2003. IEEE Computer Society.

[10] D. Nicklas, M. Großmann, T. Schwarz, S. Volz, and B. Mitschang. A model-based, open architecture for mobile, spatially aware applications. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases: SSTD*, 2001.

[11] O. Riva. Contory: A Middleware for the Provisioning of Context Information on Smart Phones. In *Proceedings of the 7th ACM International Middleware Conference (Middleware'06)*, volume 4290 of *LNCS*. Springer, 2006.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, 2001.

[13] K. Waters, R. A. Hosn, D. Raggett, S. Sathish, M. Womer, M. Froumentin, and R. Lewis. Delivery context: Client interfaces (DCCI) 1.0. Candidate recommendation, W3C, 2007.

[14] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3), 1991.