Asomnia: *A S*ervice-*O*riented *M*iddleware for Ambie*n*t *I*nformation *A*ccess

Karl Rehrl, Wernher Behrendt, Manfred Bortenschlager, Sigi Reich, Harald Rieser, and Rupert Westenthaler

Salzburg Research
Jakob Haringer Straße 5/III
5020 Salzburg, Austria
E-Mail: {krehrl, wbehrendt, mborten, sreich, hrieser, rwestenthaler}@salzburgresearch.at

Abstract. With the growing pervasiveness of information systems we are increasingly confronted with integrating heterogeneous end-user devices into existing information infrastructures. However, most existing middleware platforms either focus on plug & work functionalities or on multimedia streaming capabilities. Asomnia is a service-oriented middleware that combines the needs of plug & work infrastructures with the necessities of delivering multimedia contents. We describe Asomnia's service-oriented architecture and show how it can be applied in different scenarios.

1 Introduction

With the advent of pervasive information delivery and consumption, we increasingly face the issue of having to integrate various new types of end-user devices into existing information infrastructures. New middleware platforms are necessary in order to prepare the transition from PC based client/server computing to pervasive information systems, thus leading to ubiquitous information access and the convergence of multimedia content delivery, peer-to-peer networks and heterogeneous, context-aware devices.

The application scenario we are addressing is concerned with information systems for travellers in the area of public transport. As passengers we are all accustomed to textual information informing us about departure times, delays, etc. Furthermore, we are often confronted with the fact that the information provided is not up-to-date.

In this paper we argue for a convergent service-oriented middleware platform that allows for multimedia information to be displayed with arbitrary devices; by focusing on plug & work integration of devices in ad-hoc networks we are able to provide the necessary communicative hooks for enabling display devices to be closely connected to the information systems in the back-end in order to provide up-to-date information.

This paper is structured as follows. Section 2 outlines the requirements that lead to the design of Asomnia. Next, Section 3 provides a description of related work. Following on to that, Section 4 describes ASOMNIA's system architecture. We continue with sample scenarios in Section 5 and conclude in Section 6.

2 Requirements for service-oriented middleware infrastructure

In this section we list the requirements from an infrastructure point of view. The latter sections will refer to these requirements.

R1 Openness with respect to devices and networks A key requirement for pervasive infrastructures is the openness for heterogeneous devices. In the area of public transport we are confronted with different multimedia displays such as video walls, information kiosks or also end-user devices like PDAs or smart phones. This means, that more and more future devices will be connected over standard wireless networks like WLAN or Bluetooth. To assist a broad variety of devices, we argue for building the middleware architecture upon widely adopted device and network standards.

Moreover, in order to allow for situation aware information delivery [17, 8, 3, 4], we are increasingly forced to integrate devices with low processing power, like sensors, micro-controllers or RFID-Tags, which require a conceptual bridge in order to seamlessly communicate between devices.

R2 Simple construction and reconfiguration of services One of the most important aspects of a service-oriented architecture is the modular design, which enables a developer to easily integrate new functionality into existing applications. This means that

1. The developer of new services is encouraged to concentrate on the development of functionality rather than struggling with protocol details.
2. Distribution and installation of new services can be accomplished on-the-fly.
3. Services can be reconfigured (i.e., activated/deactivated/moved) within a running middleware infrastructure.

R3 Location independent service provision Any device in need of a certain functionality should be able to find an appropriate service irrespective of its location. In [5] this feature is referred to as "virtualization of resources", in *Centaurus* [9] a XML based ontology is used for exchanging service capabilities. The concept of "virtualization of resources" is particularly useful for travellers or public transport vehicles, using the same services in local networks at different stations.

R4 Flexible discovery mechanisms for ad-hoc networks Especially for ad-hoc networks in pervasive settings, the reliable discovery of newly or sporadically available services and devices is a crucial requirement for middleware platforms [2, 1]. Public transport vehicles arriving at a station should be able to find the necessary services to acquire new data or to announce their arrival. Travellers with mobile devices can use services available at a station or in a vehicle. Mobile peers, i.e., devices temporarily connected with each other, should enable arbitrary pairs of services to communicate [5].

R5 Plug & work functionality In order to support ad-hoc networks, it is a key requirement for a middleware architecture to make the connection process for new peers as simple as possible. This implies that new peers should be able to get their configuration and service implementations or updates from decentralised plug & work servers.

R6 Support for different communication modes Due to the mostly disconnected mode of mobile devices, communication heavily depends on the current connection status. Existing middleware systems typically depend on one specific communication mode. Object oriented middleware systems such as CORBA, DCOM or RMI are mainly based on synchronous remote procedure calls and provide transaction processing functionality; event distribution systems mainly depend on message passing; other communication modes include the usage of virtual shared memory, e.g. [12, 14]. In pervasive applications, in case a device is connected to a network, synchronous communication is possible, whereas a disconnected device can only make use of asynchronous communication.

R7 Grouping of services In service-oriented architectures it is often difficult to structure, combine or administer services running at different locations. Thus, it is a key requirement to provide mechanisms for a virtual grouping of services, which allows for allocation of access rights, multicast communication and easier administration and monitoring.

3 Related Work

In this section we categorise existing work by providing typical examples for the various middleware platforms. Furthermore, we point out additional examples of middleware architectures related to our approach.

Concerning the category of service-oriented middleware we consider the following examples as important to our work: *Jini* [18], *Cooltown* [10] and the *Open Grid Services Architecture* (OGSA) [5] by the Global Grid Forum. *Jini* uses the abstraction of a federated group of resources, which can be hardware devices or software programs. Using *Jini*, services can be dynamically added and deleted during runtime in a flexible way to reflect the dynamic nature of distributed systems. *Cooltown* influenced our work because of the possibility to virtually represent physical entities, whereas *OGSA* is driven by the idea of providing a higher-level concept of services for grid computing infrastructures. Especially important to our approach are concepts for standard interface definition, local/remote transparency and uniform service semantics.

Besides service-oriented middleware architectures other architectural styles include peer-to-peer(P2P) middleware, distributed event systems and virtual shared memory (VSM) middleware. *JXTA* [6], for instance, is a P2P platform targeted at the development of P2P networking and it is composed of a set of open protocols. The *Hermes* event-based middleware architecture [15] uses a type- and attribute-based publish/subscribe model to build large-scale distributed systems. The *EQUIP* platform [7] provides a middleware infrastructure for exploring the relationship between physical and digital artefacts. The main characteristics are cross language integration, modularisation, extensibility, dynamic loading of code, state sharing and support for heterogeneity of devices and networks.

One important issue considering pervasive applications is the convergence of multimedia content delivery, P2P networks and heterogeneous, context-aware devices. Most of the middleware systems discussed above only focus on one aspect, not considering the necessary convergence of technologies. *JXTA* primarily addresses P2P functionality and the building of ad-hoc networks, whereas *Cooltown* or *EQUIP* provide context information but do not care about the integration of heterogeneous devices or plug & work functionality. When it comes to the assistance of different communication modes, *Jini* is not suitable to off-line devices because of its synchronous communication mode. *Hermes* provides asynchronous communication modes and event distribution but it lacks plug & work functionalities and the support for heterogeneous devices.

In general, most systems provide only low level data or event distribution mechanisms, but they do not provide higher level services including dynamic configurability, plug & work functionality or the infrastructure for adaptive multimedia based content delivery. In the following sections we will describe how we addressed these issues in designing and implementing the service-oriented platform Asomnia.

4 System Architecture

Based on an application scenario in the area of public transport and the requirements outlined in previous sections, we have defined an appropriate network structure for an Asomnia network (Fig. 1) and also an architecture for middleware services running on Asomnia devices.

The network structure shown in Figure 1 reflects our approach to a 2-layered, hierarchical overlay network scheme. The network is logically divided into one global domain (first layer) and a number of local domains (second layer). There is one central device, called the control center, which is logically the root of the system, hosting some root services like the central registry or a global messaging service. This central device is the only well-know device in the Asomnia network, all others are discovered dynamically. Devices can either register at the global registry, or discover a local registry within their local network by IP-multicast. Therefore, devices are able to use only local services available in their actual local domain.

In order to be integrated in the network, each device is running a set of middleware services responsible for certain tasks. Services on devices are using a service infrastructure, which is defined in the device architecture. The device architecture is mainly focused on the following goal: to provide a set of higher level system services and an abstract concept for re-usable service components in order to enable application developers to easily build their application services

Fig. 1. The network structure of a typical application scenario

on top of the infrastructure. Therefore, important issues are the abstraction of the underlying devices, the provision of different communication modes, the easy building and integration of new services and the support of plug & work functionality. Consequently, Asomnia's device architecture consists of the following key components

- A small runtime environment, which is a combination of a core module called `Core` and a communication subsystem called `CoSu`. With the provision of a runtime environment, less effort has to be invested into porting Asomnia's service infrastructure to new devices. By implementing the communication functionalities in a dedicated module a changing of the underlying communication protocols can be achieved by modifying simply the `CoSu` (cf Requirement R1).
- The `ServiceManager` component is responsible for registering and controlling the services of an Asomnia device. At startup services are registered with the `ServiceManager` which takes care of them until they are unregistered. The `ServiceManager` provides access methods to other local/remote services. With its knowledge of local and remote services, the `ServiceManager` is able to use *either* local *or* remote calls for executing service methods (thus improving performance and off-line work capabilities).
- Services in Asomnia are rather coarse-grained functional entities, which are based on an abstract service definition (cf R2). Internally, services are built of fine-grained functional units to allow for component reuse and modular design. However, the functional units are encapsulated behind the service facade and can only be access via the public interface.
- The functionality of a service is defined by supporting a list of events, which was chosen as an abstraction to allow for different communication modes. Events can be activated from functional units or from other services by RPC or messaging, depending on the underlying `CoSu`. Only the event list

of a service is used as an interface by other services for interacting with the service (cf R3), therefore allowing for easy changing of implementations.

4.1 Communication Modes

As a key requirement we have defined the availability of different communication modes, ranging from synchronous communication to asynchronous messaging mechanisms. The actual communication mode used should be determined by the applications' needs (e.g. reliability of communication) and the connection state of the device. In fact, we have defined the following communication modes (cf R6):

Connection state/reliability	*Communication Modes*
Online/reliable	Synchronous RPC/Asynchronous reliable messaging
Online/unreliable	Asynchronous unreliable messaging
Off-line/reliable	Asynchronous reliable messaging
Off-line/unreliable	No communication necessary

Table 1. Communication modes

Asomnia by now supports these different communication modes on top of a Web Services `CoSu`. We believe that Web Services will play a major role in service-oriented architectures within the next few years [5, 6, 13]. Furthermore, the key features of Web Services e.g., openness to different platforms and heterogeneous devices and widely adopted standards, meet our requirements (cf R1) defined in Section 2. To provide the different communication modes, we made the following technical decisions:

– Synchronous, reliable communication between online devices is accomplished by the use of SOAP-RPC, for asynchronous reliable messages between online devices SOAP Messaging is used.
– If the receiver of an event is off-line, SOAP messages are stored at the central messaging service, by using a local messaging service to forward the message to the central messaging service. The messaging service is responsible for reliable delivery and handling acknowledgements as well as timeouts.

4.2 Plug & work

On the path to ubiquitous computing, it is a key requirement to provide access to ad-hoc networks in a convenient way. For example, people who permanently change location and networks [11], are not likely to configure their devices and applications constantly. Thus, we think that pervasive middleware platforms have to provide the appropriate mechanisms for a convenient and ubiquitous access to information systems, independent of underlying network technologies or specific device configurations. We refer to this mechanism as plug & work and define the following key concepts:

- Services are provided in a subdomain or specific location (cf R3). Thus, wireless devices can always use local services in their current subdomain.
- Registry and discovery of services is based on a hierarchical concept, preferring local registries and services to more general ones (cf R4).
- Connecting new devices to the Asomnia network requires only little configuration settings on the device (e.g. a unique ID, the type of the device and optionally the central registry). All other configuration settings are loaded from the nearest plug & work service. Local plug & work services use the central plug & work service to get configuration settings from a centralised database server.
- Services do not have to be installed manually but can be loaded, updated and configured from plug & work access points if necessary (cf R5).
- Specific configuration settings of devices and services can be changed remotely by the system operator via the control center.

4.3 Proxy Services

In some cases it is necessary to integrate a device which is not able to provide the processing power needed to run the Asomnia middleware (or simply cannot offer the necessary software prerequisites to execute the middleware platform, e.g. a Java platform). Micro-controllers or sensors are examples for such devices. In order to integrate these devices a proxy service can be used (cf R1). The proxy service is able to communicate with the low-processing device via an appropriate interface and a specific — often proprietary — protocol.

4.4 Member of Groups and Service for Groups

Traveller information networks in the area of public transport are typically structured hierarchically, defining groups for different kinds of transport means and describing the dependencies between groups. Hierarchical structuring supports the system operator in controlling, monitoring and administrating, e.g. specifying access rights for services, or defining the different groups of services and devices. Therefore, a concept for structuring services is required. Asomnia is based on two key concepts for hierarchically structuring services (cf R7):

- A service can be a member of one or more groups by adding the service in the "Member of Group" set property of the service.
- A service can offer its functionality to other groups by adding the services to the "Service for Groups" set property.

5 Prototype and Sample Applications

As a proof-of-concept and demonstration of the practical applicability in real world scenarios, we have built a prototype based on the Java 2 platform and the the Apache AXIS toolkit.

5.1 Controlling the "Schmunzel" SMIL Player with a Conrad C2-Unit

In our test scenario, we have built a simple demonstration network to simulate a realistic application scenario. PC devices with "Schmunzel" SMIL players [16] were used to emulate multimedia displays. On each PC we installed a service component which was designed as a control interface for the SMIL player. A simple control GUI on another PC was used as control center. Additionally, a Conrad C2-Unit (a micro-controller based on the Infineon C164CI) connected via the RS232 interface was integrated in the middleware infrastructure via a proxy service thus demonstrating the possibility to integrate devices with low processing power. The C2-Unit provides a small display for showing a few characters and a numbed for entering data. This simple interface can be used for controlling the SMIL player.

Fig. 2. The SMIL player (middle) being controlled either by a micro-controller (left) or by the control GUI (right)

Another service component on a PC simulated a simple multimedia repository with SMIL presentations. With the use of the Conrad controller we could choose a SMIL presentation in the repository and tell the SMIL players to play this presentation. The setting was perfectly suited as a showcase for the traveller information scenario. Moreover, with SMIL as an open standard for the presentation of multimedia assets and Java as an open platform for mobile computing, the key requirement for open standards has been fulfilled (cf R1).

5.2 Experiences and Discussion

The prototype implementation of Asomnia's service-oriented architecture was perfectly suited to fulfill the requirements defined in section 2. The following findings can be reported:

– The construction of the necessary services was carried out with little effort because of the concept of abstract services and the simple event-based interaction model. Because of the abstractions defined in the service runtime environment, the service developer is not longer bothered with communication details like building SOAP messages or making remote procedure calls.

- Although Web Services are not yet fully standardised, many concepts were adopted to the communication layer in Asomnia. In fact, Web Services turned out to be well suited for this task because of their assistance of different communication modes. However, Web Service technology lacks an appropriate runtime infrastructure and the concept of higher level services.
- Especially the concept of proxy services has turned out to be a powerful tool for integrating devices with low processing power like sensors. Devices with low processing power are considered a necessity on the way to ambient intelligence.
- Plug & work functionality is an important feature in pervasive computing scenarios. The high amount of heterogeneous devices connected to the network can only be managed by reducing configuration effort on the devices. On Asomnia devices at least only the type of the device or a unique ID has to be configured, which is considered as true plug & work functionality.

6 Summary and Conclusion

In this paper we have argued for the need of convergent middleware infrastructures for pervasive information systems. We have shown that existing systems do not provide the necessary assistance for different communication modes, ad-hoc networks and heterogeneous devices. In fact, many of the systems only provide low level communication, without providing the necessary abstractions.

Thus, in our paper we have defined the key requirements for a middleware infrastructure, which provides convergence between multimedia content delivery, ad-hoc networks and heterogeneous, context-aware devices. We have shown the system architecture and we have demonstrated a proof-of-concept closely related to our real application scenario.

In conclusion, we believe that Asomnia can bridge the gap between pure network connectivity (as provided by Bluetooth, etc.) and existing information systems (ranging from enterprise application solutions to peer-to-peer like applications). Therefore, middleware infrastructures such as Asomnia will enable the full potential of ubiquitous information access.

Acknowledgements

This work has been supported in part by the ASCOM Center of Competence, Salzburg, and the Austrian Forschungsförderungsfonds.

References

1. Dipanjan Chakraborty, Filip Perich, Anupam Joshi, Timothy Finin, and Yelena Yesha. Middleware for mobile information access. In *DEXA Workshops*, pages 729–733, 2002.

2. Harry Chen, Anupam Joshi, and Timothy W. Finin. Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether. *Cluster Computing*, 4(4):343–354, 2001.
3. A. Ferscha, S. Vogl, and W. Beer. Ubiquitous context sensing in wireless environments. In *4th DAPSYS (Austrian-Hungarian Workshop on Distributed and Parallel Systems)*. Kluwer Academic Publishers, 2002.
4. Sebastian Fischmeister, Guido Menkhaus, and Wolfgang Pree. Context-awareness and adaptivity through mobile shadows. Technical report, Software Research Lab, University of Salzburg, 2002.
5. Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. Draft 5, Mathematics and Computer Science Division, Argonne National Laboratory and Department of Computer Science, University of Chicago and Information Sciences Institute, University of Southern California and IBM Corporation, November 2002.
6. Li Gong. Jxta: A network programming environment. *IEEE Internet Computing*, V 5:88–95, 2001.
7. Chris Grennhalgh. Equip: a software platform for distributed interactive systems. Technical report, The Mixed Reality Laboratory, University of Nottingham, 2001.
8. M. Beigl und A. Schmidt H-W. Gellersen. Sensor-based context-awareness for situated computing. In *Workshop on Software Engineering for Wearable and Pervasive Computing SEWPC00 at the 22nd Int. Conference on Software Engineering ICSE 2000, Limerick, Ireland*.
9. L. Kagal, V. Korolev, H. Chen, A. Joshi, and T. Finin. Centaurus: A framework for intelligent services in a mobile environment. In *Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, 2001.
10. Tim Kindberg, John J. Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic. People, places, things: Web presence for the real world. In *MONET 7(5)*, pages 365–376.
11. Leonard Kleinrock. Nomadicity: Anytime, anywhere in a disconnected world. In *Mobile Networks and Applications 1*, pages 351 – 357, 1996.
12. E. Kühn and G. Nozicka. Post client/server coordination tools. In *Proceedings of Coordination Technology for Collaborative Applications, Springer Series Lecture Notes in Computer Science*, 1997.
13. Tobin J. Lehman and Allessandro Garcia. Tspaces services suite, 2001. See http://www.almaden.ibm.com/cs/TSpaces/services.html.
14. Tobin J. Lehman, Stephen W. McLaughry, and Peter Wycko. T spaces: The next wave. In *HICSS*, 1999.
15. Peter R. Pietzuch and Jean M. Bacon. Hermes: A distributed event-based middleware architecture. In *Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, 2002.
16. Siegfried Reich, Martin Schaller, and Rupert Westenthaler. Developing advanced multimedia presentations with Java. Technical report, Sun Microsystems, June 2001. Presentation T541 at JavaOne, San Francisco, June 2001.
17. Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999.
18. Jim Waldo. Jini^{TM}technology architectural overview. Technical report, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 U.S.A., 1999. Available as `http://www.sun.com/jini/whitepapers/architecture.html`.