Designing Middleware for Smart Spaces

Institute for Infocomm Research, Singapore
daqing@i2r.a-star.edu.sg

Abstract. Smart spaces augment people’s lives with ubiquitous computing technology that provides increased communications, awareness, and functionality. However, smart spaces pose many technical challenges in device interaction/integration, user/environment perception, system interoperability and scalability, as well as security/privacy/trust. In this paper, we first examine some key issues in smart spaces, then we propose a reference middleware architecture for smart spaces which can well address the identified issues. Implementation details of the middleware are presented.

Keywords. Service-oriented Architecture, Middleware, Smart Space, Context Awareness, Device Interaction, Scalability, Interoperability, Trust

Introduction

Ubiquitous computing envision the future physical spaces such as homes, cars, hospitals etc. augmented with stationary and mobile devices/sensors/actuators. Those physical spaces which can provide us with a wealth of environmental information and thus empower the occupants to intelligently interact with the environment are often called smart spaces.

Smart spaces have posed a number of significant challenges for the system architecture. The first challenge is how to enable the heterogeneous devices interact with each other, for both static and mobile devices in smart spaces. The second challenge is how to acquire and understand the context information, typical context information is about who, what, when and where of the entities in smart spaces [1]. The third one is how to represent different entities in smart spaces such as people, devices, things and software functions so that a unified programming model can be based on. The fourth one is how to ensure the scalability of smart space solutions and another challenge is about security/privacy/trust guarantee among the interaction of entities.

Numerous software architectures have been proposed for smart spaces. Oxygen of MIT focus on human-machine interaction, self-configuration, system programmability [2]. Gaia from UIUC applies the resource management approach in operating system for Active Spaces, it focuses mainly on context-awareness, security and programmability of smart space [3]. Aura at CMU adopts a task-driven approach for smart space support, the focus is on user mobility and context-awareness [4]. At UC Berkeley, the Ninja project deals with distributed Internet services, it presents an architecture for secure service discovery [5]. Context Fabric develops a service-oriented context-aware infrastructure with event and query services, it mainly addresses the privacy issues [6]. Cooltown in HP applies web technology in smart spaces and
enhances physical objects with web content [7]. Gator House from University of Florida uses service-oriented framework to handle device self-integration, context-awareness and knowledge management in smart home [8]. However, none of the architectures have addressed context-awareness, spontaneous interaction, trust and scalability at the same time. In this paper, we intend to develop a middleware for smart spaces, where spontaneous device interaction, context-awareness, system scalability and interoperability, as well as trust will be supported.

1. Challenges in Building Smart Spaces

In this section, we will examine several technical challenges that we feel must be overcome before the vision of smart spaces becomes a reality. Those challenges are by no means an exhausted list of issues posed, however, they do represent some of the key issues for building smart spaces. Those challenges include:

- Device interaction and integration
- Context processing and management
- Interoperability of heterogeneous entities
- Security, privacy and trust
- Scalability in terms of device, service and space

1.1 Device Interaction and Integration

Smart spaces are expected to contain large number of devices which interact with one another to achieve different goals. The interactions are characterized by a number of challenges: device interaction can be spontaneous; environments and devices are heterogeneous; and context of user, device and environment are dynamic.

Currently, the developers must pre-program devices to recognize the specific protocols, standards, data formats and operations of all the peer device type they expect to encounter in order to talk to each other. With the rapid increase in numbers, types and operating domains of the devices and the services they may provide, it is unreasonable to expect that every device will have prior knowledge of every other type of device.

There are two approaches to achieve spontaneous interaction between devices: centralized control and peer-to-peer collaboration. Centralized approach utilizes servers to aggregate information for all the devices registered in a local environment which will facilitate the communication and interaction between the client devices and the environment. The key challenge for the centralized approach is developing an open service architecture that allows the heterogeneous client devices to control the devices in a new environment [9], and yet makes minimal assumption about standard interfaces and control protocols. A data-centric scheme provides the solution to the challenge. It utilizes an interface definition language enabling exported object interfaces to be mapped to client devices control interface. The control messages are thus generated as the RPC command sent from the client user interface to the corresponding service daemon. The centralized approach is preferable in a relatively static environment, however, for the scenarios where most devices tend to join/exit the environment freely, frequent registering of the devices and updates of the information bring excessive
burden to the system. Therefore peer-to-peer collaboration which directly enables the interaction between two peer devices is more desirable in dynamic environment [10].

With peer-to-peer collaboration ability, a device can connect to another device, provide metadata about itself, be controlled and provide references to other devices. In order to accommodate enormous heterogeneous types of devices in the world, the infrastructure must provide a generic approach for the interactions between devices. Instead of specifying a detailed, continuously evolving communication protocol, it defines a simple, fixed set of interfaces which allow the two devices exchange capabilities, communicate with each other and use whatever communication protocols are appropriate for the information transferring. In other words, the approach establishes the minimal set of development-time agreements to defer all other agreements required until runtime. It then delivers these agreements in the form of mobile code, which can extend a recipient’s behavior to make it compatible with a new peer.

1.2 Context Processing and Management

In this context information plays an important role in making the physical spaces ‘smart’. Users and applications in smart spaces often need to be aware of their surrounding context and adapt their behaviors to context changes. We believe an appropriate context management framework requires the following support:

- A common context model shared by all devices, services and spaces. Understanding context information is the basis for context sharing among devices and services in one smart space or across different spaces. An appropriate model should address different characteristics of context information such as dependency and uncertainty. In our earlier work, we have proposed an ontology-based context model [11] to describe context information in a semantic way, which exhibits features such as expressiveness, extensibility, ease of sharing and reuse, and logic reasoning support.

- Context acquisition, context lookup and context interpretation. These services are essential for building applications with context-awareness in smart spaces. Context acquisition is closely coupled with sensors to acquire context data from physical or virtual sensors. Context lookup provides user and applications both synchronous context query service and asynchronous context event notification service. Considering that context information is widely spread over wide-area networks across multiple smart spaces, a robust lookup service can be challenging. Such challenges can be, for example, building an underlying lookup mechanism to allow context lookup from anywhere in the system considering the temporal characteristics of context information. Context interpretation provides the support of deriving high-level context from low-level context. Different interpretation techniques can be applied such as logical reasoning and machine learning. Our earlier work such as Semantic Space [12], SOCAM [13] provided the set of such services to build our middleware for smart spaces.

1.3 Interoperability of Heterogeneous Entities

Within a smart space environment, entities can range from sensors, objects, devices to software functions. Those heterogeneous entities interact and service one another to
complete different tasks. This sounds fine except that these entities are likely to originate from different sources and therefore use different ways to present their capabilities and connectivity requirements. As a result, entities within a smart space will not be able to interoperate with one another. This interoperability problem can be addressed by using service-oriented architecture.

Service-oriented architecture (SOA) is a software architectural concept that defines the use of services to support the requirements of entities [14]. In a SOA framework, entities in the environment are represented in form of services and made available to other entities in a standardized way.

Once the descriptions of entities are defined in format understandable to entities, other mechanisms such as service announcement, service registration, service discovery and composition are needed for advanced service management.

1.4 Security, Privacy and Trust

In smart spaces, the interaction and information exchange between entities and services must be secure, private and trustworthy. Security includes the three main properties of confidentiality, integrity, and availability [15]. Confidentiality is concerned with protecting the information/service from unauthorized access; integrity is concerned with protecting the information/service from unauthorized changes; and availability is concerned with ensuring that the information/service remains accessible.

Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information is communicated to others. Privacy is about protecting users’ personal information.

A new enabling component of smart space is trust management developed as a trust specification that can be analyzed and evaluated before appropriate interactions/transactions really starts. As entities and services in smart spaces often interact and collaborate with each other in an ad-hoc manner, those entities and services may come from unfamiliar administrative domains and therefore be completely unknown to each other. To safely take advantage of all the possibilities, it is essential to assess the confidence level on involving parties, estimate the likely behavior of entities, and recommend a certain level for interactions, which is the so called trust management. The trust management involves the trust model, the trust and recommendation specifications for different entities from different certified authorities, and risk threshold. In smart spaces, the trust management service uses the trust model to compute the trust value for each interaction.

1.5 Scalability in terms of Device, Service and Space

Scalability is a critical requirement in designing middleware solutions for smart spaces with many pervasive devices and services. A robust design is to not only enable the middleware scale up to a large number of devices and services in a single smart space, but also provide the supports across multiple smart spaces. Many existing systems deploy a centralized approach. With the scalability requirement, this approach may not be suitable due to its single processing bottleneck and single point of failure.

As our earlier attempt, we have built a semantic peer-to-peer (P2P) overlay network [16][17] to facilitate efficient lookup for context information in multiple smart spaces. The basic idea is to cluster peers based on their data semantics and organize
them in a semantic P2P overlay network for efficient lookup. Context data which are semantically similar are "tied" together in the system so that they can be retrieved by a query using the same semantics. While the system is designed for context lookup, the similar approach can be used for device and service lookup across multiple smart spaces.

2. A Service-oriented Middleware for Smart Spaces

In order to address the challenges posed by smart spaces, we proposed a reference middleware architecture as shown in Figure 1. The proposed middleware takes the service-oriented system architecture, it contains a number of collaborating components, namely physical and software entities, wrappers, service platform, context management services, and trust management services. As the middleware leverages on the device interaction mechanism and service-oriented architecture as elaborated in last section, it can accommodate the evolution of physical and software entities quite well. With the incorporation of inter-space service/context discovery mechanism in middleware, it is scalable in terms of device, service and across spaces. The following are the major components of the middleware:

- Physical and software entities. Physical entities consist of various physical devices, objects, sensors and actuators which are embedded or used in the environment. Typical physical entities include lamps, TVs, clock radio, doorbell, chairs, beds, temperature sensors, motion sensors, stove, and robots. Software entities are those functions and applications which can provide input or output to the environment.

- Wrappers. Wrappers are software agents or proxies which bridge the service platform with the physical and software entities. Wrappers transform the functions of those entities into the form of services and publish those services in the service platform for other services to access. Developers can thus compose any service for certain applications without having to understand how the physical and software entities work. Decoupling entities from services provide great flexibility and openness to integrate new components into the smart spaces.

- Service Platform. Service platform contains the service framework such as OSGi, CORBA or .NET which manages the life cycle of services. Basic services refer to functions provided either by physical or software entities, quite often the service framework also provides certain standard basic services such as communication service, registration service, and discovery service etc. Developers create composite services by combining basic services.

- Context Management Services. The context management services are essentially the context processing functions, all those functions are implemented in the form of context services. Context information is obtained from an array of diverse information sources and represented based on a shared context model. The context aggregation service merges the interrelated context and puts all the derived high-level context in the centralized knowledge base. With the context query service, applications can register and query their contexts of interest. The applications and composite services which depend on certain context information are called context-aware services.

- Trust Management Services. The trust management services deal with the risk
assessment and decision recommendation in the service framework. From the service providers’ perspective, all the service consumers will be assessed before they are allowed to access or consume certain services, the trust value indicates the degree of confidence and is used as a parameter to determine the access level of services. From the application point of view, all the physical and software entities will be assessed before they are allowed to provide services, the trust value associated with the services is based on the reputation of the provider, the quality of the services, etc..

3. Implementation of Middleware for Smart Home

We have developed most part of the reference middleware architecture for smart homes, several applications supported by the proposed middleware have been implemented to illustrate the key features of the middleware.

One application is to enable spontaneous interaction between devices that have little a priori knowledge of one another. The aim of developing this application is to test the middleware support for spontaneous device interaction. We carried out a trial of our middleware using the application scenario of a smart home environment, whereby a mobile Personal Device (e.g. PDA) without UPnP support is brought into the environment housing UPnP-enabled Plasma TV and Media Server (shown in Fig. 3). In this trial, notebook computers are used to act as Personal Device, Plasma TV and Media Server. In this application scenario, even though the Personal Device does not have a priori knowledge of the Plasma TV and the Media Server, the middleware in smart home “teaches” the new devices and equips it with capabilities to communicate with the Plasma TV and the Media Server. In this way, the Personal Device can discover that the
Media Server contains video files that the user wants to play on the Plasma TV, and the user is able to use the Personal Device to control the playing of the video.

![Fig. 3. Spontaneous Device Interaction](image)

The second application is named SituAwarePhone with the intention of illustrating the middleware support for context-awareness [12]. In this application scenario, users’ context can be captured, processed and queried by the middleware, users’ applications can be made context-dependent according to users’ intention. For example, when John is in an important meeting, his mobile phone can be automatically set to silent mode. Upon John returning to “interruptible” state (i.e. ending meeting), the mobile phone is automatically set back to normal mode.

The third application developed is the service/context lookup in multiple spaces [16][17]. The aim of this application is to test the scalability of middleware architecture for multiple smart spaces. In this scenario, context and services are stored in a local repository. When applications in one smart space need to access context and services in another smart space, efficient algorithms are needed to ensure the performance. Several P2P overlay solutions have been implemented for systems with thousands of spaces (Fig. 4).

![Fig. 4. Inter Multiple Space Context/Service Lookup](image)
4. Conclusion

In this paper, we discussed some key issues and challenges in design middleware for smart spaces. Based on the discussions, we have proposed and implemented a reference middleware for smart spaces. The middleware uniquely supports the spontaneous device interaction, context-awareness, system scalability and interoperability, as well as trust which represent some of the key requirements for future smart spaces. While the middleware developed by us showed some encouraging results, we still need to further validate its effectiveness by carrying out more trials using different application scenarios and perform trials of larger scale. We plan to deploy our middleware in real smart home called StarHome which is located next to our research institute and is solely used to showcase smart home technologies. We believe that more comprehensive trials will guide us to further enhance the performance and features of our middleware.

References