A Services Platform for Home-Automation Based on Decentralized P2P Architectures

Sandra S. Rodríguez Valenzuela, Juan A. Holgado Terriza
Software Engineering Department, University of Granada,
C/ Periodista Daniel Saucedo Aranda, s/n. 18071 Granada, Spain
{sandra, jholgado}@ugr.es

Abstract. The vision of Home Environment is changing towards living interaction space populated of interconnected devices, services that encapsulate the functionality, and multiple interfaces through which the user can interact with these devices, in accordance with vision of ubiquitous computing. This paper presents a pervasive service platform for ubiquitous spaces based on distributed architecture P2P using JXTA technology, with an innovative approach, that favors the building of collaborative services from proactive entities, the peers. These ones are capable to establish dynamic intercommunications synchronizing with others, form coalitions to cooperate with others for a common purpose, and are self-organized into groups.

Keywords: Home-Automation, Ubiquitous Computing, SOA, P2P, JXTA.

1 Introduction

Home-automation or domotic is a traditional application domain where the problems of heterogeneity on devices, networks, hardware platforms, and software frameworks have led to the proposals of multiple technologies and commercial products; i.e, domotic buses as Konnex, Lonworks, or HomeRF, software technologies as Osgi, HAVI, etc. [1]. The potential market is very large, but the expectative is not evolved as it expected for several reasons. First, the complexity, lack on security and high costs on installation, deployment and maintenance of such systems have obstructed the full acceptance on final potential users and building promoters. Second, the availability of a large number of domotic solutions does not help the user to the selection of the best right product, but on the contrary it has created more confusion, since the different products are not compatible, and when a user chooses a product, he or she is restricted to the technological offer provided by this product. To overcome these difficulties in our opinion we should change our vision. We can see the home environment as a living interaction space populated of interconnected devices, home services that encapsulate the functionality or compositions of functionality, and multiple

interfaces through which the user can interact with these devices, in accordance with vision of ubiquitous computing.

The ubiquitous computing envisages new interaction spaces where transparent and invisible interactions has taken place among users and varying-scale devices, increasingly small, of minor cost, of minor energy consumption, and with a high level of cooperation between them. These interactions should be transparent and invisible with respect to the rest of the devices and users in the sense that they are not intrusive but at the same time they are natural from user's viewpoint. The devices in such environments are heterogeneous, ranging from limited small-scale embedded systems to powerful computers, and many of these devices can be mobile [2].

In a high level sense, the pervasive space can be seen as an abstract logical environment where the devices provide compositions of functionality that users claim to consume. The development of software platforms that provide such abstraction should deal with the device heterogeneity, the mobility of devices or users, the seamless access to functionality and information resources, secure and reliable communications with fault tolerance, the interoperability among different networks, the evolving, extension and adaptations of functionality, and finally the interfaces between user and devices. Some of these issues required for pervasive systems are performed on Distributed Computing domain like remote communication, fault tolerance, high availability, remote information access and distributed security; and Mobile Computing domain, like mobile networking, mobile information access, adaptive applications, power consumption control and location-aware sensitivity [3]. However, the ubiquitous environments requires new and specific computing models, and therefore, new software infrastructures for supporting scalability, heterogeneity, integration, invisibility and perception [4], and new kinds of devices with limited resources, low power, low memory and high connectivity [5].

The highly complexity of pervasive systems makes difficult the development of applications with the aforementioned non-functional properties. A well-know strategy to overcome this complexity is the use of middleware technologies. The middleware is a software infrastructure placed between the execution environment (operating system or kernel) and the application layer of each device. It hides the heterogeneity of the underlying hardware device and network layer, and provides at the same time a pervasive programming model to simplify the development of applications built on top of it. The developers may have specific mechanisms for establishes point-to-point communication, the semantics for achieving coordination, the access to remote resources, etc. Furthermore, the middleware drives the way in which an application is built and deployed, addressing the system architecture, and therefore, the organization of the system, their main components and their relationship. The architecture of a distributed system can be centralized or decentralized, depending on where is the main component of the system or controller. Distributed decentralized computing marked the next step toward pervasive computing by introducing seamless access to remote information resources and communication with fault tolerance, high availability, and security [6].

A large number of middleware technologies have emerged to give support to ubiquitous computing based on centralized and decentralized architecture such as Gaia, Aura, etc [4]. But most recently it has appeared middleware proposals based on SOA (service oriented architecture) paradigm. SOA is an architectonical style that establishes the intercommunications between the devices in terms of collaborations of coupling-loosed entities, the services. This work presents a new proposal for the design of a ubiquitous services platform using Java technologies based on a decentralized architecture P2P. The new platform is based on JXTA, a P2P middleware technology, which is more scalable, interoperable, dynamic and extensible than other platform based on centralized client-server architecture.

The remainder of this paper is organized as follows. Section 2 examines different SOA middleware approaches for the development of ubiquitous spaces. Section 3 shows the basic of JXTA technology and its components. Section 4 presents the services platform for home-automation developed on top of JXTA platform, its architecture and design decisions taken during the platform development. Section 5 compares our approach with other related works. At last we present the conclusions of this work.

2 Trends in the development of ubiquitous systems

Ubiquitous spaces envision seamless support for user needs within a set of invisible cooperating devices populated the environment [2]. Therefore the resources of a ubiquitous system are dynamic by nature and they should be available to users any time in everywhere. An architectural style widely accepted that favors the principles of pervasive computing is SOA (Service Oriented Architecture). SOA imposes restrictions on the components organization, arranging the pervasive space in terms of loosely coupled entities, the services. Hence the functionality provided by the devices can be abstracted and encapsulated in terms of services, which may be combined with others in a compositional fashion to obtain more complex applications [7].

There are different alternative technologies conformant to SOA principles. A best known Java technology is the Sun Jini platform [8]. Jini platform aims to build spontaneous distributed systems based on the concept of federation services with a client-server architecture. Jini architecture provides a general framework for the registry and dynamic discovery of services, characteristics that can be adapt to support pervasive spaces [9, 10, 11].

In distributed applications JavaSpaces technology supplies a virtual space between providers and requesters of network resources or objects [12], building upon a distributed shared memory based on tuplespace. A function of a JavaSpaces service is to notify entities interested in objects when they are stored in the space. From the viewpoint of ubiquitous computing this shared space would be used for services to carry out the tasks of publication and discovery [13].

The most widespread SOA alternative in the field of ubiquitous applications is the Web Services [14]. Broadly speaking, the web services framework is designed

to simplify the process of defining standard mechanisms to describe, locate and communicate with remote applications. Such applications become web services components over the network, described by open specifications. In an ubiquitous system web services have a WSDL or service description that publish in a common UDDI, where other services can discover it and use it to interact with them using SOAP as communication protocol.

Message Oriented Middleware (MOM) is a middleware based on distributed message-queues where the applications establish communications with each other by asynchronous messages. Non-blocking message passing technique can be used in a ubiquitous system to provide low coupling among applications or services and to simplify the coordination of them across a network [15].

It can also address the discovery, collaboration and delivery of services in ubiquitous computing architectures using peer-to-peer (P2P). In this architecture all nodes can develop any kind of role, client or service. The peers perform a communication between equals that promotes the deployment of decentralized applications, and also ubiquitous systems. The P2P architecture most used currently is JXTA, for its completeness and openness. JXTA also has less administration management concerns than others, i.e. JINI, and it is better suited for limited capability device. This technology is used in several studies of ubiquitous computing that works from the peer abstraction to develop a system based on sensors nets, agents or services [16, 17, 18].

As conclusion the Table 1 summarizes the particular features of the technologies discussed in this section.

Table 1. Comparison of different technologies applicable to ubiquitous systems.

	Jini	JavaSpaces	Web Services	MOM (JMS)	JXTA
Data representation	Java Marshalled Object	Collections of information	XML	Messages	XML
Transport	Based RMI protocols	Serialization	SOAP over HTTP	Based RMI protocols	Usually TCP/IP o http
Services description	Java interface	JavaSpace interface's contract	WSDL	Administered objects	Advertisement
Services location	Jini Lookup Service	Entries	UDDI	Publish/Subscribe or point to point	Discovery Service
Language bindings	Java	Java	Java, .NET, Perl	Java	Java, C...
Remote reference	Proxy objects	Proxy objects	URL	Identification	Peer Rendezvous or Relay
Synchronicity	Synchronous/ Asynchronous		Synchronous	Asynchronous	Synchronous/ Asynchronous
Arquitecture type	Client-Server	Client-Server	Client-Server	Client-Server	peer-to-peer
	Jini	JavaSpaces	Web Services	MOM (JMS)	JXTA

3 JXTA as support of ubiquitous space

JXTA provides protocols, services, programming models and mechanisms that simplify the development of P2P applications in standard fashion [19], and ensure the interoperability, the platform independence and the ubiquity. The term JXTA is a shorthand of juxtapose, motivated by the fact that it breaks with the client-server model, which is today the model of traditional distributed computing [20].

The JXTA project involves different types of services, protocols and mechanisms to facilitate the construction of P2P decentralized systems in general and ubiquitous systems in particular. To do so, JXTA is structured in three software layers with different responsibilities. The *Core Layer* is the JXTA core responsible to hide the heterogeneity of networks and devices, and encapsulate the main primitives of P2P distributed applications such as peer, pipes, advertisements and groups. This layer have the suitable mechanisms to carry out the discovery of peers, transit advertisements, even through firewalls, the creation and management of peers and peers groups, and other security primitives. The *Services Layer* includes network services useful to P2P applications such as authentication, search, indexing and translation of protocols. Finally, the *Applications Layer* provides the implementation of P2P integrated applications that make use of the lower layers.

JXTA defines a virtual network over heterogeneous physical network by means of interconnecting peers. A peer is any node or networked device that implements one or more JXTA protocols and operates independently and asynchronously from the rest of the peers. The protocols defined in JXTA help peers to discover each other, to interact, and to manage P2P applications. The peers are self-organized in collaborative peers groups, providing a range of related services, which simplifies the interconnection of peers, and the transmission of messages between them. To communicate with each other the peers use pipes. JXTA protocols support and encourage different types of pipe, but the most common are the unidirectional asynchronous pipe, unicast reliable secure pipe and bidirectional pipes [21].

All the information about peers, groups, pipes, and other JXTA constructs are managed by advertisements published in the network. These ones are XML documents that allow other peers to learn how to connect and interact with a peer. Thereby the messages are a special type of advertisement. Message advertisements are used for various messaging protocols, as well as for user defined messages. There are two different types of messages, XML and binary. The binary message is offered in most cases because it is much more efficient. The messages are seen as the most used type of data that is transported using the pipes between peers.

4 Dynamic Open Home-Automation services platform

A new distributed, ubiquitous, decentralized and dynamic platform is developed to facilitate the access, control and management of pervasive spaces from any

computing device such as portable computer, portable device or embedded platform. This platform, named Dynamic Open Home Automation (DOHA), is based on JXTA architecture and provides a full remote control of living environment in terms of services, according to SOA paradigm. A service in the context of DOHA is an autonomous self-contained component capable of performing specific activities or functions independently, that accepts one or more requests and returns one or more responses through a well-defined, standard interface. A service that provides functionality is a provider service, while the service that requests a service is a consumer service.

Some important properties were considered during the development. The modularity is achieved by implementing each service as a separate module in the system following the SOA approach. So, each module is built as a black-box component that can be deployed at runtime, regarding of the rest of services. It allows the compositionality of services and the addition of new services into the platform at runtime with low cost, guaranteeing the scalability and flexibility of the system. Another important feature of the system is the capability of service distribution on peers, reducing the bottleneck that is typical on centralized systems.

DOHA platform hides the physical distribution of devices as JXTA peers. This abstraction allows working with logical spaces based on services at high level, as it is shown in figure 1. The cooperation between services at *services net* level involves communications between peers at *virtual net* level, and finally point-to-point communications between devices placed in different subnets at the lowest level. DOHA platform is designed to facilitate interconnection of widely dispersed service nodes across the network and provides loose coupling of services and a dynamic model of operation.

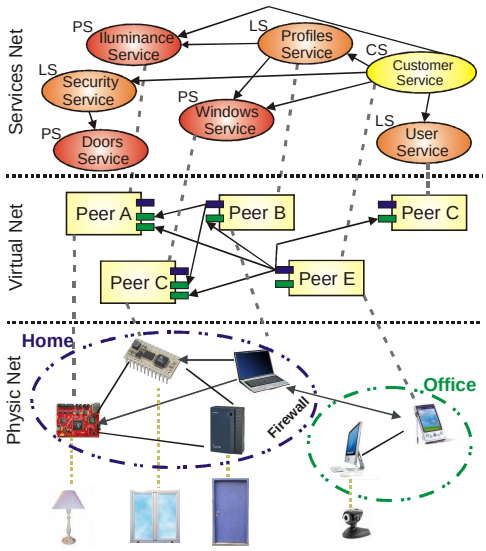

Fig. 1. Levels of abstraction of DOHA platform.

4.1 DOHA platform description

The use of a distributed decentralized architecture enables the integration of different entities in the same abstract space, such services, physical devices, applications which need to use services, etc. So, the system can naturally be scaled by adding new components in a more flexible way. Each application of the distributed system acts as a service. Further these services can collaborate and communicate with other applications that are running within the system environment in a more flexible and dynamic fashion.

Each service of DOHA is characterized by a multilayer architecture. The multilayer structure consists of several design layers that decouples the tasks performed by a service in components. This procedure facilitates the implementation and deployment of services, providing components to control the state of the service when a requested is accepted, the interactions of the service with the rest of services, etc. The *Interface Layer* guarantees the widespread access to services from any other element of the system. The *Application Layer* abstracts the functionality itself of a particular service. Finally the *Interaction Layer* contains the logic needed to be able to communicate and collaborate with other services. The selection of this architecture is performed for the benefits that suppose the reusability of the system components and its adaptability. Then, any change in a component does not affect any other components, and we can replace this component by another component in a transparent way, preserving always the same service interface.

We have identified three types of system services depending on its role and responsibility: Customer Service (CS), Physic Service (PS) and Logic Service (LS). The applications may have different roles, and consequently can act with several service's role in the system. Customer service is a system application that invokes the use of a specific service (provider service) or several services to perform a user's requirement. This one mainly should interact with users, providing a simple and natural way to modify the behavior of their living environment. The Physic Service is a provider service which interacts directly with hardware device of the system (sensor or actuator). It can provide the state of the living environment or can change it depending on the requests performed by other CS services. Finally the *Logic Service* (LS) has specific functions to provide a specific task either alone or in collaboration with other services.

Figure 2 shows the multilayer architecture of each service in relation with the service's role. The customer service can access to the DOHA platform using the interface layer of tCS service. Using the application layer, the CS service interacts with the LS and PS services across the interface layer to obtain its functionality. The interaction layer is used by the LS and PS services to interact or collaborate with other services. Finally, in the application layer all services have its own functionality and its specifics details.

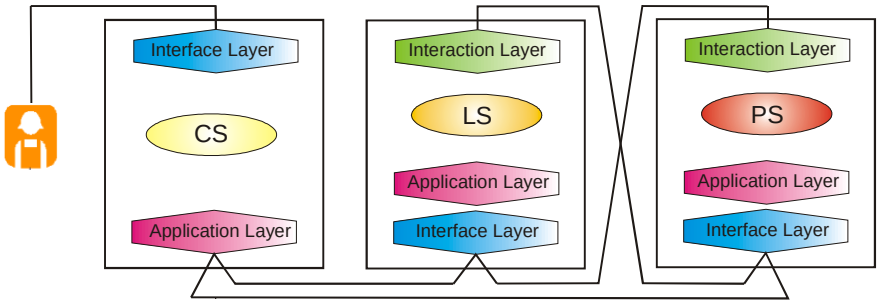

Fig. 2. Software Layer Architecture.

DOHA platform is composed of several software hierarchical layers that should be loaded on each device. Figure 3 shows the architecture of DOHA platform. The different types of services, depending on their role, use JXTA mechanisms and protocols to carry out its functionality into the system. The physic services also use JAHASE framework [22] to access or query the environment devices.

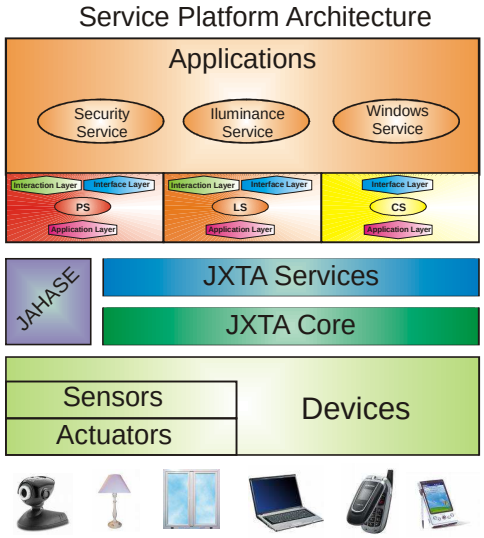

Fig. 3. DOHA Services Platform Architecture.

4.2 Communication and collaboration mechanisms

The communication among DOHA services is carried out using JXTA pipes. A service has a pool of pipes with a name, an unambiguous identifier, and a pipe advertisement for each one. This information is known by the rest of the services

in peer group due to its publication into a pipe advertisement, and in addition it could be delivered by remote networks through Rendezvous Peer. When a service wants to establish communication with each other, first it seeks one of these announcements in his cache or from Rendezvous Peer, and therefore it uses this information for the pipe creation.

The network communication protocol is open regarding the format of messages. The applications are free to implement the data protocol more adequate to its functionality. In the applications developed to prove the DOHA platform we use a data protocol based in XML. XML is quickly becoming the default standard for data exchange and provides a universal, language-independent, and platform-independent form of data representation.

The services can also establish mechanisms of collaboration among them based on orchestration or choreography. Orchestration always represents control from one party's perspective. This distinguishes it from choreography, which is more collaborative and allows each involved party to describe its part in the interaction [23]. When a service needs to collaborate or interact with each other, it must discover the needed announcement of service and then it reserves its utilization. The services have a special pipe, named the interaction pipe or Pipe I, to claim the use of collaborated services in a similar way such as input pipe in a customer service. This pipe allows to the services to act as clients in the network. Usually, a service executes an iterative loop to receive the requested messages by the clients, and use the *Pipe I* to communicate with other service such a client.

4.3 DOHA platform design

We take some design decisions during the DOHA platform development to ensure features such as robustness, scalability and security, of great importance in ubiquitous computing systems. With these decisions we also guarantee the SOA principles of loose coupling, encapsulation, abstraction, reusability, composability, autonomy, optimization and discoverability.

In figure 4 we can show the main ingredients that performs DOHA services platform. Each service is managed by a peer that is part of a peer group. The peer group is composed of a lot of peers through which it can interact. The peer group has a Rendezvous/Relay peer that allows the discovery, communication and interaction between remote peers. The services have a specification, the service’s contract, to describe the services, its functionality and its methods. This specification is registered in a public directory of the peer and can be access via CMS (Content Manager Service) remotely. The development of PS services requires the interaction with JAHASE as general platform to access hardware devices. It hides the particularities of each hardware device such as sensor or actuators, and facilitates its management. The availability of services for several consumer services is performed by the implementation of multithreaded peers and reliable transmissions between peers.

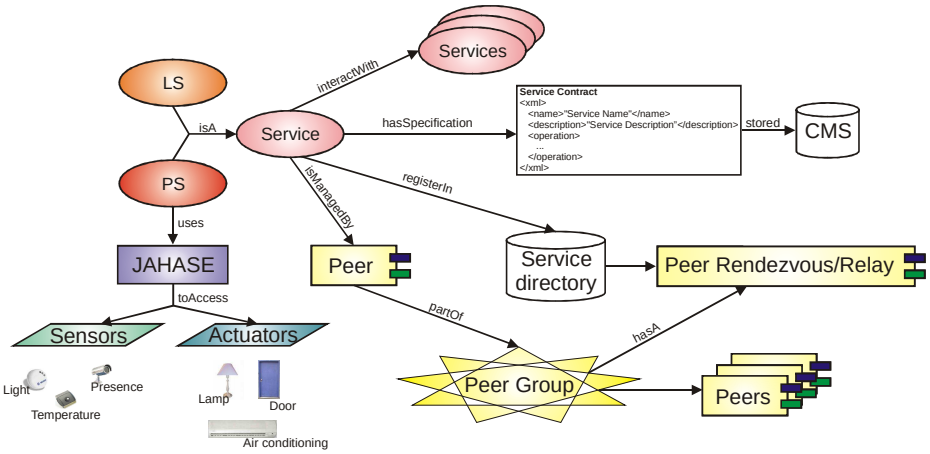

Fig. 4. Main elements of developed DOHA platform based in SOA principles.

Discovery mechanisms. To ensure the loose coupling between DOHA services are needed discovery mechanisms that allow communication among services with different locations and functionalities. In JXTA exists special peers, the Relay Peer and the Rendezvous Peer, that provide the remote discovery of advertisements between peers in different networks.

These Rendezvous/Relay peer nodes can relay requests and responses to facilitate communication between pairs of peers that are unaware of each other's identity. Alternatively, peers can invoke calls directly if identities are known [24]. JXTA can operate on a strict peer-to-peer basis and therefore does not require a specific service node to provide registry services and so on. JXTA is versatile to accommodate a brokered mode of operation as well, whereby Rendezvous/Relay nodes can take over the role of the Registry and Lookup servers.

When a service discovers new advertisements in the network, it stores these ones in its local cache. If this service is also connected to a Rendezvous peer, it can also do a remote search and discover more advertisements. A Rendezvous peer has the responsibility of coordinate all peers in the JXTA net and propagates remotely the messages and advertisements. If the peers are in separate subnets, we can use almost one Rendezvous peer to manage the reception of remote messages and broadcast these ones within its local net. If the local net has a firewall or NAT (Network Address Translation), the peer can use a Relay peer to surpass it and allow remote discovery of its advertisements by peers of other external nets.

Services contracts. Each service has specified its functionality by means as services contracts. These contracts are specifications with a description of the service, its functionality, the methods that can be invoked from other services or customers appliances, and other important information. The contracts in XML format are human and machine-readable, and can be downloaded the first time the invoker requests the service.

The information contained in a service contract hides the logic functionality of the service, but instead provides the specific functions that may be invoked, its signature, their preconditions and their postconditions. Also, the contract can show other meta-information related with non-functional requirements such as communication protocol, assets of quality of services (QoS), availability, etc.

In the following example a template of a service contract is shown, in which the developer may define the specification of the service and the provided functionality independently of communication protocols used by consumer services. For example, in a home service's contract it exposes the name, the description of the service, and the operations that can be requested.

Code 1. Example of XML structure for a *service contract.*

```
<xml>
<service name="Service Name">
  <description>"Description Service"</description>
  <type>"Presentation|Process|Business|Data|..."</type>
  <functional>
    <requirements>
      "Functional requirements of the service"
    </ requirements >
    <operation>"Operations of the service"/operation>
    <invocation>"TCP|Bluetooth|..."</invocation>
  </functional>
  <not_functional>
    <Security_Constraints>...</Security_Constraints>
    <Quality_of_Service>...<Quality_of_Service>
    <Service_Level_Agreement>...<Service_Level_Agreement>
  </not_functional>
</service>
</xml>
```

CMS. The static information sharing in ubiquitous interaction model requires a remote access that should not be affected by the information dispersion among the network elements. The Content Manager Service is a component of each DOHA service managed by a peer for sharing content among multiple peers from a group included in JXTA, and that is used by peers to download the service's contracts of the services with which interact. Each service has its own public CMS directory to share the service's contract and other public resources. When a consumer service discovers a new service advertisement and wants to make use it, it takes via CMS the service's contract file from the remote shared folder and updates its own CMS directory and with this information then it can proceed to reserve the use of service directly.

Development of Physic Services. PS services (e.g. illumination service) are the services tight with physical environment through sensors and actuators. The main responsibilities of a PS service are related with the monitoring of the state of the environment and the actuation to modify the state of the environment. To facilitate the management of physical device, we can implement the PH service on top of JAHASE and JDOMO framework [22]. JAHASE gives to developers a software

infrastructure to facilitates the Java programming of different kinds of embedded and microcontroller devices with varying memory and processing resources, providing common access to digital buses as I2C, SPI, etc, and digital and analog IO ports. JDOMO at top of JAHASE implements high level abstraction for home-automation devices such as a light, presence sensor, or a HVAC regulator. This abstraction can be used to facilitate the management of these devices control, although the developer can create his own class to control the devices.

JDOMO-JAHASE framework also provides a graphic interface to view the changes causes in low level over the embedded devices. It guarantees to developers a correctly use of the devices and the corresponding actions to modify the device's state.

Multithreaded peers. A service must be able to provide its functionality to all customers who require it anytime, for which is need to manage concurrently the received request messages. So that we can ensure their full availability, feature of great importance in a system of ubiquitous computing.

But the communication model among JXTA peers requires that a pipe advertisement has been published by each peer and take by other peers to establish communication point to point. There are two possible solutions to provide a multithreaded service for each peer of the system, a single input pipe for all peers or an input pipe for each customer peer.

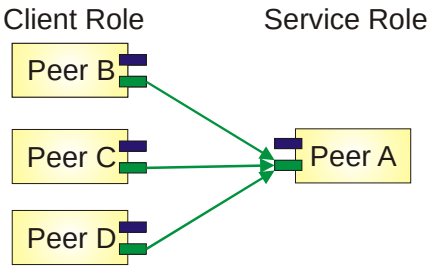

Fig. 5. Single input pipe.

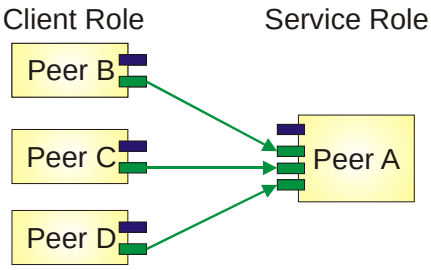

Fig. 6. An input pipe for each peer.

If we use a single input pipe, such abstraction serves as a synchronize element, not allowing receive messages simultaneously from two different peers. In figure 5, i.e., three peers, B, C and D want to obtain the service of the PeerA, and all use the unique peer that is available. When the service attended a request of client PeerB, for instance, it cannot attend the requests of the rest, PeerC and PeerD.

Another option is to consider an input pipe for each customer peer, thus it may receive messages concurrently at each input pipe. This would require implementing a flow control with an independent thread to address messages that are delivered in each pipe. In figure 6 the peer with a service role, PeerA, has one input pipe for each peer that wants to interact with them, three in this example.

The second solution could involve a substantial increase in pipes and, consequently, the need for a larger pool of memory that can be important and problematic in cases of using computing devices with limited memory resources.

When performance in real time is also required, this solution cannot be carried out because the assignment of pipe for each client peer requires a couple of dynamic memory reserve and release which could affect the maximum response times, and also could fragment the memory.

We propose to use a hybrid model of the two solutions with a controlled pool of pipes to meet the requests made from other services with more flows of control to ensure the availability but instanced from the beginning of the activity of the service. At the communication level this solution requires the availability of independent pipes that allow concurrent receipt of messages from different users to services. When someone reserve a service, also reserves a communication pipe, which will lead in turn associated an execution thread.

Detection and control of network failures. In a ubiquitous computing environment it is essential to ensure the quality and simplicity of the nodes network that sustains it. The distributed and dynamic nature of P2P networks are not errors free due to overloading, communication failures, missed messages, etc., that may affect the proper functioning of the applications. Because of this, we have taken some design decisions that help to mitigate its potential consequences on the DOHA platform operation.

One important design goal is not to block the services infinitely waiting receive message in the pipe. There are several methods to implements the messages reception. With *InputPipe waitForMessage()* method the peer is block waiting the message reception in the pipe. If occur an error in the network and the message is not arrived, the peer will be block. To avoid it, rather than spending an infinite time in the input pipe waiting for a message, we are implemented a timeout facility using *InputPipe poll(timeout)* method. This provides protection to each peer and allows implementing a preventive mechanism for fault tolerance.

Other problem associated at JXTA communication is the publication and discovery of messages. The services public advertisements in the network, but they are not guaranty that these advertisements arrives to destination peers. To prevent excessive propagation of requests we have associated a limited lifetime at each advertisement, and after it, the advertisement dies. After the lifetime, the peer is need to public one more time this advertisement as a regular republication of advertisement in the net. Besides the peers also perform regular searching on the network for discover news advertisements.

5 Discussion of results and related works

In our study we have used the peer-to-peer architecture based on SOA approach in order to obtain a system with the major characteristics associated with ubiquitous computing. In DOHA services platform we have designed autonomous services capable to develop a collaborative behavior to carry out its functionality, with the publication and discovery procedures associated with them. A set of adaptations have been made on JXTA middleware, specifically in JXTA Core elements, to

optimize the system operation and get a behavior as independently and distributed as possible.

In the literature there are several studies in trends of construct ubiquitous systems from a P2P architecture. AMUN middleware (Autonomic Middleware for Ubiquitous Environment) is employed in the ubiquitous mobile agent system UbiMAS [17]. AMUN uses JXTA peer-to-peer system as communication infrastructure and uses an event-based approach for message delivery. An Event Dispatcher offers to services the functionality to send messages and to register themselves as listeners to specified types of messages. In this way the services can register for different message types and get informed when one of these messages arrives. The principal difference for our approach is that we use the same message type associated to all kinds of services. All services public advertisements in the network when are available, and these events are filtering in each service, without the needed of centralized event dispatcher figure. When a service is interested in these ones, takes their specification via CMS, where are the type and behavior of the service advertised.

Other similar approach is GAS-OS Architecture [15]. GAS-OS Kernel Communication Module is responsible for communication among GAS-OS nodes and implements a P2P communication adopting the basic principles and definitions of the JXTA project. The services implement protocols for resource and service discovery, advertisements, routing as well as the queuing mechanisms to support asynchronous message exchange in the same way that it is carried out in our study. In addition, DOHA platform complements these features with some additional design considerations such as multithreaded pipes, and detection and control of network failures.

In CoCA [18], a services platform is implemented as Collaboration Manager, a module that works based on peer-to-peer negotiation and communication protocols. The role of the collaboration manager in CoCA services platform is to share computing resources like context, rules, ontology, etc., to solve computing problems and to provide comprehensive context-aware service, which would otherwise be difficult and some times impossible for a single pervasive device to solve. This module uses the principle of virtual network overlay. Among the basic requirements for collaborative computing between CoCA services in the neighborhood space is the ability to self-organize into groups, discover each other and each other's services and resources, for what is used JXTA, as well as DOHA platform. However, the central figure of the Collaboration Manager can be a bottleneck in the system operations, which is resolved in DOHA platform using CMS among services.

Extensions to JXTA are used in [16] to accommodate mobility and limitations in wide area wireless technologies in the GPRS, WCDMA family. They have proposed several modifications to the JXTA functionality, such as Rendezvous peer, mapping and location service, and evaluated them in a real GPRS mobile network. The concept was proven successfully, it is possible to use JXTA in a mobile environment to provide a connectivity platform for individual service providers and service users based on the description of their service abilities and needs. In our case, DOHA services platform has been tested in a home automation

model with limited resource devices and we have obtained a good operation of whole system [25].

6 Conclusions

During DOHA services platform deployment we have carried out various stages of software development incrementally until obtain a complete and correct system. As a final result it has been obtained a services platform, open, dynamic, reliable and extensible. DOHA platform uses an peer-to-peer architecture instead of the traditional centralized client-server architecture that is generally used in the domotic proposals. With the decision to use a P2P network based on technology JXTA, the DOHA platform has won modularity, scalability and independence.

DOHA services platform allows you to develop services that are independent in terms of functionality, allowing the implementation and deployment of new services easily, based on the design of the overall structure established. Communicating through JXTA is complex, because peers and pipes are very susceptible to the network condition. However, the fault tolerant recovery mechanism implemented allows that services can recover from any failure of other services with whom they were communicating or interacting.

Likewise, the network overloading or other problems associated with distributed communication could affect the optimum operation to the applications. So it is important to establish an error control, storing all events susceptible to network stability, such as the amount of time that a service is awaiting receipt of a message, the messages correct reception, management groups, etc.

Another important goal is to develop an adaptation to DOHA services platform to include embedded devices with more stringent resources. The platform is successfully applied to large-scale embedded devices, but when the memory resources are scarce, there is no space for JXTA middleware. A variation of JXTA for J2ME (CLDC-MIDP2) recently appeared is expected to be used.

References

1. Romero, C.; Vázquez, F.; de Castro, C.: *Domótica e Inmótica. Viviendas y Edificios Inteligentes.* Ra-Ma (2005)
2. Weiser, M.: The Computer for the 21st century. *Scientific American*, Vol. 256, pp. 94-104 (1991)
3. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, Vol. 8, pp. 10-17 (2001)
4. Saha, D.; Mukherjee, A.: Pervasive Computing: A Paradigm for the 21st Century. *IEEE Computer*, Vol. 36, pp. 25-31 (2003)
5. Weiser, M.: Some Computer science issues in ubiquitous computing. *Mobile Computing and Comunication*, Vol. 3, pp. 12-21 (1993)
6. Banavar, G.;Beck, J.;Gluzberg, E.;Munson, J.;Sussman, J.;Zukowski, D.:Challenges: An Application Model for Pervasive Computing. *Mobicom ACM Press*, pp. 266-274 (2000)

7. Stojanovic, Z.; Dahanayake, A.: *Service-Oriented Software System Engineering: Challenges and Practices*. Idea (2005)
8. JiniTM Technology. http://www.jini.org. Access the 1 of June of 2008
9. Xu, B.; Gao, Q.; Yang, X.: Extensions to Jini Service Architecture for Pervasive Computing. *1st International Symposium on Pervasive Computing and Applications*, Vol. 3, pp. 90-94 (2006)
10. Kiani, L.; Riaz, M.; Zhung, Y.: A distributed middleware solution for context awareness in ubiquitous systems. *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Vol. 15, pp. 451-454 (2005)
11. Lee, Y.; Lee, S.; Lee, H.: Development of Secure Event Service for Ubiquitous Computing. *Springer Berlin*, Vol. 344/2006, pp. 83-94 (2006)
12. Rakotonirainy, A.; Indulska, J.; Loke, S. W.; Zaslavsky, A.: Middleware for Reactive Components: An Integrated Use of Context, Roles, and Event Based Coordination. *Springer Berlin*, Vol. 2218/2001, pp. 77 (2006)
13. Lee,S.; Lee,Y.; Lee,H.: Jini-Based Ubiquitous Computing Middleware Supporting Event and Context Management Services. *Springer Berlin*, Vol. 4159, pp. 786-795 (2006)
14. Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S.: Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, Vol. 6, pp. 86-93 (2002)
15. Drosos,N.;Christopoulou,E.;Kameas,A. Middleware for Building Ubiquitous Computing Applications Using Distributed Objects.*Springer Berlin*, Vol. 3746, pp. 256-266 (2006)
16. Krco, S.; Cleary, D.; Parker, D.: Enabling ubiquitous sensor networking over mobile networks through peer-to-peer overlay networking. *Science Direct*, Vol. 28, pp. 1586-1601 (2005)
17. Bagci, F.; Schick, H.; Petzold, J.; Trumler, W.; Ungerer, T.: Support of Reflective Mobile Agents in a Smart Office Environment. *Springer*, Vol. 3432, pp. 79-92 (2005)
18. Ejigu, D.; Scuturici, M.; Brunie, L.: Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing. *Journal of Computers*,Vol.3,pp.40-50 (2008)
19. Gong, L.: JXTA: A network Programming Environment. *IEEE Internet Computing*, Vol. 5, pp. 88-95 (2001)
20. Wilson, B.J.: JXTA. *New Riders* (2005)
21. Brookshier, D.; Govoni, D.; Krishnan, N.; Soto, J.C.: JXTA: Java™ P2P Programming. *Sams Publishing* (2002)
22. Viúdez, J.; Holgado, J.A.: Plataforma Java para el Desarrollo de Aplicaciones en Entornos Empotrados. *I Simposio en Desarrollo de Software*, Granada, pp. 147-162 (2007)
23. Peltz, C.: Web Services Orchestration and Choreography. *IEEE Computer Society*, Vol. 36, pp. 46-52 (2003)
24. McIntosh, R. L.: Open-source tools for distributed device control within a Service-Oriented Architecture. *Science Direct*, Vol.9, Iss.6, pp. 404-410 (2004)
25. Rodríguez, S. S., Serrano, M. D., Holgado, J. A.: Desarrollo de una Aplicación para el Control Domótico del Hogar sobre una Plataforma de Servicios Distribuida basada en JXTA. *To appear in the Second Symposium on Software Development*. Granada (2008)