2012 International Conference on Computer and Software Modeling (ICCSM 2012)
IPCSIT vol. 54 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V54.03

Network Architecture for Ambient Intelligence Environments and a New RMI Approach to Support Different Devices in Ruling Smart Homes

HosseinYarahmadi[1+] and Mohammad Kamankesh[2]

[1,2]Department of Computer Engineering, Broujerd Branch, Islamic Azad University

Abstract. In this paper, we are going to present a network architecture which is used in an Ambient Intelligence (AmI) environment. As the users of the smart homes need to connect to this network with their handheld devices like notebook and smartphones, they need a network architecture which could support different types of clients. The common way to implement remote method invocation is RMI library which has been published by Sun. but it is not able to support different types of client sockets. In this paper, we are going to present a new approach of RMI to support different types of client sockets. With using the new RMI approach, the users are able to control the ambient intelligent environment which they are living in and define new rules for the environment behaviors.

Keywords: Ambient Intelligence, Mobile Agents, Remote Method Invocaion.

1. Introduction

Ambient Intellignece is a type of viewing the world from a new viewport where the human race is surrounded by a intelligent devices which are seamlessly embeded in living environment of the user but they are invisible from the user sights [1, 3, 6, 7]. These environments are designed in a way which could support the users by performing everday activities of the user. For example consider today modern smart homes. There are countless devices in different parts of the home which are able to perceive the environment's status, talk to each other and do appropirate interactions toward the changes in the environment. In such environments, major challenges exsist which must be solved to provide user-ambient interaction without any interrupt in user activity. Considering a smart home again, an intuitive interaction system must allow the user for example to determine his living conditions like temprature, humidity and lighting, each of the rooms has ints own capabilites and infrastructure. Another challenge for the user is how to specify his living conditions. There must be different ways to specify living conditions for different users, with different knowledge about specific technical information of smart homes. Designing and implementing a smart home which different parts of it could interact with the user without any interruption and ability of controlling the living conditions of the home using different ways , need a good interaction and communication between devices of the home. The interaction and communication could be achieved by establishing a network in the home and using mobile agents, the challenges could be solved.

The main contributions of the paper is to present network architecture for smart environments and provide infrastructures that be able the users to use their handheld devices like notebooks or smartphones to control or rule the ambient intelligence environments which they are living in.

In this paper, we are going to present the network architecture of the smart home in chapter 2. Then present RMI in chapter 3. In 4th chapter, we will present why the RMI has some restriction to be used for our project and we will present a new approach for RMI to be implemented in our system. In chapter 5, we have

[+] *E-mail address*: hyarahmadi@iaub.ac.ir

described the evaluation method. In 6th chapter, we will present the results and in 7th chapter we will present a summarization of the paper.

2. Designing a Network Architecture Allowing Standard Access to Smart Home Devices and Rules

Such smart homes are composed of physical spaces (walls, windows, doors, etc.), embedded stationary and mobile devices (e.g. lights, TV or beamers) as well as user's personal appliances such as his notebook or smart phone. There several actors, e.g. lighting devices, visualization devices, and room condition equipment (temperature, lighting and etc.).

The smart environment (see Figure 1a) is a simulation for ubiquitous-computing environments. We are using this environment to test the intelligent-learning and adaptation mechanisms our embedded agent needs with the hopes of realizing the ambient-intelligence vision in ubiquitous computing environments. As an intelligent dormitory, the simulated smart environment is a multiuse space— that is, it contains areas for varied activities such as sleeping, working, and entertaining— that compares in function to other living or work spaces such as a one-room apartment, hotel room, or office. It contains the normal mix of furniture found in a study or bedroom, letting the user live comfortably. The furniture (most of which we fitted with embedded sensors) includes a bed, work desk, bedside cabinet, wardrobe, and PC-based work and multimedia entertainment system. The PC contains most office-type programs to support work as well as audio and video services for entertainment (to play music CDs, listen to radio stations using Dolby 5.1 surround sound, and watch television and DVDs).

To make the smart home as responsive as possible to its occupant's needs, we fitted it with an array of embedded sensors (such as temperature, occupancy, humidity, and light-level sensors) and effectors (such as door actuators, heaters, and blinds). It provides the user with a visualization tool showing the home's current state and enables direct control of the various effectors in the room.Although the smart home looks like any other home, the ceiling and walls hide numerous networked embedded devices residing on three different networks: Lonworks, 1-Wire, and IP. They provide the diverse infrastructure present in ubiquitous-computing environments and let us develop network independent solutions. Because we need to manage access to the devices, gateways between the different networks are critical components in such systems, combining appropriate granularity with security.

Fig. 1: The simulated smart home and the interface of the software which can be run on smart phones and notebooks to control and rule the smart home.

Lonworks, Echelon's proprietary network, includes a protocol for automating buildings. Many commercially available sensors and actuators exist for this system. The physical network installed in the simulated environment uses the standards of Lonworks TP/FP10 network, and virtual Echelon's iLON 1000 Web server provides the gateway to the IP network. This server lets us read and alter the states and values of sensors and actuators via a standard Web browser using HTML forms. Most of the sensors and effectors in the smart home are connected via a Lonworks network, virtually.

The 1-Wire network, developed by Dallas Semiconductor, was simulated to connect simple devices over short distances. It offers a range of commercial devices including small temperature sensors, weather stations, ID buttons, and switches. The 1-Wire network is connected to a Tiny Internet Interface board [8], which runs an embedded Web server serving the status of the networked devices using a Java servlet. The servlet collects data from the network devices and responds to HTTP requests.

The IP network forms a backbone to interconnect all the networks and other devices, such as the multimedia PC. This PC is the focus for work and entertainment in the smart home; it also uses the HTTP protocol to display its information as a Web page.

The smart home's gateway server is a practical implementation of a virtual HTTP server acting as a gateway to each of the room's sub networks. This shows that by using a hierarchy of gateways, it would be possible to create a scalable architecture across such heterogeneous networks in intelligent inhabited environments and ubiquitous computing environments. The smart home gateway server allows a standard interface to all the room's subnetworks by exchanging XML-formatted queries with all the principal computing components. This overcomes many of the practical problems of mixing networks. This gateway server lets the system operate over any standard network such as EIBus or Bluetooth. We could readily develop it to include plug-and-play, letting the system automatically discover and configure devices using intelligent Mechanisms. In addition, such a gateway is clearly an ideal point to implement security and data mining associated with the sub network. Figure 2 shows the logical network infrastructure in the smart home.

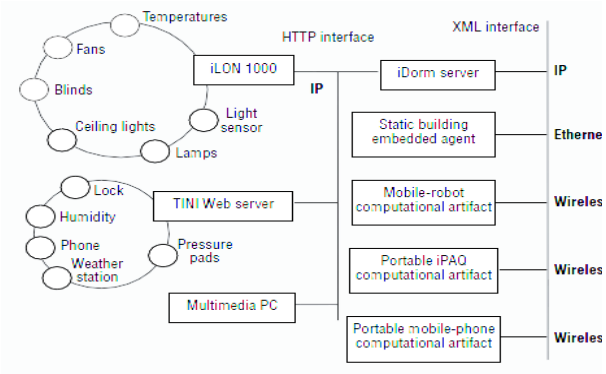

Fig. 2: Network Infrastructure of the Smart Home.

environments are designed in a way which could support the users by performing everday activities of the user. For example consider today modern smart homes. There are countless devices in different parts of the home which are able to perceive the environment's status, talk to each other and do appropirate interactions toward the changes in the environment. In such environments, major challenges exsist which must be solved to provide user-ambient interaction without any interrupt in user activity. Considering a smart home again, an intuitive interaction system must allow the user for example to determine his living conditions like temprature, humidity and lighting, each of the rooms has ints own capabilites and infrastructure. Another challenge for the user is how to specify his living conditions. There must be different ways to specify living conditions for different users, with different knowledge about specific technical information of smart homes. Designing and implementing a smart home which different parts of it could interact with the user without any interruption and ability of controlling the living conditions of the home using different ways , need a good interaction and communication between devices of the home. The interaction and communication could be achieved by establishing a network in the home and using mobile agents, the challenges could be solved.

The main contributions of the paper is to present network architecture for smart environments and provide infrastructures that be able the users to use their handheld devices like notebooks or smartphones to control or rule the ambient intelligence environments which they are living in.

In this paper, we are going to present the network architecture of the smart home in chapter 2. Then present RMI in chapter 3. In 4th chapter, we will present why the RMI has some restriction to be used for our project and we will present a new approach for RMI to be implemented in our system. In chapter 5, we have

described the evaluation method. In 6th chapter, we will present the results and in 7th chapter we will present a summarization of the paper.

3. Using RMI for System Remote Access

In our system, the users must be able to see what is happening in the smart home, what is the current status of the device, etc. Each user is able to access the status of the environment and each device individually using his/her notebook, computer or smartphone by connecting to the Central Controller Computer (CCC).

To implement this system sun recommends RMI library. RMI is a runtime library which be able the programmers distributed Java technology-based on Java technology applications, in which the methods of remote Java objects can be invoked from other JVM on different tools. When an object on a client tries to invoke a method of a remote object on a server, the object establish a communication with stub object and a skeleton object when an object on a client invokes a method of a remote objects on a server. The stub object on client's JVM is the corresponding object with the remote object. A stub object is client's proxy for remote objects, and its roles are to hide network connections and serialization of parameters. A stub object has an RMIClientSocketFactory object which creates a socket to communicate with a server. A skeleton object which is corresponding with a remote object is on server's JVM, and the skeleton object invokes methods of the remote object in effect. Roles of skeleton objects are to hide network connections and deserialization of parameters. Stub objects and skeleton objects are managed by an RMI runtime on each of the JVMs; moreover the objects are called automatically by each the RMI runtime if necessary. An RMI runtime has an RMIServerSocketFactory object which creates a server socket to wait for incoming calls from clients. RMIClientSocketFactory and RMIServerSocketFactory are provided by java's core library. Figure 3 shows a model of relations among an RMI runtime, a stub object and a skeleton object. A server executes the following steps to export a remote object so that an object on a client can invoke methods of the remote object on the server.

1. An RMIClientSocketFactory object and an RMIServerSocketFactory object are created.
2. A stub object and a skeleton object are generated using above objects and the port number which is used to wait for incoming calls from clients.
3. The stub object and the skeleton object are registered in server's RMI runtime.
4. The stub object is registered in server's RMI registry which allows remote objects on the server to register themselves as available to objects on the client.

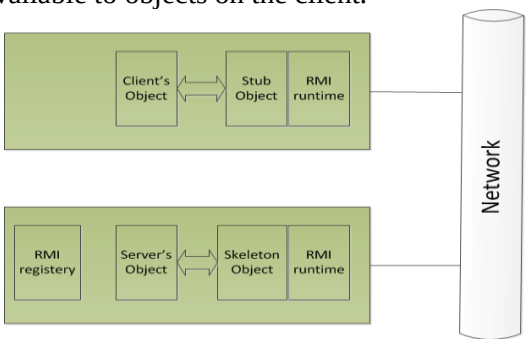

Fig. 3: a model of relation among an RMI runtime, a stub object and a skeleton object when an object on a client invokes a method of remote objects on a server.

When an object on a client tries to invoke a method of a remote object on a server, the object gets the stub object which is corresponding with the remote object from server's RMI registry. After that, the object invokes the method for the stub object. By client's RMI runtime, a socket is created by the RMIClientSocketFactory object which is contained in the stub object. Next, client's RMI runtime communicates with server's RMI runtime by the socket. Furthermore, server's RMI runtime creates a server socket by the RMIServerSocketFactory object which is registered in server's RMI runtime, after that the server socket communicates with the socket.

4. New RMI Implementation for the System

Ambient The behaviors of RMI which we require are that a client can use sockets of different types at the same time and a server can accept incoming call from the sockets on only one port. To realize this, these two problems must be considered:

1. An RMIClientSocketFactory object is created by a server, after that when a client tries to use the object, the object is managed by the RMI runtime on the client. Therefore, an object on the client cannot invoke methods of the RMIClientSocket-Factory object. Namely, the client cannot create sockets of different types by the RMIClientSocketFactory object.
2. A server exports using a pair of an RMIClientSocketFactory object, an RMIServerSocketFactory object and a port. If a server exports using pairs of multiple RMIClientSocketFactory objects and the same port, a client selects requiring RMIClientSocketFactory object

It is impossible to solve the first problem since RMI Specification does not define the manner to access an RMI runtime. Moreover, Sun's implementation of RMI does not provide the manner to solve the second problem.

In a system which uses Sun's implementation of RMI, when a server receives a request from a client, the server creates an object of ServerSocket class which is contained in java's core library by an RMIServer-SocketFactory object. Next, the ServerSocket object creates a server socket, after that the server socket waits for incoming calls. In order to create multiple server sockets which are corresponding with connecting sockets, steps of above execution must be changed as follows:

1. A server socket which is created by an object of ServerSocket class receives a kind of sockets from a client.
2. The other server socket which is corresponding with the kind of sockets is created, after that the server socket communicates with client's socket.

For the above, on the client side, a socket which is created by an RMIClientSocketFactory object is necessary to send the kind of sockets, so that MultiChannelClientSocketFactory class which is implemented RMIClientSocketFactory interface is developed. On the server side, in order to receive the kind of sockets, MultiChannelServerSocketFactory class which is implemented RMIServerSocketFactory interface is developed. Furthermore, MultiChannelServerSocket class is developed so that the server socket which is corresponding with the kind of sockets is created. Figure 4 shows a relation of above classes. The above classes are defined as MultiChannelSocketFactory.

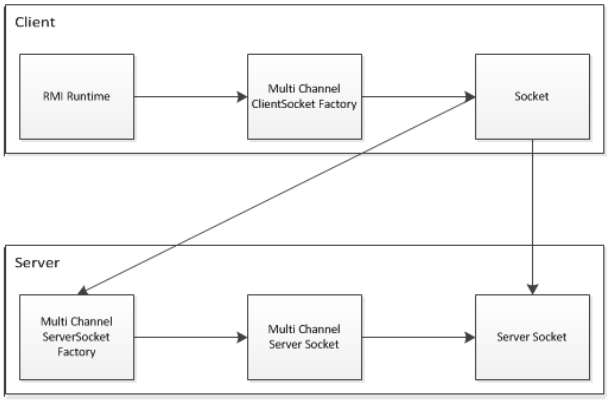

Fig. 4: a relation of classes which are contained in MultiChannelSocketFactory

5. Evaluation Method

The development of the platform was done following the User centred design approach. Following this approach, for the initial user test the platform was tested in our lab by 10 of our friends by 30 friends in distance. The tests included interaction with the initial prototypes, together with personal interviews and focus groups to capture the user's experience and to compile the input for the upcoming development period. For the final test of the platform, the system was tested by 16 our friends for a period of 5 hours. The platform's final prototype was also tested in a smart home which were designed and implemented for an elderly person. All of my friends which I have mentioned in this text were completely aware of Ambient Intelligence topic and smart homes and 2 of them had the experience of living in a smart home.

6. Results

The opinions about the system were much divided among those not having a clear statement, those thinking that it was a good application, and those reporting from the beginning. However, a majority of 58.2% thought it would be helpful in improving their social relationships, 41.4% thought it would help them to be more satisfied of smart homes, 63.7% thought it would help them to get closer with friends and 89.2% were confident in the idea that it would improve their quality of life.

Regarding the main menu interface, most of the sample (59.3%) found it pleasant or very pleasant, not being tiring for the eyes. All of them considered the interface was readable, with appropriate font size and color. The voice control demonstration worked well and participants were impressed.

7. Acknowledgment

This study was supported by Islamic Azad University, Broujerd Branch, Iran. The authors would like to acknowledge staffs of University.

8. References

[1] H. Yarahmadi, M. Dehghan; A Hybrid Model for protecting the Integrity of Mobile Agents; Computer Society of Iran, 2006, PP:1933-1938.

[2] Ali A. NazariShirehjini. A Generic UPnP Architecture for Ambient Intelligence Meeting Rooms .and a Control Point allowing for integrated 2D and 3D Interaction. Joint sOc-EUSAI[C], 2005.

[3] C. Giacomo; L. Leonardi; F. Zambonelli; Mobile-Agent Cooridnation Models for Internet Applications, IEEE, 2000. PP:82-89.

[4] E. Aarts; R. Harwing; M. Schuurmans; Ambient Intellignece. McGraw-Hill, Inc. 2002, PP: 235-250.

[5] Russel

[6] H. Hargras; V. Callaghan; M. Colley; Graham Clarke; A. Pounds; A. Cornish and H. Duman; Creating an Ambient-Intelligence Environment using Embedded Agents; IEEE Computer Society; 2004, PP:12 – 20.

[7] L. Encarnacao; T. Kirste. Toward smart appliance ensembles. True Visions, Virtual Information and Knowledge Environments, Springer, 2005.

[8] T. Kirste, Smart Environments and self-organizing appliance ensembles. Aarts E, Encarnacao J L (eds). True Vision, Springer, 2005.

[9] www.ibutton.com/TINI.

[10] A. Holmes, H. Duman; A. Pounds; A. Cornish; The iDorm: Gateway to Heterogeneous Networking Environment", Proceeding Int'l Test and Evaluation Association (ITEA) Workshop Virtual Home Environments, ITEA Press, 2002, PP: 30 -37.

[11] S. Motomura; T. Kawamura; K. Sugahara; New Implementation of RMI to Protect the Integrity and Confidentiality for Mobile Agents, Proceedings of the 3rd international conference on Web Information Systems and Technologies, PP: 364-370.

[12] A. Zwierko and Z. Kotulski, Integrity of Mobile Agents: A New Approrach, International Journal of Network Security, Vol. 4, No.2, 2007, Vol 4, No.2, PP: 201-211.