Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

Mobile Agent based Integrated Control Architecture for Home Automation System

Chao-Lin Wu, Wei-Chen Wang, and Li-Chen Fu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract—**The architecture of the conventional home automation system is usually centralized and thus causes many problems, e.g. lots of network traffic, heavy computation load, and poor fault tolerance. Another problem with the architecture is that there exists a variety of different home control networks incompatible with one another, which makes it difficult to integrate them in order to provide high level management functions in a home automation system. Besides the problems mentioned above, the most serious problem with a home automation system is its dynamically changing environment. Such frequently changing situation of the peripherals disallows straightforward configuration and maintenance of the home automation system. In this paper, we propose a mobile agent based integrated control architecture for home automation system, which is able to solve the problem mentioned above. This architecture is a distributed one, reducing the network traffic and computation load by delegating the management function to each control node. To integrate various kinds of home control network, we use computer network as the backbone and apply the concept of gateway to facilitate different networks to communicate with one another. Finally, we take advantages of the characteristics of mobile agent to cope with the problem of the dynamic environment and to enhance the fault tolerance mechanism.**

1 Introduction

A Home Automation System (HAS) is not a new research topic in the field of computer technology, but the conventional control architecture of HAS usually belongs to centralized type, which will face many different problems. For example, in order to maintain normal system operation, the centralized server has to periodically check the peripheral status and execute every detailed part of each management function, and this will cause heavy load in network traffic and server computation. When the fault tolerance issue is considered, the failure of the centralized server loading to malfunctioning of the HAS gives a major deficiency. Moreover, to set up a centralized HAS by connecting all the peripheral devices to a server is a necessary but seems to be an intensive effort task.

Another problem comes with the integration of the heterogeneous home control networks. The purpose of the home control networks, e.g. LonWorks[9], X10[10], BACnet[11], is to control conventional electronic appliances which originally are controlled only manually, thus providing integrated control for HAS to perform high level management. The problem is that these home control networks can hardly communicate among themselves. On the other hand, due to the advent of computer technology, recently many home appliance manufacturers emphasize on the development of Information Appliance (IA), and these IAs can communicate with one another through some network based on certain high level protocols, e.g. UPnP[2][3], Jini[4] and HAVi[5]. By taking good use of these protocols, HAS can directly control those IAs, but still these high level protocols are lacking compatibility. Therefore, to achieve a full-scale collaboration and coordination among heterogeneous appliances, it is necessary to integrate different control protocols including those for conventional home control networks and the high level protocols for IAs. However, to attain the goal, the system complexity drastically rises, and the integration and configuration of the HAS naturally become not trivial.

Besides these issues mentioned above, the most critical problem for HAS is the dynamically varying home environment. It is quite common for the inhabitants to carry some devices and to move around in the home, and many of the devices are flexibly connected to HAS in the sense that they can be turned on or off to serve various purposes, such as energy conservation or to conform to service disruption. As a result, the physical states of these devices vary frequently because they are disconnected from or reconnected to the network on and off. This dynamically changing configuration problem makes it difficult to set up the HAS and maintain it.

To solve the above problems, we propose a mobile agent based integrated architecture for HAS. This architecture reduces the load of network and computation in the conventional centralized HAS, and has a good mechanism for fault tolerance. It can integrate a variety of home control networks, and can deal with the dynamic home environment. The rest of the paper is organized as follows. Section 2 presents the system architecture and the design issues, and Section 3 gives the system overview. Further detailed discussion on the system is provided in Section 4, whereas Section 5 suggests the way of system evaluation. Conclusions are made and some future works are discussed in Section 6.

2 System Architecture and Design Issues

This section first describes Mobile Agent and Agent Host [6][12], which is the core technology of our architecture, and then shows the mechanism to integrate various kinds of home control networks. Finally, some additional system components and auxiliary design issues are introduced for enhancement of the HAS.

2.1 Mobile Agent and Agent Host

Agent Host is a kind of software running on OS, and plays as the platform for Mobile Agent to reside. Figure 1 shows the architecture of Mobile Agent and Agent Host. The main function of the Agent Host is to perform Agent Operations as listed below:

- create a new agent,
- suspend a running agent,
- resume a suspended agent,
- dispatch a suspended agent,

0-7803-8463-6/04/$20.00 ©2004 IEEE

- accept a suspended agent from other Agent Host

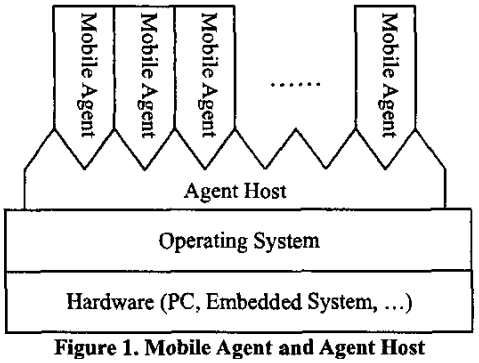

Figure 1. Mobile Agent and Agent Host

The Mobile Agents residing on the Agent Host together determine the actual functions of the Agent Host, and there are several kinds of Mobile Agent, which are shown below.

- Interface Agent
  This kind of agent will be responsible for interaction between the user and the system. The status of the HAS is, after being collected by this agent, presented to the user, who will then manage the HAS according to such collected information. Next, the Interface Agent will dispatch other agents relevant to the management requests from the user to perform the management tasks.
- Device Control Agent
  This kind of agent is the only one which is responsible for direct communication with all the devices.. Its tasks are to control the aforementioned devices, to monitor their status, and to cooperate with other agents, such as to receive their control commands and/or to report to them the device status. There are many kinds of Device Control Agent in the HAS, according to the protocol used to communicate with the controlled devices.
- Message Agent
  When some agent wants to communicate with some remote agent, Message Agent will be created and dispatched. The task of Message Agent is to carry the messages to the specified target, which will be some agent on some Agent Host.
- Function Agent
  This kind of agent will perform high level control management. Each one is responsible for some specific function, including User Preference, Lighting Control, and Entrance Guarding in our research environment.

2.2 Heterogeneous Network Integration

According to the type of connected appliances, control protocol can be categorized into two kinds. One is the home control network for conventional electronic appliances, and the other is the high level protocols for IAs. For each control protocol in either category, the HAS needs a control node capable of manipulating it. This node will be connected to the devices located in its responsible area, and function as protocol gateway [1] after being connected to the backbone of the HAS, which is the computer network.

The reason to choose computer network as the backbone is that the computer network is more and more popular in the home environment, and is also a good means for data communication. In addition, with adoption of computer network, all sorts of computer can be integrated into the HAS easily, so that the HAS will be endowed with powerful computing capabilities.

Basically, protocol gateway is an Agent Host running Device Control Agent(s). Its actual function depends on the capability of Device Control Agent run by it, and the implementation of Device Control Agent will have to be done case by case due to the different characteristics of different control protocols [13]. The structure of protocol gateway is shown in the Figure 2.

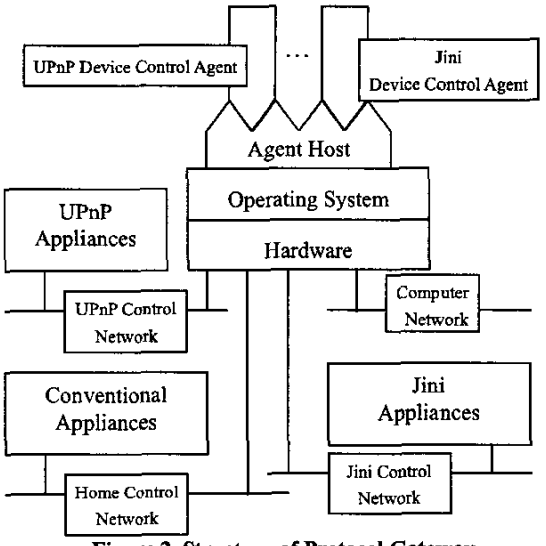

Figure 2. Structure of Protocol Gateway

2.3 Agent Database

Agent Database is an Agent Host with database, whose structure is shown in Figure 3.

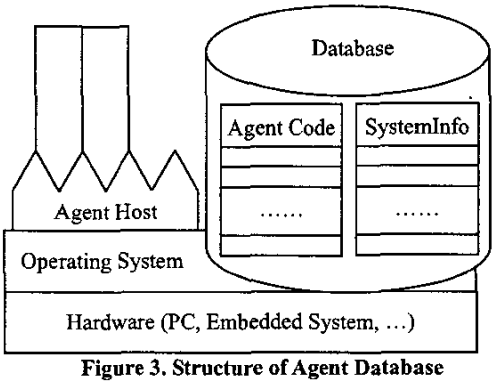

Figure 3. Structure of Agent Database

The database stores the codes of many different kinds of Agent, which can be dynamically downloaded and executed by Agent Host to augment its existing function. The related details will be discussed in section 4.4. Another purpose of this database is to store the system information, i.e., the status of each Agent Host in the HAS, which will be useful to

achieve "Fault Tolerance". More detailed discussion on that will be given in section 4.5.

2.4 Agent Communication Mechanism

The main purpose of this mechanism is to enable Agent Hosts in our HAS to communicate with one another. The agent communication can be separated into two parts. The first one is the advertisement message, which records the capability of the Agent Host. The second one is the agent language, which allows more complex forms of interaction than query/response mechanism.

- Advertisement Message

  The first step for an Agent Host to join the network is to implement its description, expressed in XML, concerning its capability, the description of the Agents run on it, and the related physical devices. Here, we use the UPnP device description as our example of advertisement message. The main purpose of the advertisement message is to facilitate each Agent Host to find other Agent Hosts, to learn about their capabilities, and/or to interact with them. The second step is to broadcast the advertisement message to the home network.

By using the UPnP protocol, it is easy for the Agent Host to be dynamically connected to the network and to describe itself. But to achieve the agent communication, it is not enough to use the UPnP protocol only. The exchanging message used in the agent communication is very complex, and it needs to be handled by a higher level language. So, we use a standard agent communication language in our HAS.

- Agent Language

  We use KQML (Knowledge Query and Manipulation Language) [7] as our agent's communication language. KQML, a language and a protocol, offers the intelligent agents the ability to exchange information and knowledge among themselves. It is a part of a larger effort, the ARPA Knowledge Sharing Effort [8], which aims at developing techniques and methodology for building large-scale sharable and reusable knowledge bases. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents. KQML can be used as a language for an application program to interact with an intelligent system or for two or more intelligent systems to share knowledge in support of cooperative problem solving.

With the mechanism composed of these two languages mentioned above, we can achieve Agent Migration, and the example for Agent Migration is given in section 4.2.

3 System Overview

In this section, we will describe how our architecture works, and the detailed system mechanisms, including setup, configuration, and management.

3.1 System Setup

Before HAS had been established, there are only many IAs and conventional electronic appliances independently disposed in the home environment. Although the IAs can autonomously form different groups according to their own high level control protocols, these groups are still independent from each other. Thus, the first step to set up the HAS is to connect all the appliances together.

To connect those conventional electronic appliances to be in charge by the HAS, some home control network(s) need to be used. The selection of home control networks should depend on the availability of its connectivity function and the inhabitants' preference, e.g. the construction price, or the convenience to build this network. Generally speaking, one control node can still be taken to connect all the appliances through only one kind of home control network, but this may make the work of constructing the networks costly. Here in this research work, we instead assume that people prefer to set up home environment separately from one living space to another, and may even allow multiple kinds of home control network in one living space to lower the cost of construction or just for convenience. Therefore, more than one control node set by our architecture should have the ability to communicate with the computer networks, and they can function as protocol gateways. As for the network formed by IAs, a control node which functions as the protocol gateway should also be set for each network in each living space, and this control node should be able to communicate with both the IAs and the backbone computer network. Of course, a control node can even function as a protocol gateway for multiple control protocols if its supporting computing hardware has sufficient capability.

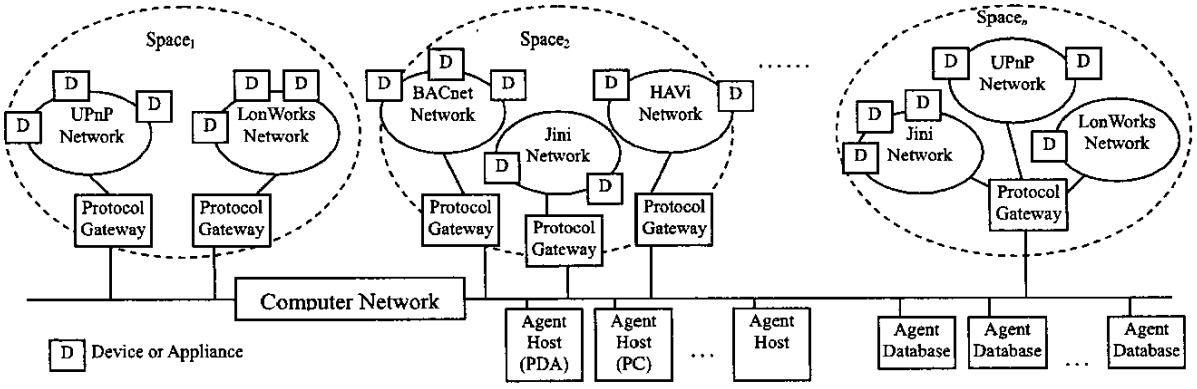

Figure4. Overview of the Home Automation System Architecture

Through these control nodes which function as protocol gateways, all these appliances in charge by the HAS are connected together, and in turn the heterogeneous network integration is achieved. On the other hand, all these control nodes can function as Agent Hosts, each of which is capable of running Device Control Agent to control devices, or running Function Agent to perform high level management. In addition to these necessary control nodes as mentioned above, there are some other nodes in the system. Some of them play the roles as Agent Databases, and some of them are there to provide additional computing resource, but they can all perform as Agent Hosts.

As a result, there are multiple control nodes manipulating the same kind of control protocol while being located in different living space, but there are also multiple control nodes talking different control protocols while being located in the same living space. Besides, there are other nodes which function as Agent Databases or additional computing resources, and all these nodes can perform as Agent Hosts as well, all of them are connected through computer network. The situation is as depicted in Figure 4.

These nodes mentioned above can be implemented with some embedded system if its hardware has enough ability to function as Agent Host, but for the convenience of experiment, we implement these nodes with personal computer.

3.2 System Configuration

After the system hardware is set up according to the previous section, Agent Database will broadcast messages requesting each Agent Host in the system to report its status. Then, the Agent Host in each control node will dispatch the Message Agent, carrying integrated information about control node, to the foregoing Agent Database, and the system information carried will be stored into the Agent Database for further use. The system information includes system status retrieving performed by Interface Agent, future information update by Agent Host in each control node. As for adding new Agent Host, discussion will be left to Section 4.4.

When trying to manage the HAS, user can invoke the Interface Agent on an Agent Host, and that Interface Agent will then access the Agent Database to retrieve the system status. If such information is not available because Agent Database fails, Interface Agent will also broadcast messages requesting each control node to report its status with Message Agent. In either situation, Interface Agent will retrieve enough system information and show them to the user. After knowing the current system status, user can group or categorize the devices according to his preference, e.g. the spaces he lives or the functions he often uses. The grouping preference for each user will be stored in the Interface Agent individually, and hence user can continue his work with his familiar way next time when resuming the Interface Agent.

3.3 System Management

When managing the system, user can delegate Function Agent, whose code is stored in the Agent Database, to Agent Host in each control node to communicate with the Device Control Agent to fulfill the management purpose. For example, when users want to keep some room with enough brightness, Interface Agent will create a Message Agent, which will be dispatched to the Agent Database to create a Function Agent, and then this Function Agent will migrate to Agent Hosts in the control nodes belonging to the specified room. This Function Agent will migrate among these Agent Hosts to communicate with Device Control Agent locally. They will check the status of brightness, and enhance the brightness by turning on more lamps if necessary and possible. This Function Agent will stay at Agent Host in the control node to keep cooperating with the Device Control Agent, and dispatch Message Agent to report back the status, which is about if the task is completed. The flow chart is given in Figure 5.

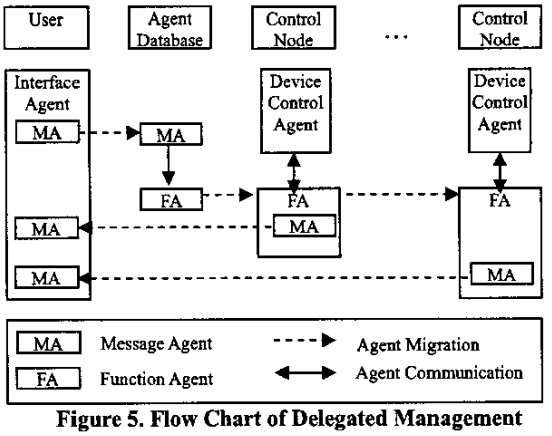

Figure 5. Flow Chart of Delegated Management

These reports carried by the Message Agent are sent to Agent Database asynchronously, which means that system status will be updated only when some events are triggered or device status are changed. However, one possible situation when no report occurs is that the system malfunctions. In case this happens, users can also check the system status actively by sending Message Agents out when the HAS not reporting events at all. Message Agents will be dispatched to Agent Host in each control node assigned by Interface Agent, and when arriving, Message Agents will communicate with Device Control Agent on the control node locally to gather information of each control network. These Message Agents will move around the system to collect more information if necessary, or return back to the Agent Host where they are dispatched. These gathered system status will be reported to Interface Agent, and Interface Agent will show these information to the user. The flow chart is given in Figure 6.

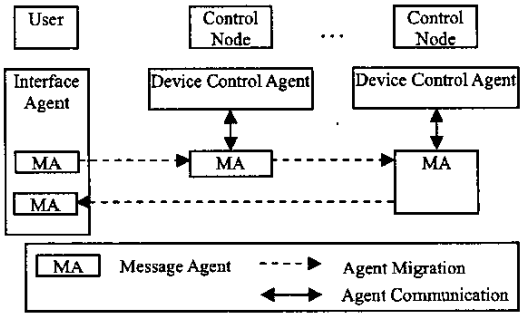

Figure 6. Flow Chart of Active System Check

4 Discussions

In this section, we will discuss the advantages of our architecture and how the problems, which are originally caused by the conventional HAS, are solved.

4.1 Reducing Load of Network and Computation

In the conventional HAS, all the functions are performed by the centralized server. But in the architecture based on Mobile Agent, the management is achieved by Function Agent delegate to Agent Host in each control nodes by the user. These Function Agents communicate with the Device Control Agent locally, and thus reduce the network traffic when message exchange happens frequently between Function Agent and Device Control Agent. This also reduces the computation load by distributing management tasks to Agent Host in each control node. The reaction capability of system is also enhanced by communicating and managing locally.

Another point is that the system status is asynchronously reported from distributed control node to Agent Database, so the system doesn't have to perform polling to retrieve information. Without polling, the system can avoid unnecessary detecting with events rarely happen, and also avoid slow reactions when dealing with events happening frequently or randomly. This method will help reducing the network traffic because message transmission happens only when necessary.

4.2 Advanced Load Sharing through Agent Migration

Another advantage achieved by Mobile Agent is the advanced load sharing. When some management function performed by Function Agent needs a great deal of computation resources, the computation task can be delegated to some other Agent Host. Function Agent will check available Agent Hosts from system information stored in Agent Database, and negotiate with one of the available Agent Hosts. The negotiation is done by KQML mentioned in Section 2.4. If this chosen Agent Host needs to reserve its computation resource at that time, e.g. playing MPEG4 files or running a critical program, it can deny this migration. However, if this chosen Agent Host is willing to accept this task, Function Agent will create another Function Agent and dispatch it to this willing Agent Host with enough information for the computation task. This new Function Agent will compute on the chosen Agent Host, and dispatch back Message Agent carrying finished computation result. The flow chart is given in Figure 7.

4.3 User Mobility

In the conventional HAS, users are fixed to the server location when managing the system. But thanks for the mechanism provided by the mobile agent, as long as the nodes capable of functioning as Agent Host, users can perform management function at any node. People can download and execute the Interface Agent at their desired node with their previous preference, then perform the management as the previous section mentioned. In this way, user can roam around the home environment and access the system any time through nodes with the ability to interact with people, e.g. a PC or personal digital device capable of communicate with the computer network.

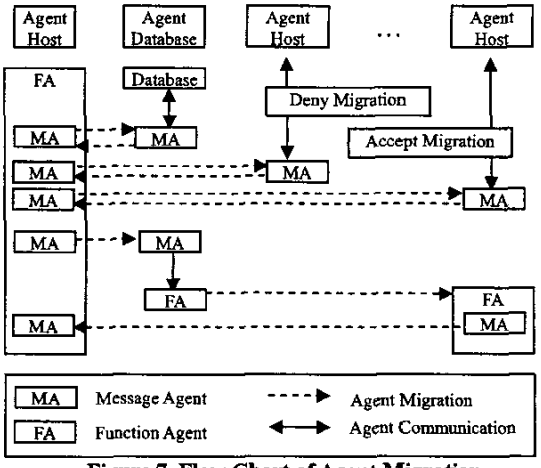

Figure 7. Flow Chart of Agent Migration

4.4 Dynamic Resource Added/Removed

In the HAS, some resources are dynamic. These resources are devices which will be occasionally disconnected from HAS and reconnected to HAS at some other time or at some other place. With the capability of mobile agent, these dynamic resources can be added or removed freely without bringing negative effects to the HAS.

One of the example is a laptop enabling its Agent Host to join the HAS to provide its computation resources as additional computation power. This laptop will use UPnP mentioned in Section 2.4 to advertise its existence to Agent Database, so other Agent Hosts requiring computation support can discover this laptop through querying Agent Database, and negotiate with this laptop to share computation load. Next, considering the situation the laptop mentioned above is removed from HAS when user carries it out. In this case, Function Agent performing computation task in this laptop can suspend its task, migrate to another Agent Host, and then resume its computation task. Without mobile agent, this laptop can not be removed from HAS before the computation task is finished, or the unfinished computing result will be lost.

Another example is the portable computer peripheral or portable IAs. In the conventional HAS without installing the function to communicate with Jini in advance, the HAS can not control even can not discover this new Jini appliance when a Jini appliance migrate to a room where the control node does not understand Jini protocol. However, in our architecture, control node can be enhanced by downloading Device Control Agent capable of speaking Jini protocol from Agent Database, and thus this Jini device can be controlled by the HAS. The advantage is that this Device Control Agent can be dynamically installed on the Agent Host when necessary, and also can be removed from Agent Host when the need is no longer existed, thus relieving the storage space of the control node, which is an advantage when Agent Host running on the embedded system..

4.5 Fault Tolerance

In addition to providing dynamic agent support, Agent Database also plays an important role in Fault Tolerance mechanism. As long as the agent functions normally, it will asynchronously update the system status and its own status.

Once some agent malfunctions due to hardware error, when hardware is repaired, agent can recover its previous state according to the information stored in the Agent Database. Moreover, when the task performed by the malfunction agent is not hardware-dependent, e.g. computation tasks, the computation result previously stored in the Agent Database can be used and this computation task can be resumed by other agent running on another Agent Host without waiting for the hardware repairs. If the malfunction is not related to the Agent Host, then a new agent will be created to resume the unfinished task, running on the original Agent Host. The agent status stored in the Agent Database is defined by each agent itself according to its requirement.

All the system status, Agent Status, and Agent Code will be lost if the Agent Database malfunctions. However, the HAS can still function normally only without status storing and dynamic new function support. This drawback can be reduced by some database techniques, e.g. arranging multiple Agent Databases in the system. Comparing to conventional HAS, which has to back up the entire system in order to achieve the same fault tolerance performance, the cost of multiple databases is much reduced since Agent Database stores data only.

5 System Evaluation

In this section, we will illustrate the system evaluation result of our architecture. This experiment focuses on the response time when agent migration happens under different network traffic situation.

In order to evaluate the response time of the agent migration, we develop a small program that can continuously broadcast the dummy messages to the home network every a period of time. These dummy messages represent the traffic load of messages (message/second). Then we produce some virtual messages to request the agent migration. The response time is calculated from the time when the Agent Host receives the agent migration request to the time when the migration is finished. The result of this experiment is shown below.

Traffic Load / Agent Function	1000	1100	1200	1300	1400	Average (ms)
User Preference	0.352	0.395	0.428	0.504	0.529	0.441
Lighting Control	0.133	0.161	0.166	0.174	0.177	0.162
Entrance Guarding	0.222	0.241	0.307	0.334	0.452	0.311
Video Conference	0.699	0.721	0.832	0.877	0.920	0.809

We simulate five kinds of the traffic load conditions and calculate the average response time of each condition. The average response time of agent migration is all within 1 millisecond, even when there are many other messages sent simultaneously in the frequency of over 1000 messages per second. According to the experiment result, the mechanism of agent migration is efficient enough for our home environment.

6 Conclusion and Future Work

In this paper, we propose the Mobile Agent based Integrated Control Architecture for HAS. This architecture integrates heterogeneous network through the concept of gateway. And based on Mobile Agent, this architecture not only solves many problems caused by conventional HAS, but also creates many advantages, e.g. Reducing Load of Network and Computation, Advanced Load Sharing, User Mobility, Dynamic Resource, Fault Tolerance. We implement a prototype HAS based on this architecture, and it achieves good performance.

When considering the agent migration in load sharing, although our architecture can perform it, in order to support Agent Host making the decision to start/accept/deny agent migration, we still need the mechanism to measure the status of Agent Host. The advertisement language we currently use for Agent is a prototype based on UPnP, and the detailed definition will be one of our future topics. On the other hand, Jini may achieve the same performance as UPnP, and we will study it. Although in our architecture, Agent Host can dynamically download agent from Agent Database to enhance its capability, the authorization and authentication of this operation is still an issue. Finally, we will focus on developing the advanced function of HAS, providing human better life with more automated control performed by agent with more intelligence based on the user preference

Acknowledgement

This research is sponsored by Computer & Communication Research Labs in Industrial Technology Research Institute under the project T2-93006-5.

References

[1] "A virtual overlay network for integrating home appliances", Nakajima, T.; Satoh, I.; Aizu, H.; Applications and the Internet, 2002. (SAINT 2002). Proceedings. 2002 Symposium on , 28 Jan.-1 Feb. 2002, Pages:246 – 253

[2] UPnP Forum, http://www.upnp.org/

[3] B.A. Miller, T. Nixon, C. Tai, M.D. Wood, "Home networking with Universal Plug and Play," IEEE Communications Magazine, Volume: 39, Issue: 12, Page(s): 104 -109, December 2001.

[4] Jini CommunitySM , http://www.jini.org/

[5] R.Lea, S.Gibbs, A.Dara-Abrams, E.Eytchson, "Networking Home Entertainment Devices with HAVi," IEEE Computer, Volume: 33, No.9, 2000.

[6] S. Papavassiliou, A. Puliafito, O. Tomarchio, Jian Ye, "Mobile agent-based approach for efficient network management and resource allocation: framework and applications", IEEE Journal on Selected Areas in Communications, Vol. 20, Issue: 4, P.858 -872, May 2002.

[7] T. Finin, R. Fritzson, D. McKay, R. McEntire, "KQML as an Agent Communication Language," in Proceedings of the 3rd International Conference on Information and Knowledge Management, pp. 456-463, 1994.

[8] R. Patil, R. Fikes, P. Patel-Schneider, D. McKay, T. Finin, T. Gruber, R. Neches, "The DARPA knowledge sharing effort: Progress report," in Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference, November 1992.

[9] Echelon Corporation, http://www.echelon.com/

[10] X10.com, http://www.x10.com/

[11] BACnet, http://www.bacnet.org/

[12] A. Puliafito, O. Tomarchio, and L. Vita. "MAP: Design and Implementation of a Mobile Agents Platform", Journal of System Architecture, 46(2):145–162, January 2000.

[13] "Jini meets UPnP: an architecture for Jini/UPnP interoperability", Allard, J.; Chinta, V.; Gundala, S.; Richard, G.G., III; Applications and the Internet, 2003. Proceedings. 2003 Symposium on , 27-31 Jan. 2003, Pages:268 – 275