OFFLINE SERVICE DISCOVERY IN HUMAN, ROBOT, ENVIRONMENT INTERACTION

Dong To Nguyen, Sang-Rok Oh, Bum-Jae You

Intelligent Robotics Research Center - Korea Institute of Science and Technology

Abstract: In this paper, an offline service discovery method is proposed with an interaction framework of humans, robots, and environments. It creates a dynamic robot control system with flexible and uncomplicated configuration. A number of experiments have been conducted successfully. The experiments show that this framework works well and provides some advantages to existing systems. *Copyright © 2005 IFAC.*

Keywords: Mobile Robots, Agents, Interaction, Interfaces, Intelligent control.

1. INTRODUCTION

Nowadays, besides direct human robot interface (HRI), Internet HRI, that explores Internet to communicate, has become important. The Internet HRI can be implemented by using a client-server model, a distributed model or using agent technology. The Internet HRI using agent technology is mentioned in some publications recently. Hideaki Hideaki et al (1997) introduced an agent-based architecture where users communicate with robot via a video camera located outside of the robot. They focused on a ubiquitous access method that explores the agent as a means to communicate human and robots via the distributed sensors. Maxim Makatchev and Tso (2000) proposed a framework using a proxy agent architecture. They focused on solving the communication problems by using the proxy agent as a mediator between robots and users. Similar to the above researches, our research explores the agents for the HRI using distributed sensors and multi-agent communication method. Different from these, our research targets the service in offline situation.

In our Internet HRI approach, the world is considered as a set of workspaces. Each workspace is controlled by a multi-agent system. Firstly, this multi-agent system is used for fusing distributed sensors to monitor the environment. Secondly, it works as the center to dynamically coordinate with robots and users in the environment. When robots enter a workspace, they register their services, explore the information of this workspace and provide their ability to the system and users. Users use their Personal Digital Equipment (PDA) or notebook to access the system, discover and access the services provided by robots in the workspace. This system is able to organize, discover and explore the services of each partner.

Service discovery methods are very important in agent technology. In the agent-based standard such as Foundation for Intelligent Physical Agents (FIFA 2000), service discovery in an agent platform (AP) is provided by a special agent, which named as Directory Facilitator (DF). Agent can register, deregister and modify service information in DF. DF provides the service management mechanism to design an Internet HRI interface system. However, DF does not provide the ability to interact with the offline service. The presence of a robot in the system is not permanent. The robot may be offline at the time when the users want to assign the tasks, but it may be available later. Therefore, a mechanism to discover the services, assign the tasks to the robots, and provide the interface for service interaction when robotic agents are offline, is needed. Our framework supports a method to deal with the offline situation.

The responsive environment is the environment that can provide information about itself to the other partners working in it. Turning an environment into a responsive environment by installing sensors around the environment is one approach that is used in some human tracking systems, where a number of cameras are installed and connected together. Huang and Trivedi (2003) proposed a system, where the camera array is installed around the workspace for human movement tracking. In the implementation described later, environment sensor information and local sensor information are combined by exploring a multi-agent architecture. Developed from the idea of exploring a central control system to control a group of robots by changing the behaviour of the whole group that proposed by Ali and Arkin (2000), our responsive environment can change the behaviour of any robot placed in its environment to make this robot adaptive to the environment.

2. PROPOSED FRAMEWORK

2.1 Overall Framework

In the Internet HRI, one problem is the offline situation, when a robot is temporally out of control due to recharging of battery or suffering from network problems. Our framework is designed to solve this problem. Figure 1 describes the system with robotic, user and environment agents.

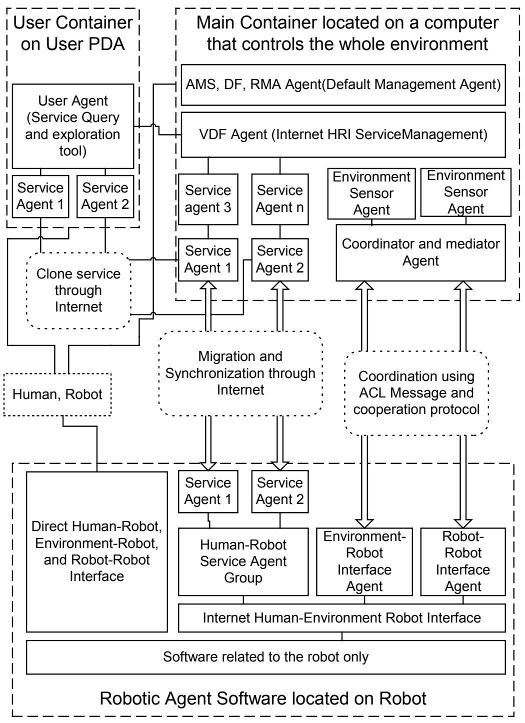

User Container on User PDA
Main Container located on a computer that controls the whole environment
User Agent (Service Query and exploration tool)
AMS, DF, RMA Agent(Default Management Agent)
VDF Agent (Internet HRI ServiceManagement)
Service Agent 1
Service Agent 2
Service agent 3
Service Agent n
Environment Sensor Agent
Environment Sensor Agent
Service Agent 1
Service Agent 2
Coordinator and mediator Agent
Clone service through Internet
Migration and Synchronization through Internet
Coordination using ACL Message and cooperation protocol
Human, Robot
Direct Human-Robot, Environment-Robot, and Robot-Robot Interface
Service Agent 1
Service Agent 2
Human-Robot Service Agent Group
Environment-Robot Interface Agent
Robot-Robot Interface Agent
Internet Human-Environment Robot Interface
Software related to the robot only
Robotic Agent Software located on Robot

Figure 1. The overall framework of an Internet-based HRI system

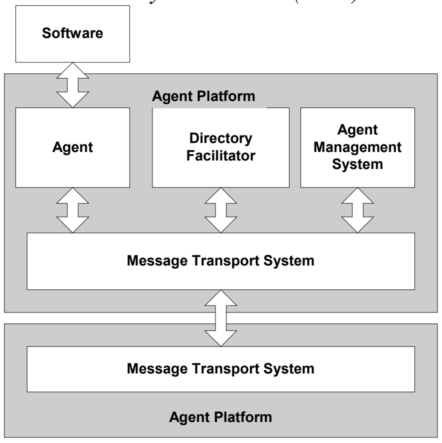
Robotic agents: The software is designed to separate two parts: the control part and the interface part. In the interface part, all behaviour and services are arranged as agents. In this paper, only the Internet-based services are considered. The internet-based services can be categorized into some types and with each type the system processes the service in different ways.

- The Internet human-robot interface services are services that require human interaction. With these types of services, robots create the slave agents for the services and migrate them to the main container. These services require no predefined algorithm and configuration. At the main container, user can manage, send command and work with the services.
- The robot and environment interface services are services that require coordination with each other or with the distributed sensors. These services use the predefined coordination agents and algorithms to connect these agents together in a special protocol. The coordination protocol should take into account the possibility of communication problem.

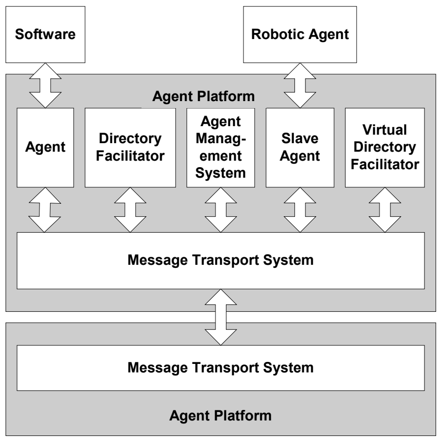
Responsive Environment: The responsive environment is controlled by a main container and environment agents so that it can respond to the change in the environment. It gathers information from distributed sensors and works as the management part for the services of each robot in the environment. It also provides the interface of the services on request. The coordinator agent should be located on the main container to coordinate robots and distributed sensors.

User Agents: Users access the system via a User Agent (UA) located on their PDA. On the PDA, there is no information about the robots. UA will query and clone the available services from the main container. With this UA, users can also interact with the services via Agent Communication Language (ACL) communication.

2.2 Virtual Directory Facilitator (VDF)



a. FIPA Agent Platform



b. FIPA Agent Platform with VDF extension

Figure 2. Management architecture of FIPA standard and VDF extension

The FIPA standard is selected to implement the system. The central part in the FIPA standard is the definition of a management standard using a central control in the main container. In our framework, the implementation part and the interface part of a service are separated so that the interface part can be moved around the network by migrating and cloning. Each robot appears in the system as an agent. Each time this agent registers to the AP, it will create its service slave agents. Using the mobility feature of the AP, slave agents will migrate to locate on the main container. These agents work as slaves for the robotic agents. They are created at the time the robotic agents appear in the system and remains for a particular period of time that specifies by the robotic agents. The slave agents collect information when the robotic agents are offline, get the requirement or interact with users and other robotic agents. It can provide complicated graphic interface. Therefore, users with no experience about the robot can interact with it because after being migrated to the user device, it can be used as a normal application program. Then the slave agents synchronize with the robotic agent when it is online. Each robotic agent can visit many workspaces and drop slave agents there. The existence of this slave agent is temporal. To control such a slave agent group, a special agent in the agent platform is created. This is called a VDF agent. VDF works similarly to DF in FIPA standard but all these services are provided by slave agents. VDF is responsible for synchronizing the slave agent and robotic agent. The UA located on the user's PDA works as a service query-and-display tool. UA can query all the services currently available on a system and then select the service that the user wants to use. After a service is selected, VDF will clone a copy of the service interface, move it to the container in PDA. After users interact with the interface, all data will be synchronized among interface agent.

This system helps each robot to broadcast its service even after it deregisters from the system temporally. Users save the time for interaction with services because they can access the system at any time, work with the interface they find in the VDF and explore this service by invoking the interface. The availability of the system increases since the demand of long term or permanent Internet connection is reduced and each robot only needs to be connected for a short time to migrate its service interface or to synchronize the data.

2.3 Robot-Environment interaction

A mobile robot normally uses only its local sensors. It is hard for robotic agents to work and cooperate in the unknown environment because it does not know the change in the environment beyond its sensors' range and the existence of the other robotic agents. With a multi-agent system, our environment becomes a responsive environment. This multi-agent system helps to overcome these problems by using coordinator agent to coordinate distributed sensors with robot agents. By installing sensors around the workspace and connecting these sensors to the multi-agent platform, the information about environment can be provided. Using this information, the responsive environment can discover the position of each robot and control the behaviours of a robot.

3. EXPERIMENTS AND RESULTS

3.1 Experiment design:

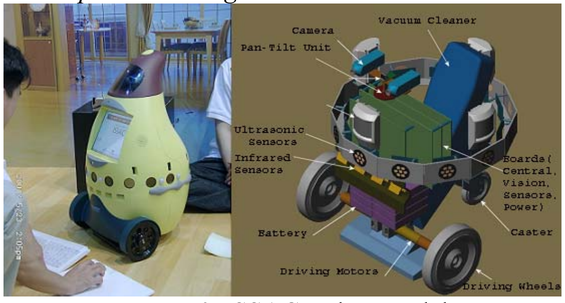
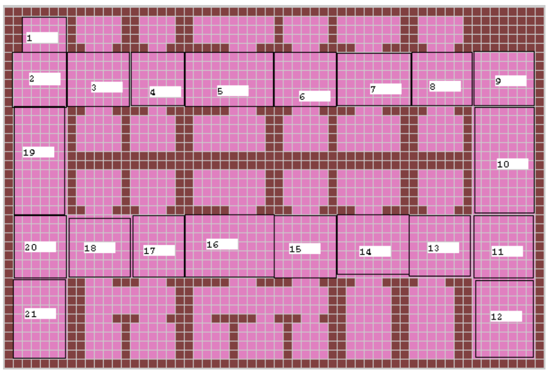


Figure 3. ISSAC Robot Model

Human-Robot Interface Experiment: The first experiment validates the human-robot interface using slave agents, VDF and user agents. To do this experiment, two service interfaces on ISSAC, cleaning service and behavioural customisation service, are selected. The input for the cleaning service is time and zone to be cleaned. When the conventional control system is used, ISSAC should have a permanent wireless connection to maintain the high available level. Because it is difficult to satisfy this, the service shows the useful point of the multi-agent system in sense of increasing the availability of the system. The behavioural customisation service illustrates the other application of the framework. By assigning option to a group of slave agents in VDF, the behaviour of all the robots with the same types can be changed. If there are some ISSAC robots in the same environment the behaviour of the whole group can be changed by using only one service interface. When the option is changed on the main container, every robots placed in this environment will change its behaviour and the robot becomes an environment awareness robot. Whenever ISSAC reaches the system again, its behaviour will change accordingly. As the framework is not only for ISSAC but also for any robot, a common offline control system such as a store and forward type system could not create the interface of the service because the features and behaviour lists of each robot are different. Thus, the service interface should be designed on the ISSAC and then migrate to the AP later. The experiment is conducted with the UA on a PDA, iPAQ 5450, using JadeLeap (Bellifemine, et al., 2003). This iPAQ connects to the network via an 11 Mbps wireless connection. The UA is written as an agent in an agent container on JadeLeap platform. In contrast to the conventional robotic control system, this UA does not have any particular information about robot or environment. After registering to the AP, the UA can search for the existing VDF, query the VDF on available services. VDF contains the services of robotic agents that are currently online and offline. After getting the list of services available in the system, UA may send query VDF to request a special service such as cleaning service. This service interface is cloned from the main container and migrated to locate on the user container. Then users run the interface on the user PDA, work with the service interface and leave the system.



Figures 4. 21 regions in the environment

Environment-Robot Interface Experiment: The second experiment validates the coordination with the distributed sensors using coordinator agent. ISSAC should go to some target points assigned dynamically by users via Internet HRI. ISSAC should plan to go to the list of targets by using map data and environment information. During the time ISSAC performs the task, the plan may need to be changed because users add or remove the task, or environment is changed. To implement the task, our multi-agent framework is explored. There are three main types of agents involved in the coordination: Robotic agent (RA), environment-agents (EA), Environment Monitor Agent (EMA). Along the corridors, a group of EA is installed. These EAs work as the external sensors for RAs. Cameras are used as the distributed sensors. It is difficult for environment sensors to locate exactly the position and the shape of obstacles. The regions in the sub-environment are therefore only classified by complexity level. Three levels: Free, Obstacles and Block levels are defined. In the main container of the AP, an EMA is located. The environment sensors should provide information about the area under their control. The EMA works as a coordinator agent for robotic agent services and at the same time works as a user interface for user agents. It is the place where the targets and the track of all RAs in the system, are stored. Users use UA on their PDA to assign the tasks to the system by cloning the interface for assigning tasks from VDF. Our lab environment is divided into 21 observable regions as in Figure 4 below.

The conventional tele-robotic system for comparison:

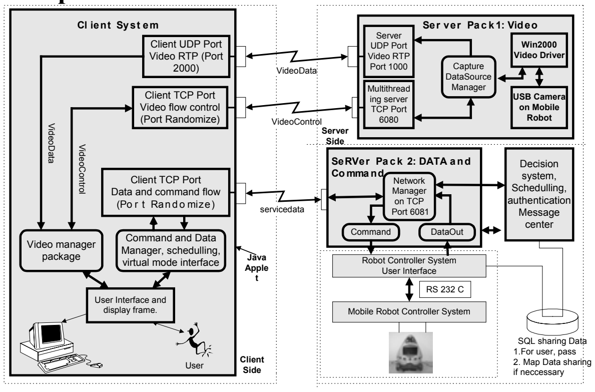
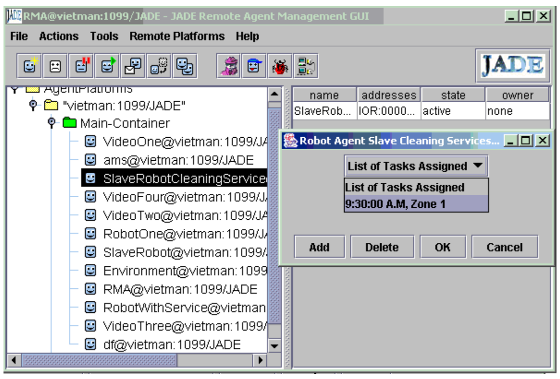

Client System
Client UDP Port Video RTP (Port 2000)
Client TCP Port Video flow control (Port Randomize)
VideoData
VideoControl
Client TCP Port Data and command flow (Port Randomize)
Video manager package
Command and Data Manager, schedulling, virtual mode interface
Java Applet
User Interface and display frame.
User
Client Side
VideoData
VideoControl
servicedata
Server Side
Server Pack1: Video
Server UDP Port Video RTP Port 1000
Multithreading server TCP Port 6080
Capture DataSource Manager
Win2000 Video Driver
USB Camera on Mobile Robot
SeRVer Pack 2: DATA and Command
Network Manager on TCP Port 6081
Command
DataOut
Decision system, Schedulling, authentication Message center
Robot Controller System User Interface
RS 232 C
Mobile Robot Controller System
SQL sharing Data
1.For user, pass
2. Map Data sharing if neccessary

Figure 5. Conventional control System on ISSAC

A conventional tele-robotic control system on the same robot model (ISSAC) is also developed to compare with the proposed framework. The full implementation of the system is described in (ToDong, et al., 2002). This system solves many situations in online and offline mode. This conventional system includes two parts. On the right side of Figure 5 is the server part, which is located on the mobile robot. The code for controlling the robot and obtaining sensor data is located on the embedded board. This board connects to the server program on the controlling computer through serial port. The server is coded in two separate packs. One is for sending audio/video data to client using Java Media Framework and the other is for receiving command, service requirement data from client, storing this data locally to implement later. On the left side of Figure 5 is the client part. This client part is also located on the ISSAC as an applet. At the time users visit the ISSAC control pages, this applet is downloaded and executed on the client computer.

Using this system, users can control the ISSAC and work with its services. Users can customize ISSAC behaviour option but only with one robot at a time. Users can assign the cleaning task to the robot using the interface in the applet downloaded from ISSAC. However, users can only work with the robot service when both the user and the robot are online simultaneously. The service requirements are then stored locally. During the time users connect with the robot, a permanent stable network connection is required. This problem can be overcome by writing the application on client computer instead of downloading this program as an applet through the network. However, in that case, users should know whether this robot is working on this environment. If it is, then users obtain the client program of this robot and send our requirement.

Experiment result:

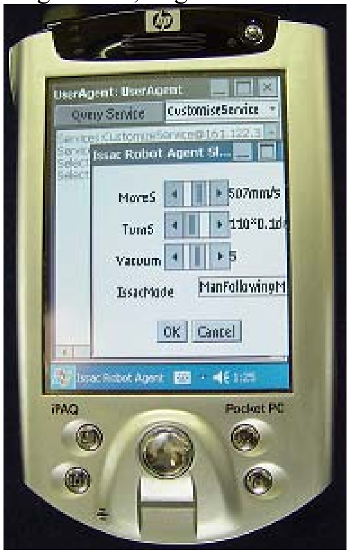

RMA@vietman:1099/JADE - JADE Remote Agent Management GUI
File
Actions
Tools
Remote Platforms
Help
JADE
Robot Agent Slave Cleaning Services...
Add
Delete
OK
Cancel

AgentPlatforms
"vietman:1099/JADE"
Main-Container
VideoOne@vietman:1099/JA
ams@vietman:1099/JADE
SlaveRobotCleaningService
VideoFour@vietman:1099/J
VideoTwo@vietman:1099/JA
RobotOne@vietman:1099/J
SlaveRobot@vietman:1099/
Environment@vietman:1099
RMA@vietman:1099/JADE
RobotWithService@vietman
VideoThree@vietman:1099/
df@vietman:1099/JADE

name	addresses	state	owner
SlaveRob...	IOR:0000...	active	none

List of Tasks Assigned
List of Tasks Assigned
9:30:00 A.M, Zone 1

6.a Cleaning service, migrated to main container



6.b. Behaviour customisations services

Figure 6. Service Interface

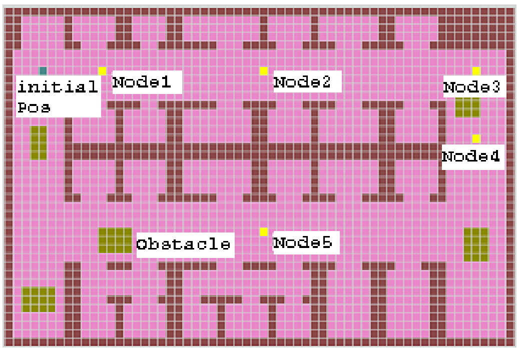
Figure 6 shows the result of implementing our framework for cleaning service and behaviour customisation service. Both the proposed framework and conventional framework can offer cleaning services and customisation services. However, with cleaning service, the proposed framework gives the ability of working offline with the services, which users cannot do on the conventional system. The behaviour customisation system shows the ability of changing the behaviour of a group of robots. Moreover, the interface for changing this behaviour is provided by the robots themselves, so many types of robots can provide this service if these robots are designed according to the architecture without changing the code at the VDF and UA. Each service of the proposed system is cloned and moved to user container separately on request. On the contrast, all the interface code of the conventional system should be loaded when users connect with ISSAC.

In Table 1, the size for each service in the proposed system is smaller than the size of the service in the conventional system because the service interfaces in the proposed system are separated from each other. Meanwhile, users should download the whole applet with all services in the conventional system. The time for starting any service in the conventional system is the same as the time for downloading the whole client program. The time to complete a service seems to be similar. However, the conventional system requires users and robots to be online at the same time.

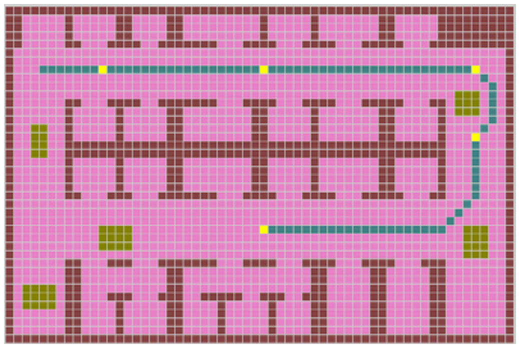
Table 1. Service experiment result

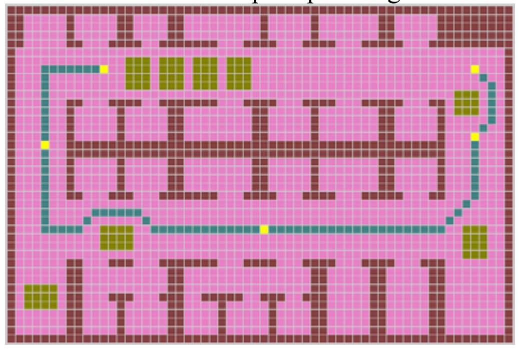
	Average service starting time	Average time to complete a service	Time that both user & robot require to be online
Behaviour Service Slave Agent 18.6 Kb	18 (second)	60 (second)	0 (second)
Cleaning Service Slave agent 28 Kb	20 (second)	120 (second)	0 (second)
Behaviour Service on conventional system 96Kb	85 (second)	60 (second)	145 (second)
Cleaning Service on conventional system 96Kb	85 (second)	120 (second)	205 (second)

Figure 7 show the results of using coordinator agent in dynamic path planning. After locating all the targets via EMA, the path planning is created by using the Floyd algorithm for finding shortest path. The path-planning task can be done on the RA side or on the EMA side. The travelling code corresponding to the environment map is stored on the EMA. If the path planning is performed on RA side, the initial estimated travelling cost should be sent before starting. This information is stored as a two dimensional table of 21 nodes, which shows the travelling time between each node couple. Then during the navigating time, if some change in the environment is detected, this information is sent to the RA from EMA. If the path planning is done on EMA side, the next targets from EMA should be sent during the navigating tasks. The number of messages in that case is smaller. However, the stable communication is required during the time of performing tasks.



7*a*. Initial State at EMA


7*b*. Archive path-planning


7*c* Path change at Node 1 due to environment change and one more destination added

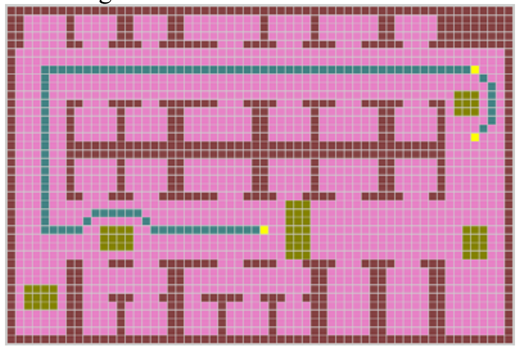

7*d*. Path change again at node 5 due to environment change (Blocked)

Figure 7. Dynamic path planning using the responsive environment coordination.

The path planning system should upgrade the Floyd table when RA arrives at each node so that any change in the environment creates a change of path planning at each node. Each time RA arrives at a node, the plan needs to be changed. System recalculates the Floyd table so any new change in the environment is calculated. The environment sensors may change the travelling cost value between nodes during the last move. This cost depends on the complexity of the region and neighbourhood region. From the new Floyd table, a new graph containing remaining target points and current robotic positions is formed. The edges of this graph are the shortest paths found in Floyd algorithm. Now it becomes a traditional Travelling salesman problem (TSP). TSP is a NP-complete problem so no solution can offer real-time performance. The shortest path on this graph is found using full search if there is less than 8 points. If the number of remaining nodes is bigger than eight, some heuristic methods can be used to reduce the calculation time.

To create the dynamic Floyd table, the relation between the environment state and the travelling code is needed. The travelling code can be calculated by the distance between nodes. However, obstacles may change this value. The state space of n regions with only m complexity levels is m^n. Therefore, to do the path planning in real time, an estimation of travelling cost is proposed. In our implementation, the simplest case, where the EA recognizes three states **{Block, Empty, Obstacle}** levels, is chosen. The value table contains just the initial value. Statistical information is used to recalculate it later on. The initial value table V[21][21][4] is calculated using the state couple of Si and Sj by:

- V[i][j][0] = for free-block, block-free, obstacle-block, block-obstacle and block-block.
- V[i][j][1] = distance between nodes i and j for free-free state couple of Si-Sj.
- V[i][j][2] = distance between nodes i,j * 4 for free-obstacle and obstacle-free couple of Si-Sj.
- V[i][j][3] = distance between nodes i,j * 8 for obstacle-obstacle couple of Si-Sj.

Table 2. ACL Message Load

	Path planning on RA	Path planning on EMA
Initialisation Message	441	0
Messages if there is a change in environment	number of node change	0
Message transfer when robot arrive at a node	0	2
Time to get information at each node.	0	1 (second)

Table 2 shows the number of messages needed in path planning process. Doing the path-planning on EMA requires not many messages. However, during the navigation, the next targets are sent to RA. Therefore, this method fails when network problem occurs. On the other hand, doing the path-planning on RA requires many messages at the first time. In case, network problem happens, it follows its current path planning to finish the tasks.

4. CONCLUSION

The Internet HRI solution using a responsive environment and service management is good for autonomous robot working in dynamic environments with dynamic assigning tasks. To explore these environments and working with robots, the use of VDF, slave agents and EMA is proposed. Through experiments, the proposed framework shows some advantages to the conventional tele-robotic control system. Users can access services of the system without installing the specific program for each robot. Total time for agent migration and synchronization is small enough to give the system a high ability level and help users save a lot of time to control the system. This method also reduces the time that robots need to be connected to the network and can tolerate some levels of connection disruption.

REFERENCES

Ali, K. and Arkin, R.C. (2000), "*Multi-agent Teleautonomous Behavioural Control*", in Machine Intelligence and Robotic Control, **Vol. 1, No. 2**, 2000, pages 3-10.

Bellifemine, F., G. Caire, A. Poggi and G. Rimassa (2003), "*JADE - A White Paper*", in Journal "EXP - in search of innovation", September 2003, volume 3, pages 6-19.

Bum-Jae You, Myung Hwangbo, Sung-On Lee, Sang-Rok Oh, Young Do Kwon, San Lim (2003), "*Development of a home service Robot 'Issac*", in Proceeding of the International Conference on Intelligent Robots and System (IROS2003), V**ol. 3**, pages 2630-2635.

Dong To Nguyen, Sang-Rok Oh, Bum Jae You (2002), Myung Hwang Bo and Brian Kwang-Ho Lee, "*A Control architecture for an Internet-based robot system*", in Proceeding of the International Conference on control, automation and system (ICCAS 2002), pages 1677-1683.

FIPA – Foundation for Intelligent Physical Agents (2000) http://www.fipa.org.

Hideaki Takeda, Nobuhide Kobayashi, Yoshiyuki Matsubara and Toyoaki Nishida (1997), "*Towards ubiquitous human-robot interaction*", in Working Notes for IJCAI-97 Workshop on Intelligent Multimodal Systems, pages 1-8, 1997.

Huang K. and M. Trivedi (2003), "*Video arrays for real-time tracking of person, head, and face in an intelligent room*" in Machine Vision and Applications, vol. 14, June 2003, pages 103-111.

Maxim Makatchev and S. K. Tso (2000), "*Human-Robot Interface Using Agents Communicating in an XML-Based Markup Language*", in Proceedings of the 2000 IEEE International Workshop on Robot and Human Interactive Communication, Osaka, Japan, September 27-29, 2000, pages 270-275.