International Journal of Energy, Information and Communications
Vol.8, Issue 1 (2017), pp.1-14
http://dx.doi.org/10.14257/ijeic.2017.8.1.01

An IoT Application Connecting Edge Resources and Cloud Resources using Agents

Yuki Kaeri*[1], Yusuke Manabe[2], Kenji Sugawara[3] and Claude Moulin[4]

[1]Graduate School of Computer and Information Science, Chiba Institute of Technology, Japan
[2,3]Faculty of Information and Network Science, Chiba Institute of Technology, Japan
[4]Sorbonne Universités, Université de Technologie de Compiègne, JR Unit - CNRS 7253, Heudiasyc, Centre de Recherche de Royallieu, 60205 Compiègne Cedex, France
[1]yk.kaeri@gmail.com, [2]ymanabe@net.it-chiba.ac.jp, [3]suga@net.it-chiba.ac.jp, [4]claude.moulin@utc.fr

Abstract

The new technology to integrate Edge Computing and Cloud Computing is called Internet of Things (IoT). Current research on IoT mainly focuses on how to connect Edge Computing to Cloud Computing. However, Oihui Wu, et al. argued that simply connecting them is not enough; beyond that, objects should have the capability to learn, think, and understand both physical and social worlds by themselves. An architecture of IoT applications is being developed to connect Agent Spaces, including Personal Assistant Agents and Service Agents in order to support user activity at the office, to both the Edge and the Cloud. In this paper, an architecture of IoT applications is proposed connecting Edge Resources and Cloud Resources. According to the proposed architecture, an Agent Space is designed in which agents can control the resources via programs named Resource Connectors. Finally, an IoT application was prototyped to support users working at the office with a Personal Assistant Agent in an Agent Space that has a vocal interface with users and a web interface.

Keywords: *Internet of Things, Personal Assistant, Agent Platform, Office Work Support*

1. Introduction

Using the Information and Communication Technology (ICT), people are able to easily send (store) and receive (access) information in the Cloud. Furthermore, the rapid development of the ubiquitous technology also enables information systems to access Real Space (RS) by mobile communication devices such as smart phones. Various kinds of sensor devices are set to be deployed in any place with RS and copious amounts of multimodal information of users, and their surroundings can be acquired and saved in the Cloud. Using this information, very convenient services called "Anytime," "Anywhere," and so on have been widely provided for users on a daily basis [1].

However, users feel a heavy burden when trying to quickly find information or needed assistance because the important information may be hidden in enormous amounts of useless information in the Cloud [2]. When a user has to watch over some event or find somebody through information that the Cloud provides, the user may have to keep watching some information devices. Unskilled users may be prevented from opportunities they could take if they had the skill of utilizing information systems.

* Corresponding Author

The agent-based technology is one of the solutions to realizing a function to distribute information from the Cloud to an unskilled user in RS [3]. An Agent Space (AS), that is a well-designed collection of agents, can be a platform to meet users' requests in the RS with proper social information in the Cloud [4]. The users' requests change from moment to moment according to the users' activities in their society. On the other hand, contents and organization of the social information in the Cloud also change according to the change of the society in the RS. Then, a problem to meet users' requests with social information must be very difficult for conventional static algorithms to solve autonomously [5]. This is a reason why users require skills to handle the computer and network conveniently. This is also another reason why even skilled users should spend their time to keep watch at web sites and data in the Cloud.

Recently, computational resources, including ubiquitous devices and networks working near users, become more reliant on the development of advanced ICT applications. Information processing using resources and networks is called Edge Computing. The new technology to integrate Edge Computing and Cloud Computing is called Internet of Things (IoT). The IoT is expected to have significant home and business applications to contribute to the quality of life and to grow the world's economy [6]. Current research on the IoT mainly focuses on how to connect Edge Computing to Cloud Computing. However, Oihui Wu, et al. argued that simply connecting them is not enough; beyond that, objects should have the capability to learn, think, and understand both physical and social worlds by themselves [7]. An architecture of IoT applications is being developed to connect AS, including Personal Assistant Agents (PAs) and Service Agents (SAs) in order to support user activity at the office [8], to both the Edge and the Cloud.

In this paper, the sensors—the devices as well as data captured by them—are called Edge Resources. In addition, programs to control sensors and to analyze them in local computers are also called Edge Resources. The raw data captured by the Edge Resources and analyzed data transformed from the raw data can be saved in Cloud databases. Web data and programs, including the data described previously, are called Cloud Resources. The IoT Applications are expected to support users' activities in life using the Edge Resources and the Cloud Resources.

In Section 2, an architecture of IoT applications connecting Edge Resources and Cloud Resources is proposed. In Section 3, according to the proposed architecture, an AS is designed in which agents can control the resources via programs named Resource Connectors (RCs). In Section 4, an IoT application was prototyped to support users working at the office with a PA in an AS with a vocal interface between users and a web interface.

2. An Agent Space Connecting Edge Resource and Cloud Resource

An AS is designed to help users access useful services suggested by the Edge Resources and the Cloud Resources as shown in Figure 1. A program that connects an agent to an Edge Resource or a Cloud Resource is called an RC in this paper. An RC sends agent messages to agents and receives them from agents in the AS. A collection consisting of agents and RCs to connect the Edge Resources and the Cloud Resources in order to realize an application for supporting a user is called a coterie as shown in Figure 1.

Each agent in an AS makes a decision to act based on several kinds of knowledge based frameworks to realize users' requirements cooperatively with other agents in an AS using social knowledge and according to user's preference and activity. A set of agent runtime systems working in computers connected via LANs and the internet is called an AS platform. Normally, an AS platform is realized by an Agent Platform. An AS platform provides a set of basic cooperation protocol and a basic ontology set in order for agents and RCs to communicate with each other. A group of agents and RCs realizes an application logic using its application oriented cooperation protocol and an ontology designed using the basic protocol and a basic ontology. This ontology is designed to link resources and agents using common symbols defined in an AS, and consists of resource details described according to specifications of the devices or the programs and agent details described according to usage of the resources by the agent.

In this paper, some PAs are presented that were designed to watch users' activity and objects surrounding them and some SAs that read and write data in the Cloud. A PA works for a particular user to process his/her requests that are acquired through a keyboard interface, vocal interface, gesture interface and so on. An SA reads and writes particular documents, web pages, or databases according to tasks given by a developer, and provides information about them for other agents.

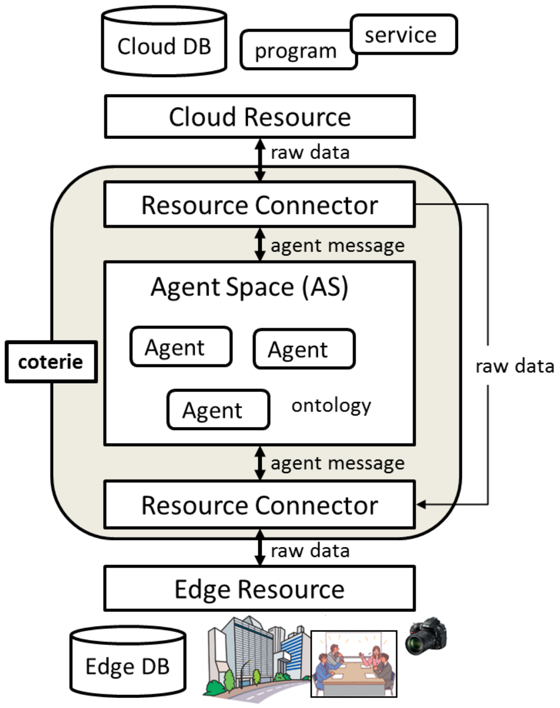


Figure 1. Architecture of IoT Applications Connecting Edge Resources and Cloud Resources

An RC is designed to transform signals from a particular device deployed in a room or data from a particular program in the web (Cloud Resource) into symbols that compose a message sent to agents. The message is composed based on a protocol and an ontology set defined by application logic that the agent realizes as a component. For example, an RC named Speech-To-Text Agent includes a plug-in program that recognize voice signal from a microphone and transforms it into a sentence. Then the RC sends a message that includes the sentence as content to a PA that watches a particular user who is talking to it.

3. A Design of a Coterie Consisting of an Agent Space and Resource Connectors

3.1. Agent Platform for Agent Space

An agent platform is a collection of runtime systems and development support systems of agents that are connected via LANs and the Internet. OMAS agent platform (OMAS-P) was used as a platform of agents that is developed using Allegro Common Lisp and has been used to develop various kinds of agent-based systems [8]. The OMAS-P is better suited for developing knowledge-driven multi-agent systems that realize intelligent application logic. A runtime environment of the OMAS-P running on a PC is called a coterie, and it can be connected to other OMAS-Ps via the Internet using some communication agents deployed in each OMAS-P. Therefore, several OMAS-Ps can be unified in an AS where agents can send and receive agent messages by indexing agent names with each other. A developer using the agent platform can add agents to the platform dynamically according to their needs. Therefore, the concept of the coterie is suited for developing larger AS.

In this paper, Resource Connector Platform (RC-P) was developed as a platform of RCs because the proposed plan was to use plug-in modules that are installed in Windows OS in order to control devices and software as shown in Figure 2. Accordingly, the RC-P is developed by the C# (.NET framework).

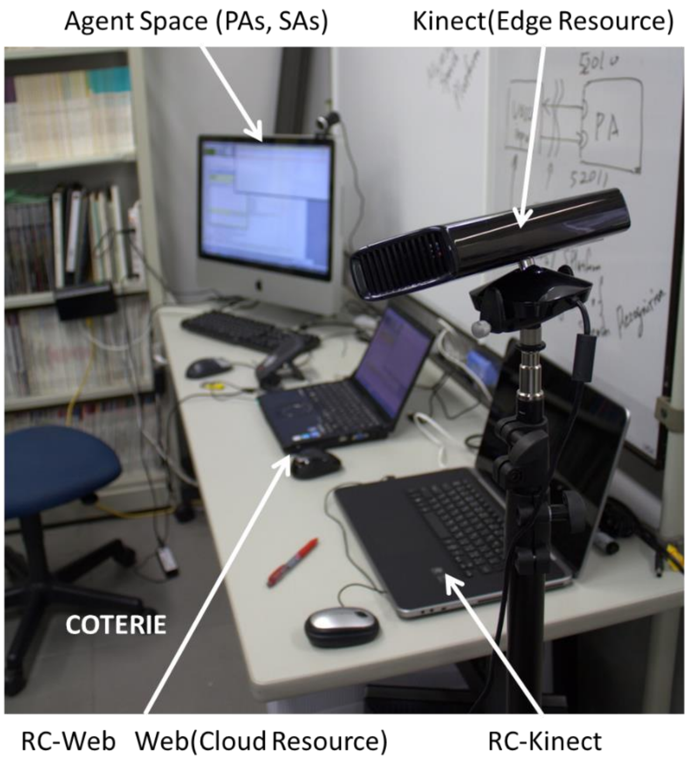


Figure 2. Experimental System of a Coterie that connects a Web resource and an Edge resource

The RC-P in Figure 2 is connected by the same communication protocol as the OMAS-P and is able to wrap plug-in modules in the .NET framework. An RC running on the RC-P can join to a coterie of agents running on the OMAS-P easily and dynamically. Although the RCs do not have functions to realize intelligent logic, they can transform signals from a device or data from a program into an agent-based message for the agents.

A coterie in Figure 2 consists of an AS working in a PC and an RC-P for Edge Resources working in another PC and an RC-P for Cloud Resources working in the third PC. An agent in the AS is generally a multi-agent system to realize a complicated application logic that works cooperatively with other application systems in the Cloud. An agent generally consists of component agents that run on the OMAS-P sending and receiving messages based on the OMAS communication protocol in this paper.

Figure 3 shows a structure of an RC running on the RC-P that connects agents in an AS working on an agent platform, that is the OMAS-P or the Jade agent platform, for example, to some Edge Resources or Cloud Resources. An RC consists of three modules: an Agent Platform Interface, a Plug-In Interface, and a Translator. The Translator reads a Resource

Profile and an Agent Profile that define specifications of a resource and an agent to connect, and translation methods between agent messages and signals/commands (raw data) that are output from and input into the resource. The procedures of the Agent Platform Interface, the Plug-In Interface, and the Translator are general, but the Resource Profile depends on the resource, and the Agent Profile depends on agents that receive the agent messages. Depending on an IoT application, a developer of the RC should describe the Resource Profile and the Agent Profile using provided forms written by the JSON.

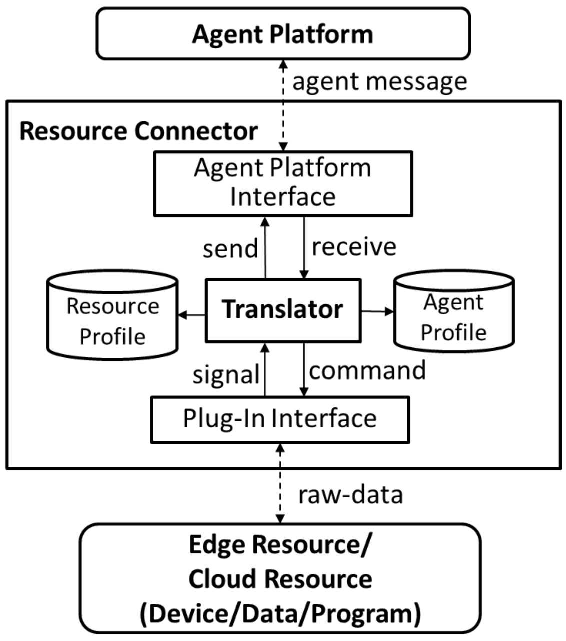


Figure 3. Structure of a Resource Connector

Figure 4 shows a basic set of communication protocols between an agent and an RC that are controlled by a Translator that refers to an Agent Profile. For example, the type-1 in Figure 4 defines a simple interaction between a requester agent in an AS and an RC. After the RC receives the request message, a Translator module translates the message to a Command to control a resource based on a specification of it written in a Resource Profile, and a sequence of received signals are translated into an agent message based on the Resource Profile and the Agent Profile. The agent message is replied to the requester agents. The type-2 in Figure 4 shows a protocol in which an RC becomes a requester agent to send a request message to a different agent or an RC to process the request message that was received from an original requester agent. The type-3 in Figure 4 shows an operation when receiving a broadcast message or an inform message from an agent. The type-4 in Figure 4 shows an operation when a request is sent as a command. At this time, the RC executes the command and does not reply to the agent.

3.2. An Experimental Design of an Agent-based IoT Application for Supporting Office Work

To validate the functions of the proposed application using RCs that connect Edge Resources and Cloud Resources, a task was chosen of an office work support as a simple test bed. A system can be constructed by the proposed method and evaluated to determine whether software that achieves the following scenario can be constructed. A user who works for a travel agency has to make a plan for a client to take a trip to a resort area along with her colleagues. She searches many web pages to find good airlines, railroad maps, rental cars, hotels, and site-seeing spots to meet various kinds of requests of the client very quickly.

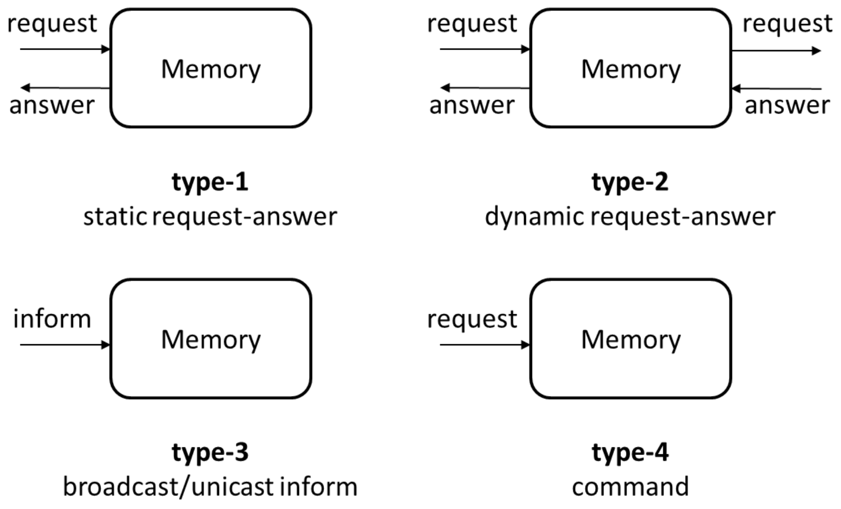


Figure 4. Basic Agent Protocol between agents and RCs defined in Agent Profile

Figure 5 shows a simple structure of a coterie consisting of RCs and an AS that was prototyped for the office work support scenario. This coterie includes a PA to support a user in the above office scenario, two SAs to provide a database service, and a web page searching service for the PA.

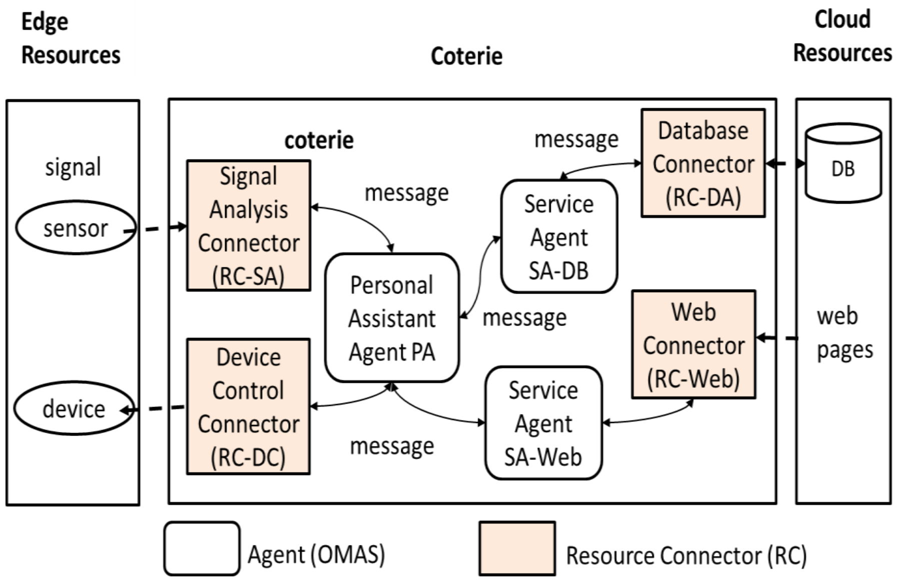


Figure 5. Structure of a Coterie consisting of RCs and Agents

Figure 6 shows an experimental design of a PA and RCs that cooperate with the PA. They work in an agent-based IoT application shown in Figure 5 to support office works. The PA is a multi-agent system that consists of OMAS agents named Decision agent, a Voice Conversation agent, a User-Watch agent, and a Task Support Agent.

A Video Conversation agent receives messages that include spoken texts from an RC-Speech-To-Text and sends messages to an RC-Text-To-Speech. The User-Watch agent receives messages that include user's actions that are defined in a knowledge model. An RC named RC-Kinect generates symbols of the actions from signals captured by a Kinect device. A Task-Support agent sends messages including commands to an RC-Visio to write figures in a display and receives messages including events from the RC-Visio that are captured by a Visio program when a user writes something on the display. An RC-iTunes is an RC that receives messages form a Decision Agent (DA) directory to control the iTunes application to show video images to a user.

3.3. Integrated Development Environment of OMAS-P and RC-P

The OMAS-P has provided a graphical user interface to develop programs and support debug for agent developers. Figure 7 shows part of the OMAS support functions, consisting of an OMAS-Message-Diagram to check interactions among agents, an OMAS-Message-Panel to observe contents of messages that agents send and receive, and an OMAS-Control-Panel to control and check each agent's actions. In an RC-List, a list of resources currently available on the computer by the RC, is displayed. In an RC-Control / Message-Panel, the setting of the current operation of the resource, the configuration and the status log of a data sender / a receiver are displayed. In the OMAS-Message Diagram, all messages exchanged in an AS are displayed, and the status log and flow of sender / receiver can be confirmed by a sequence diagram. Agent developers can be supported by these functions. In this paper, the OMAS-P was extended to cover the RC-P to observe, control, and check activities of RCs in a manner similar to the OMAS-P.

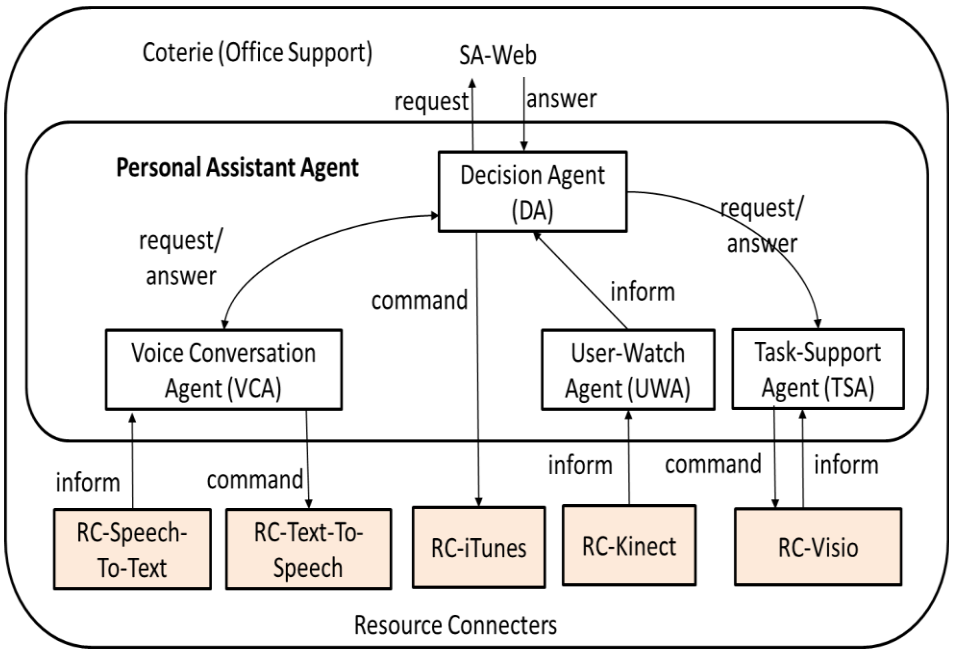


Figure 6. A Structure of a Personal Assistant Agent (PA) Consisting of OMAS Agents and RCs

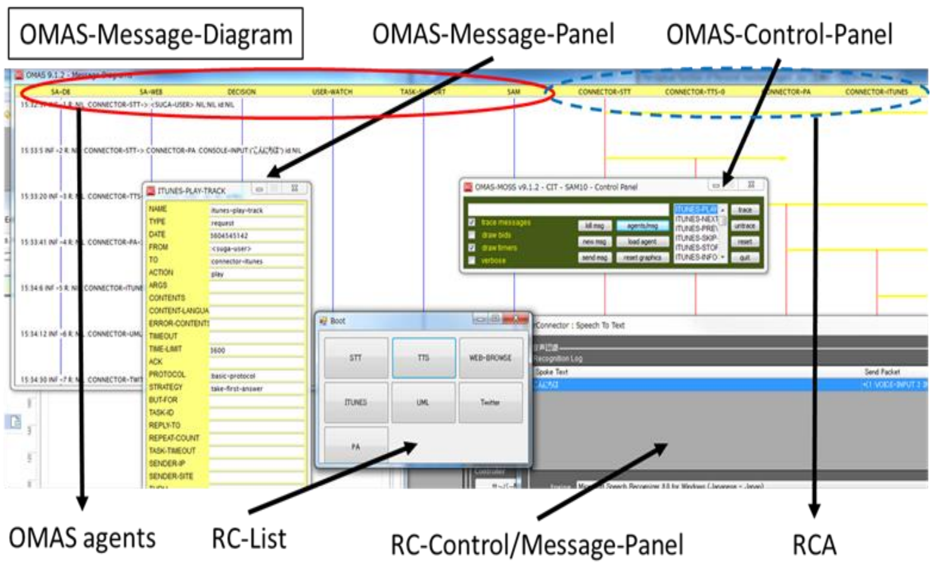


Figure 7. Developer Tool for OMAS-P and RC-P

4. Implementation of an Agent Space to Support a User at a Desk

4.1. Scenario of Supporting a User Working at a Desk

A worker who is a user of a PA in this scenario, is working at her desk to make a plan of a trip for a client as shown in Figure 8. A colleague comes by her desk and asks her about the plan of which they are in charge. She wants to show and explain why the schedule is behind what they arranged.

First, she wants to show him a plan of “5 days in Tokyo” for French clients. Then, she asks her PA to show it on a big display on a wall conversing with the agent by voice as follows;

User> Agent. Show me the document about hotel reservations that I made yesterday afternoon.

PA> This is a list of documents you made yesterday afternoon.

The list is shown in a tablet PC that only she can see.

User> Show us the Visio document titled Trip-plan-2016-11-11.

PA> Is this right?

User> OK.

A Visio document is shown in a big display that two persons can see at the same time.

Next, she tells him that the total price of rooms in a list of hotels inside and near Tokyo costs too much against the total trip budget. Then, she wants to change the list to better hotel reservations based on the colleague’s opinion. To do so, she prepared some bookmarks of hotel reservation sites the previous afternoon.

User> Agent. Show me a list of bookmarks that I created in the late afternoon.

PA> This is a list of web pages you opened between 4 pm and 5 pm.

User> Show us number 5.

Then, a web page of a hotel reservation site is shown in a big display.

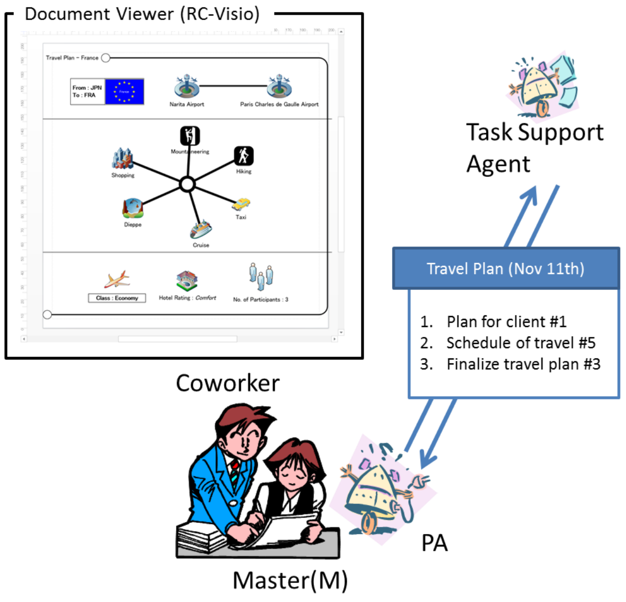


Figure 8. Scenario of Supporting a User Working at Desk

4.2. Implementation of Voice Conversation Agent

The Voice Conversation Agent (VCA) working as a component agent of a PA is a knowledge-intensive agent that finds user's requests by task-oriented conversation using a set of Extended Hierarchy State Transition Matrix and Ontology concerning the topic area [9] as shown Figure 9. A VCA for a particular user has ontology concerning the user, colleagues of the user, expertise on her job and so on, to find her requests. When the VCA makes a request clear as a task, it sends the task to the Decision agent in the PA. Then the Decision agent decides how to solve the problem to answer a result to the VCA. For example, if a task sent by the VCA is "to find all documents that she made between 12:00/26/03/2016 and 24:00/24/03/2016", the Decision agent will return a list of names of document back to the VCA in the office work scenario.

A VCA controls the RC-STT (Speech-To-Text) and the RC-TTS (Text-To-Speech) to interact with the user by voice. An RC-STT running on the RC-P recognizes a sentence that the user speaks, and sends a string of characters to an input variable of the VCA. When the VCA makes a sentence to talk to the Master, the VCA sends a message to the RC-TTS in the OMAS protocol.

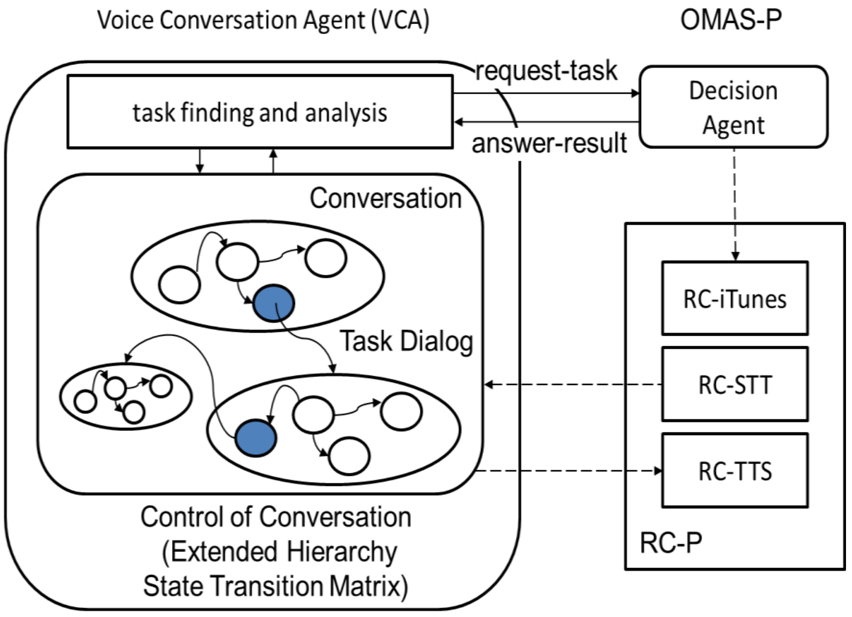

Voice Conversation Agent (VCA)
OMAS-P
task finding and analysis
request-task
answer-result
Decision Agent
Conversation
Task Dialog
RC-iTunes
RC-STT
RC-TTS
RC-P
Control of Conversation (Extended Hierarchy State Transition Matrix)

Figure 9. Vocal Conversation Agent in OMAS-P and Vocal Resource Connectors

In this example, if the user sends a request to the Decision agent to “Play a video of the resort spot for a colleague,” the RC-iTunes running on the RC-P plays a video image when it receives a command message from the Decision agent.

4.3. Implementation of RC-Kinect

It is necessary for a PA to understand a request of the user through interaction with the Master using awareness of the Master other than by vocal conversation. In order to watch the user and her surroundings, a Kinect sensor device and a software development environment of Open NI for Kinect were used in this experiment. An RC-Kinect Agent consists of a connecting function (Kinect-Connect) to the OMAS agent and a wrapping module (Kinect Open NI) to wrap an OpenNI for Kinect as shown Figure 10. A plug-in program corresponding to the necessary profile and resource in Figure 10, an agent and an RC capable of the operation shown in the sequence diagram were developed.

The right side of Figure 10 shows a message diagram between the User-Watch agent and the Kinect Agent. Figure 11 shows the report message to the User-Watch agent (as OMAS PA in this case) corresponding to an event “Person-1 appears” and “Person-1 disappears” when the Kinect Open NI was connected to the Kinect Agent. In RC-P-IDE for RC-Kinect Window, the skeleton and image of the person detected by Kinect, and information such as the number of people detected at present are displayed for debugging.

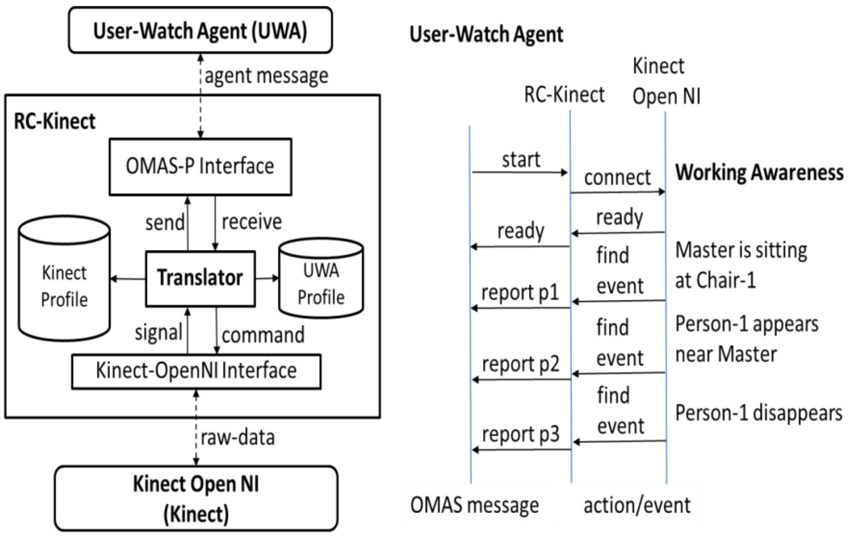
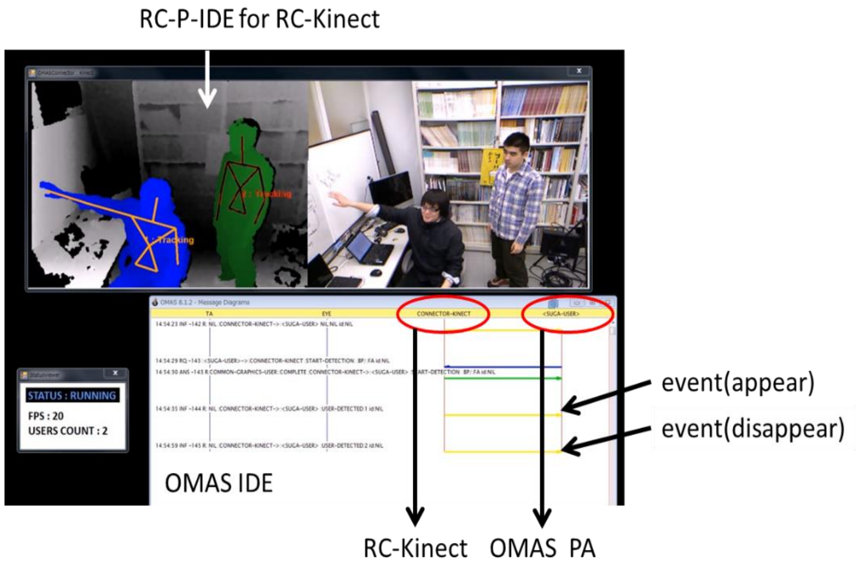


Figure 10. Design of RC-Kinect



```
14:54:23 INF -142 R: NIL :CONNECTOR-KINECT-> :<SUGA-USER> NIL NIL id NIL
14:54:29 RQ -143 :<SUGA-USER>-> :CONNECTOR-KINECT :START-DETECTION :BP/ :FA id NIL
14:54:30 ANS -143 R:COMMON-GRAPHICS-USER: COMPLETE :CONNECTOR-KINECT-> :<SUGA-USER> :START-DETECTION :BP/ :FA id NIL
14:54:35 INF -144 R: NIL :CONNECTOR-KINECT-> :<SUGA-USER> :USER-DETECTED 1 id NIL
14:54:59 INF -145 R: NIL :CONNECTOR-KINECT-> :<SUGA-USER> :USER-DETECTED 2 id NIL
```

Figure 11. Development Support Tool for RC-P

4.4. Implementation of Web Page Agent

Web Agent in Figure 12 is an RC that mines web pages as a collection of data using a plug-in module running on Windows OS according to a request from an SA-Web. The Web Agent processes a number of texts and images in web pages using certain data mining algorithms and transforms them into a message including knowledge representation defined by an ontology set based on a protocol between the Web Agent and the SA-Web agent.

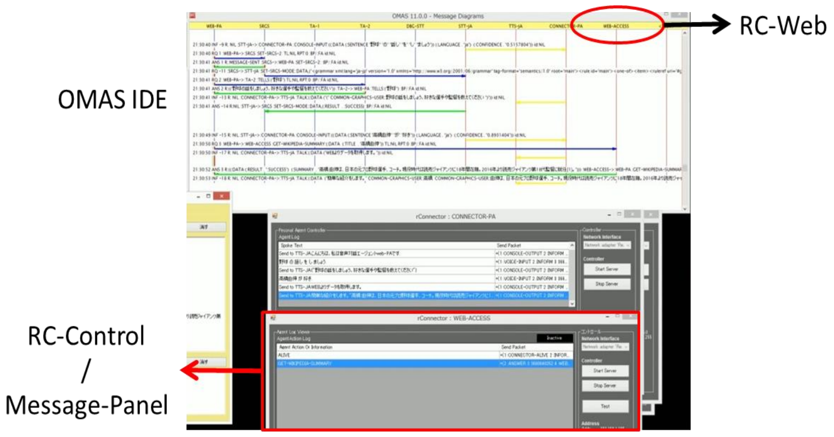


Figure 12. Debug Views of Web Agent

5. Conclusion

In this paper, sensor devices as well as data captured by them are called Edge Resources. In addition, programs to control them and to analyze them in local computers are also called Edge Resources. The raw data captured by the Edge Resources and analyzed data transformed from the raw data can be saved in Cloud databases. Web data and programs as well as the data described previously are called Cloud Resources. One of the challenging problems in developing an IoT application is to design an architecture to connect Edge Resources, including sensors and pattern recognition programs, to Cloud Resources, including databases and web pages. We proposed an agent-based methodology to develop IoT applications that connect Edge resources and Cloud resources to agents.

Figure 1 shows our proposed architecture for realizing intelligent IoT applications using an AS and RCs. AS consists of various kinds of multi-agent systems that provide services of IoT applications to watch users' actions and changes of environments using sensors, for example, as well as to find knowledge for the users from Cloud Resources. A multi-agent system is also expected to help unskilled users utilize sophisticated services of Cloud Computing using a vocal interface or a gesture interface, for example. In the implementation for the scenario in Chapter 4, we confirmed that it has sufficient functionality to achieve the purpose of the scenario because the implementation of the functions and the confirmation of the implementation were made according to the proposal of this research. One strong point of the proposed method is that each resource can be reused. We are currently conducting experiments to construct a system using the proposal of this paper. In addition, the weak point is that plug-in programs must be developed for corresponding resources.

In this paper, we presented an IoT application of a PA that converses with a user using a vocal interface in Edge Resource to show a web page requested by the user. Although a set of functions of pattern recognition of the PA is still small and limited, the structure of the PA is flexible because agents and RCs (Resources) can be replaced easily in the proposed architecture. When the function and performance of an Edge Resources or a Cloud Resource is improved, our PA can incorporate the improved resources by modifying the Resource Profile and Agent Profile of the RC. We are developing new IDE of RC-P to enhance this property in the future.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Number 26330260. We also would like to thank Editage (www.editage.jp) for English language editing.

References

[1] K. Lyytinen and Y. Yoo, "Issue and Challenges in Ubiquitous Computing", CACM, vol. 45, no. 12, **(2002)**, pp. 63-65.
[2] M. Kitsuregawa, "Info-plosion: Retrospection and Outlook", The Journal of the Institute of Electronics, Information, and Communication Engineers, vol. 94, no. 8, **(2011)**, pp. 662–666.
[3] S. Russel and P. Norving, "Artificial Intelligence A Modern Approach", Prentice Hall, **(2003)**.
[4] N. Shiratori, K. Sugawara, Y. Manabe, S. Fujita and B. Chakraborty, "Symbiotic Computing Based Approach towards Reducing User's Burden Due to Information Explosion", IPSJ Journal, vol. 51, no. 12, **(2011)**, pp. 1234-1243.
[5] K. Sugawara and J.-P. A. Barthès, "An Approach to Developing an Agent Space to Support Users' Activities", Proc. of Fifth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services CENTRIC 2012, Lisbon, Portugal, **(2012)**, November 18-23.
[6] A. Al-Fuqaha, M. Guizani, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", IEEE Communication Survey & Tutorials, vol. 17, no. 4, **(2015)**, pp. 2347-2376.
[7] O. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang and K. Long, "Cognitive Internet of Things: A New Paradigm Beyond Connection", IEEE Internet of Things Journal, vol. 1, no. 2, **(2014)**, pp. 129-143.
[8] J.-P. A. Barthès, "OMAS - A flexible multi-agent environment for CSCWD", Future Generation Computer Systems, vol. 27, **(2011)**, pp. 78-87.
[9] K. Sugawara, S. Ben Yaala, Y. Manabe, C. Moulin, N. Shiratori, J.-P. A. Barthès, "Conversation-based Support for Requirement Definition by a Personal Design Assistant", Proc. of 10th IEEE International Conference on Cognitive Informatics and Cognitive Computing, pp. 262-268, Banff, Canada, **(2011)**, August 18-20.

Authors

Yuki Kaeri, received his Master's degree in Computer and Information Science from the Graduate School of Chiba Institute of Technology, Chiba, Japan. Currently he is pursuing his PhD in the same school. His research interests include Multi-Agent Systems, Human-Agent Interaction, Symbiotic Computing, Web Recommendation Systems, Ubiquitous Computing, and Internet of Things.

Yusuke Manabe, is an associate professor of the Department of Information and Network Science, Chiba Institute of Technology, Chiba, Japan. He received a PhD in Software and Information Science (2008) from Iwate Prefectural University, Japan. His research interests include Intelligent Informatics, Cognitive Science, Chaotic Time Series Analysis, and Soft Computing. He is especially interested in human skills analysis, context-aware computing, and symbol-grounding mechanisms. He is a member of JSAI, JCSS, SOFT, and IEICE in Japan.

Kenji Sugawara, is a professor of the Department of Information and Network Science, and the Dean of Faculty of Information and Computer Science, Chiba Institute of Technology, Chiba, Japan. He received a Doctoral degree in Engineering (1983) from Tohoku University, Japan. His research interests include Multi-Agent Systems, Artificial Intelligence, Ubiquitous Computing, and Symbiotic Computing. He is a fellow of IEICE Japan and a member of IEEE, ACM, and IPSJ.

Claude Moulin, PhD in Computer Sciences since 1998, Claude Moulin has been working for four years in research centers in Italy. He is now Assistant Professor at the University of Technology of Compiègne (France) and a member of the laboratory HeuDiaSyc (Information Knowledge and Interaction Team). He has a background in Knowledge Base Systems, Knowledge Engineering and web service semantic description, ontology design, and formalisms. His current research interests include Multi-Agent Systems and interaction with Multi-Modal Interfaces.