Journal of Information Processing Vol. 18 175–189 (June 2010)

Recommended Paper

Design of Adaptive Communication Mechanism for Ubiquitous Multiagent Systems

Taishi Ito,[†1,†2] Hideyuki Takahashi,[†2]
Takuo Suganuma,[†1,†2] Tetsuo Kinoshita[†1,†2]
and Norio Shiratori[†1,†2]

Agent-based middleware that has abilities of adaptation to dynamically changing environments is a significant direction for system developments in ubiquitous computing environments. In this paper, we focus on the communication infrastructure of agent-based middleware in ubiquitous computing environments. We propose an adaptive communication mechanism between agent platforms, which can select communication schemes flexibly, based on properties of inter-agent communication and resource status. We designed the proposed mechanism and implemented a prototype system. Furthermore, we performed an initial experiment by using the prototype system on a network environment with two types of access network. We confirmed that the dynamic selection of an inter-platform communication scheme works effectively according to the change of network resource status. From the experimental results, we confirmed that efficiency is improved 5% and stability is improved 22% when compared to that of the traditional mechanism.

1. Introduction

Recently, ubiquitous computing (ubicomp) environment [1], which comprise embedded computers, mobile terminals, sensor networks, and wireless access links, are emerging. Research and development is progressing actively to create new services and technologies to enrich this environment, such as context-aware service provisions, service integration, and middleware [2)–5)].

In general, because limitations of availability of computational and network resources exist, application systems that work on the ubicomp environment, in other words, ubiquitous applications, are required to have ability to adapt to various types of environment. They also have to cope with decentralization of context information and services. Hence, middleware to develop ubiquitous applications more effectively is promising [6].

Meanwhile, many studies of agent-based middleware have been investigated in recent years [7)–14)]. From the viewpoint of its high adaptation ability to environments, a multiagent system composed of agent-based middleware can fulfill the requirements of a ubiquitous application. Therefore, agent-based middleware is expected to be a potential framework to realize effective and advanced ubiquitous application development. However, there are problems to be resolved when agent-based middleware is applied to ubiquitous application development, such as instability and inefficiency of communication among agents, performance degradation caused by the overload of agents, scalability problems, etc. These problems originate in the characteristics of the ubiquitous computing environment where the limitation of the resources is more notable than the average computing environment.

Our study aims to provide advanced and stable services in ubicomp environments by using agent-based middleware to resolve the above-mentioned issues. In this paper, we focus on the problem of instability and inefficiency of communication between agents: we build an advanced communication infrastructure for the agent platform, considering situations of the environment and properties of inter-agent communications. We propose an "Adaptive Inter-platform Communication Mechanism" that can flexibly select inter-platform communication schemes according to network situations or computational resources and properties of inter-agent communications. This mechanism enables the agent platform to adapt to the various kinds of inter-agent communication requirements in a ubicomp environment with limited resources. Consequently, the mechanism can provide advanced and stable ubiquitous services with the ubiquitous applications that consist of the multiagent system.

In our previous works, we have presented the basic concept and outline of the proposed scheme, and have shown some results of the initial experiments [15)–17)].

†1 Graduate School of Information Sciences, Tohoku University
†2 Research Institute of Electrical Communication, Tohoku University

The initial version of this paper was presented at the DPS workshop held on Dec. 2008, which was sponsored by SIG-DPS. This paper was recommended to be submitted to IPSJ Journal by the chairman of SIG-DPS.

In this paper, we present a detailed design of the Adaptive Inter-platform Communication Mechanism, and describe the implementation of the prototype system based on the design. In addition, we performed further experiments by using the prototype system on two types of access networks. From the experimental results, we confirmed that the dynamic selection of an inter-platform communication scheme works effectively, according to changes in network resource status.

The remainder of this paper is organized as follows. In Section 2, we present the related work and problems. In Section 3, the design of the Adaptive Inter-platform Communication Mechanism is described. The implementation is presented in Section 4. Moreover, experiments and evaluation are presented in Section 5. Finally, we conclude this paper in Section 6.

2. Related Work and Problems

2.1 Ubiquitous Application Based on Agent-based Middleware

There is a previous work that has investigated middleware specialized for developing ubiquitous applications [6]. In the literature, the following items are pointed out as the functional requirements for middleware for ubiquitous applications:

(R1) Adaptability to changing environment
(R2) Flexibility to compose application dynamically
(R3) Cooperativeness to share information between various applications

On the other hand, research and development of a variety of agent-based middleware have been promoted in recent years [7)–14)]. Agent-based middleware is a software infrastructure to construct multiagent systems. Here, an agent is made from a software component that is an element of an application system. It is formed by adding such abilities as autonomy, mobility, organizational behavior, and cooperativeness, to the software component. A multiagent system comprises the organization of multiple agents based on the ability of each agent described above. The multiagent system has a characteristic to adapt to various situations by changing the combination of agents. Therefore, the system can be constructed flexibly according to the situation. For these reasons, a multiagent system can fulfill the functional requirements (R1) to (R3), and it would be suitable to construct ubiquitous applications as a multiagent system.

2.2 Difficulties

We have difficulties when we apply traditional agent-based middleware to a ubicomp environment. Here, we assume a ubicomp environment as a computing environment where various types of computers, such as embedded PC, PDA, and handheld PC, with heterogeneous functions and performances, co-exist. They are also connected by wireless access networks in the range from 128 kpbs (PHS) to several tens of Mbps (IEEE802.11). Therefore, the ubiquitous application is supposed to be constructed on a computer with low performance and narrow bandwidth network connection. In such an environment, agent-based middleware, that can continuously provide services as stable as possible, is required, even when QoS is deteriorated or the operational situation of the system becomes unstable. Concretely, we have the following difficulties when agent-based middleware is applied to a ubicomp environment.

(P1) Instability and inefficiency of communication among entities: This is mainly caused by the restriction of network resources,
(P2) Performance degradation: This is due to the limitation of the computational resources and the excessive load of agents,
(P3) Low scalability: This is a problem of low extensibility in terms of the function and size of the overall system, originating from (P1) and (P2),
(P4) Complexity in deployment: This is an operational issue that is how to deploy the agent platform on heterogeneous computing environments.

2.3 Target Problem

In this paper, we focus on problem (P1) described in the previous section. We concentrate on the agent platform that is a workspace where the agents live. The main function of the agent platform is to manage agents' life cycles. It also has an important role supporting inter-agent communications. When an agent sends a message to another agent in a remote host, the agent sends the message to the local agent platform, and then the platform transfers the message to the agent platform in other hosts where the destination agent exists. Finally, the remote agent platform passes the message to the destination agent.

In a ubicomp environment, because networks with narrow bandwidth and unstable connections are generally used, improving the efficiency and stability of communication between agent platforms (inter-platform communication) is es-

Table 1 Inter-platform communication schemes of typical agent-based middleware [18].

Developer	Agent-based middleware	Used communication schemes
Chiba Inst. of Tech.	DASH [9]	ACL, RMI
IKV++ Technologies AG	Grasshopper [12]	sockets, RMI, IIOP
University of Minnesota	Ajanta [10]	Java RMI, ATP
TILab	JADE [13]	ACL, RMI
Open Source	FIPA-OS [11]	ACL, IIOP, RMI
British Telecom. Lab	Zeus [14]	KQML, ACL

Table 2 Examples of inter-agent communication semantics.

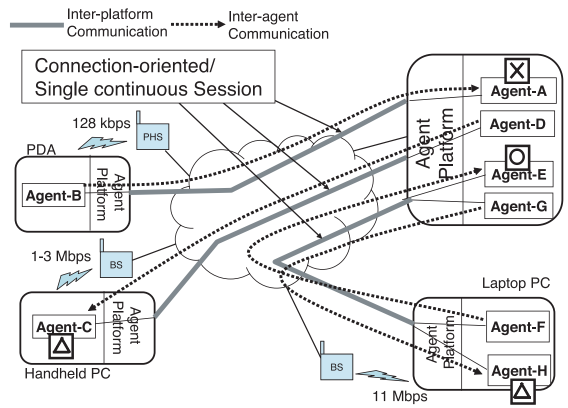
Type	Sender	Recipient	Kind of Communication semantics
(1)	Agent-A	Agent-B	Periodical and continuous report of user's physical location
(2)	Agent-C	Agent-D	Agent termination request
(3)	Agent-E	Agent-F	Transfer of operational history after the service completes
(4)	Agent-G	Agent-H	Transmission of contract net protocol message

sential. However, existing agent-based middleware has limitations resolving this problem. This is because, one and only one communication scheme is statically used in the inter-platform communication even if the types of the communication environments are different. We show the typical agent-based middleware and those inter-platform communication schemes in **Table 1**.

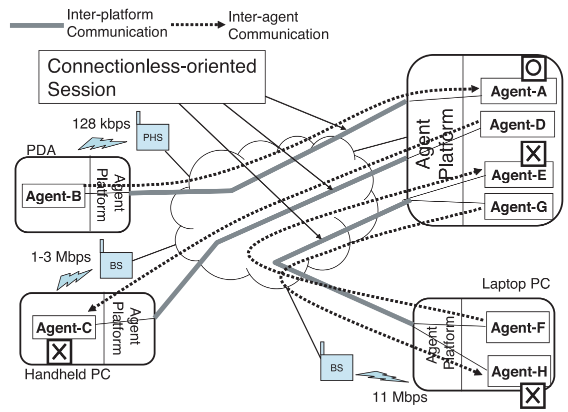
In **Table 2**, we show some examples of semantics of inter-agent communication. Suppose that these different types of communications between agents are performed over the single communication scheme between agent platforms. We show the specific examples in **Fig. 1** to clarify this problem.

Figure 1 (a) represents a case of inter-platform communication by using connection-oriented transport service with single continuous session. The agent-based middleware with this communication scheme has an advantage when reliable communication between agents is required, and the wide bandwidth of network can be available. However, for the inter-agent communication which needs real-time transfer such as delivering of user's location information, and if the network connection is unstable, then the requirement can not be fulfilled due to the transmission delay. In this case, the communication type (3) in Table 2 is suitable, but (1) is not suitable.

On the other hand, Fig. 1 (b) depicts the inter-platform communication scheme



(a) Connection-oriented, single continuous session communication scheme



(b) Connectionless-oriented communication scheme

Fig. 1 Issues in inter-platform communication by using single communication scheme.

by using a connectionless-oriented transport service. In this case, it has an advantage when the inter-agent communication requires real-time transmission rather than the reliability. By contrast, when the reliability is needed such as the case of agent termination instruction is issued, some problems may occur in cooperative behavior of agents due to no arrival of the important message. In this case, (1) is suitable, but other types are not suitable.

These examples show that, existing inter-platform communication with single communication scheme can not achieve the improvement in efficiency and stability of the inter-agent communication in ubicomp environment where the absolute amount of network resources and computer resources are restricted and the change of them are drastic.

3. Adaptive Inter-platform Communication Mechanism

3.1 Outline of the Proposed Mechanism

To resolve the problem described in Section 2.3, a dedicated mechanism is required to switch many types of inter-platform communication schemes according to the information of ubicomp environment in which the agent-based middleware works, and the semantics of the inter-agent communications. Because the inter-agent communication semantics represent requirements of agents for their communications, it provides the important information for selection of the suitable communication scheme. In general, agent-based applications are constructed extemporarily and dynamically on the middleware, thus, the communication semantics also change drastically according to the agent organizations. Therefore, we have to embed the monitoring and analyzing function for the inter-agent communication semantics in the proposed mechanism. This is the agent-specific function required for the proposed mechanism. Fortunately, all the inter-agent communications are basically performed through the platform, and the messages used for the communications are formalized by the agent communication protocols, so we can manage the communications intensively and systematically. We utilize these agent-specific characteristics to realize the inter-agent communication monitoring and analyzing.

Figure 2 shows the effect of the proposed mechanism. The assumed inter-agent communication semantics are the same as those defined in Table 2. This mechanism selects the communication scheme of low transmission delay when the inter-agent communication requires real-time property. Whereas, it offers the communication scheme with reliability when the agent needs the reliable communication instead of the real-time property. Therefore, the agent platform can be achieved, with the flexible inter-platform communication which meets the requirements of the inter-agent communications as much as possible.

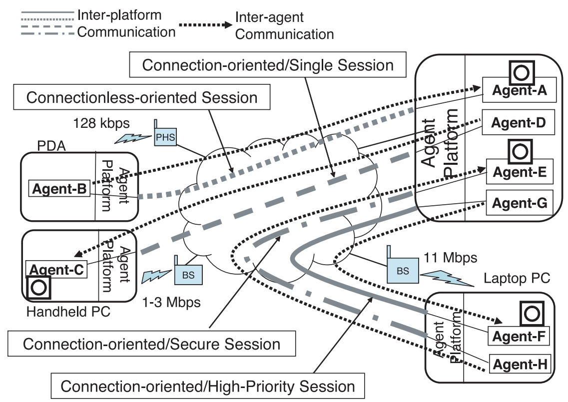


Fig. 2 An examples of the appropriate selection of the inter-platform communication scheme based on the inter-agent communication semantics.

We propose an "Adaptive Inter-platform Communication Mechanism." This mechanism has functions to select a suitable inter-agent platform communication scheme based on the situations of network/computational resources and the properties of inter-agent communications. The inter-platform communication scheme M is represented by the five-tuple as follows:

$$M = < c, a, b, p, s >$$

Here, c represents a connection type of the transport communication such as connectionless-oriented or connection-oriented; a shows a property of duration time of a transport session such as short, long, or continuous session; b expresses a kind of data transmission scheme such as streaming and bulk data transfer; p means a priority of an agent message such as high, medium, low, and urgent; s is a security property such as encrypted or plain. The proposed mechanism outputs the selected communication scheme in the form of this model.

3.2 Architectural Design of the Proposed Mechanism

Figure 3 shows the architectural design of Adaptive Inter-platform Communication Mechanism. This architecture consists of six functions as follows:

(1) Computational Resource Monitor: This function is for observation of the computational resources of the computer on which the middleware runs, such as CPU and memory utility status. It stores and manages the data of utilization

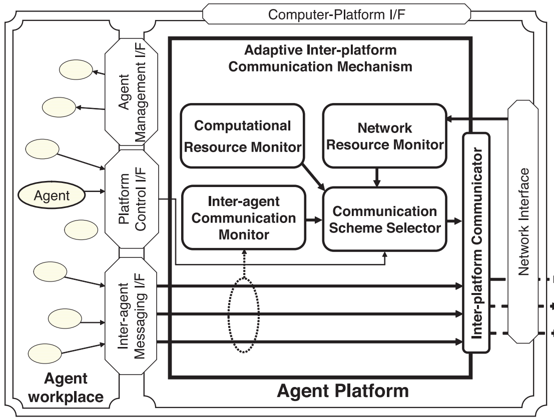


Fig. 3 Architecture of Adaptive Inter-platform Communication Mechanism.

of CPU, memory or other resources obtained from computer platform interfaces.

(2) Network Resource Monitor: This function is for observation of the network resources. It manages the network-related data obtained from network interface, such as the type of communication link, available or utilizing network bandwidth, amount of network traffic, or status of the link (connected, disconnected, and re-connecting), etc.

(3) Inter-agent Communication Monitor: This function is for observation of inter-agent communication semantics. It infers the characteristics of the inter-agent communication based on header information, message contents, intervals, frequency, a pattern of message exchanging between the agents, etc. For instance, when an agent sends messages to another specific agent in a particular interval, and only the fixed position of the contents in the sequence of the agent messages is just difference, this function recognizes that the agent sends some kinds of resource status information periodically, such as transition of computational resources and user's location. This is the agent-specific function to trace the inter-agent communication semantics that change drastically according to the dynamic construction of agent organization.

(4) Communication Scheme Selector: This is a key component in the proposed mechanism; this makes a decision about suitable inter-platform communication scheme (M) based on the information from all the monitoring functions (Computational Resource Monitor, Network Resource Monitor, and Inter-agent Communication Monitor). In this function, the decision making is accomplished by the production system for internal processing. Details are described in Section 3.3.

(5) Inter-platform Communicator: This function selects one communication scheme that satisfies the suggestion from the Communication Scheme Selector, and maintains the mapping a specific inter-agent communication message onto a selected transport communication scheme.

(6) Network Interface: This function actually establishes and maintains the transport communication path between agent platforms and transfers agent messages through the path. It also gives information about network resource situations to Network Resource Monitor.

3.3 Internal Structure of Communication Scheme Selector

Information that is given to the Communication Scheme Selector from various monitoring functions is diverse. If the function for decision making on appropriate communication scheme is written by the procedural type programming language, there will be problems in its description and readability, because we have to code many condition branches. The system also lacks scalability because the algorithms are hard-coded. In this work, we propose the design of this function with the "production system" to resolve these problems. The employment of the production system enables the stepwise refinement of the function by adding the "rules" and "facts." Moreover, it enables intuitive and immediate description of the process of this function because we can describe the operations on the information as knowledge in the if-then rule fashion. In this knowledge, the observed information is described in the conditional part, and the selected communication scheme is described in the action part, in the rule description.

The internal structure of this function is shown in **Fig. 4**. Working Memory (WM) stores a set of facts. A fact represents a situation of a thing using a set of pairs of attribute and value. The Rule Base stores a set of rules that are described in if-then fashion. Inference Engine checks whether the conditional part of a rule in the Rule Base and a fact in the WM are matched. If all the conditions for the rule are matched to the facts in the WM, the rule is fired and the action

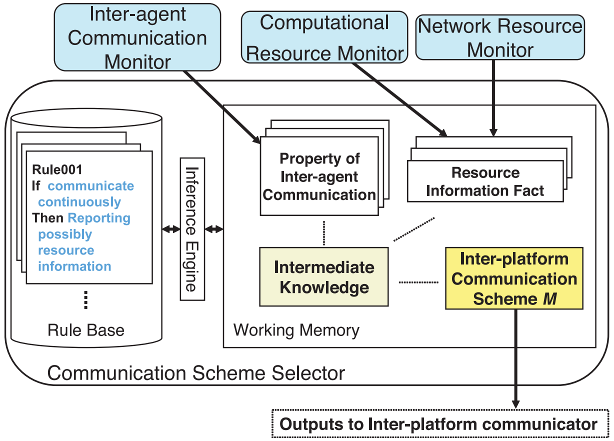


Fig. 4 Internal structure of Communication Scheme Selector based on production system.

part of the rule is executed by the specified sequence of operations such as creation, modification, and deletion. In this function, the fact represents resource situations obtained from various monitoring functions, information about characteristics of the inter-agent communications, the intermediate knowledge used in the process of inference, the final inference result, etc. Finally the rule derives a suitable communication scheme M according to these situations.

Here, we mention that how the rule-based system can resolve the problems in efficiency and stability described in (P1) in Section 2.2. In ubicomp environment, many types of access networks and computation devices are co-existing. Therefore, we have to cope with the knowledge on the wide range of available network resources and computational resources, to maintain the efficiency and stability of inter-agent communication. For instance, we can use many types of the access networks such as Zigbee, Bluetooth, Wi-Fi, WiMAX, PHS, FOMA, etc. Each access networks have individual characteristics in their communication, thus we have to deal with many kinds of knowledge on these networks. Moreover, new technologies are incorporated day by day in the ubicomp environment, so when a new access network or sensor device is installed, our system should include it quickly and easily. Therefore, our system should have ability to deal with the complexity and scalability of the knowledge on networks and computational devices. The rule-base system meets these requirements. The rules and facts represent the segment of the knowledge compactly and they increase the maintainability and extensibility of the total system. Moreover, the Inference Engine can induce the best possible solution from many patterns of complex conditions to recover the undesired situation in efficiency and stability.

3.4 An Example Operation of the Communication Scheme Selector

In this section, we show how the Communication Scheme Selector works, by illustrating a simple example of the scheme selection. Here, a fact is generally represented in the following form:

$$(identifier\ :attribute\ [value]\ \ldots) \tag{1}$$

The structure of intermediate knowledge is generally represented in the following form:

$$\begin{aligned}&(inference - result\\ &\quad :communication\ [communication\ endpoint]\\ &\quad :positive\ [positive\ suggestion\ of\ M]\\ &\quad :negative\ [negative\ suggestion\ of\ M]\\ &)\end{aligned} \tag{2}$$

The : *communication* attribute indicates a communication endpoint of inter-agent communication, the : *positive* attribute is a positive suggestion, and the : *negative* attribute is a negative suggestion. We describe an example of operation of the Communication Scheme Selector. First, the Network Resource Monitor function observes a situation: "The Round Trip Time (RTT) between this machine and a remote machine X is 2,500 milliseconds." In addition, Inter-agent Communication Monitor function observes a situation: "Agent A is communicating with Agent B in a remote machine X at interval of 10,000 milliseconds." Hereby, the following facts are created in Working Memory (WM).

$$\begin{aligned}&(ping\text{-}update\\ &\quad :remote\text{-}address\ X\ \ :rtt\text{-}millisec\ 2{,}500\\ &)\end{aligned} \tag{3}$$

$$\begin{aligned}
&(\textit{agent-outgoing-communication}\\
&\quad :\textit{from}\ A\ \ :\textit{to}\ B\\
&\quad :\textit{remote-address}\ X\ \ :\textit{remote-platform}\ P\\
&\quad :\textit{frequency-millisec}\ 10{,}000\\
&)
\end{aligned}\tag{4}$$

Next, for example, we assume that a threshold value of the RTT is set to 2,000 milliseconds. This threshold value is used for determining whether a situation of network is overflow or not. We also assume that a threshold value of transmission of inter-agent communication interval is set to 10,000 milliseconds. This value is used for determining whether a situation of inter-agent communication is high-frequency or not. Hereby, the following facts are created in WM.

$$(\textit{rtt-threshold}\ \ :\textit{threshold}\ 2{,}000)\tag{5}$$

$$(\textit{frequency-threshold}\ \ :\textit{threshold}\ 1{,}000)\tag{6}$$

Consequently, from Fact (3) and Fact (5), the Inference Engine infers the following intermediate knowledge. This knowledge means that it is feasible to use unreliable communication scheme and not to use reliable communication scheme for communications to X.

$$\begin{aligned}
&(\textit{inference-result}\\
&\quad :\textit{communication}\\
&\qquad (\textit{link}\ \ :\textit{from}\ *\ \ :\textit{to}\ *\ \ :\textit{remote-address}\ X\ \ :\textit{remote-platform}\ *)\\
&\quad :\textit{positive}\\
&\qquad (\textit{suggestion}\\
&\qquad\quad :\textit{guarantee-type}\ (\textit{UnReliable})\ \ldots)\\
&\quad :\textit{negative}\\
&\qquad (\textit{suggestion}\\
&\qquad\quad :\textit{guarantee-type}\ (\textit{ReliableArriveOne})\ \ldots)\\
&)
\end{aligned}\tag{7}$$

In the same way, the Inference Engine infers the following intermediate knowledge from Fact (4) and Fact (6). This knowledge means that it is feasible to use reliable communication scheme.

$$\begin{aligned}
&(\textit{inference-result}\\
&\quad :\textit{communication}\\
&\qquad (\textit{link}\ \ :\textit{from}\ A\ \ :\textit{to}\ B\ \ :\textit{remote-address}\ X\\
&\qquad\quad :\textit{remote-platform}\ P)\\
&\quad :\textit{positive}\\
&\qquad (\textit{suggestion}\\
&\qquad\quad :\textit{guarantee-type}\ (\textit{ReliableArriveOne})\ \ldots)\\
&\quad :\textit{negative}\\
&\qquad (\textit{suggestion}\\
&\qquad\quad :\textit{guarantee-type}\ ()\ \ldots)\\
&)
\end{aligned}\tag{8}$$

In addition, the Network Resource Monitor function observes a situation; reliable communication scheme and unreliable communication scheme are available for all communications. The Inference Engine infers the following intermediate knowledge.

$$\begin{aligned}
&(\textit{inference-result}\\
&\quad :\textit{communication}\\
&\qquad (\textit{link}\ \ :\textit{from}\ *\ \ :\textit{to}\ *\ \ :\textit{remote-address}\ *\ \ :\textit{remote-platform}\ *)\\
&\quad :\textit{positive}\\
&\qquad (\textit{suggestion}\\
&\qquad\quad :\textit{guarantee-type}\ (\textit{ReliableArriveOne}\ \textit{NotReliable})\ \ldots)\\
&\quad :\textit{negative}\\
&\qquad (\textit{suggestion}\\
&\qquad\quad :\textit{guarantee-type}\ (\textit{ReliableArriveSome})\ \ldots)\\
&)
\end{aligned}\tag{9}$$

The : *communication* attributes of Fact (7), Fact (8), and Fact (9) are corresponding to the communication of Agent A and Agent B. Therefore, the Communication Scheme Selector summarizes these facts to the following fact.

$$
\begin{array}{l}
(\mathit{suggestion\text{-}set} \\
\quad :\mathit{communication} \\
\quad\quad (\mathit{link} \\
\quad\quad\quad :\mathit{from}\ A\ \ :\mathit{to}\ B \\
\quad\quad\quad :\mathit{remote\text{-}address}\ X\ \ :\mathit{remote\text{-}platform}\ P) \\
\quad :\mathit{positive} \\
\quad\quad (\mathit{suggestion} \\
\quad\quad\quad :\mathit{guarantee\text{-}type} \\
\quad\quad\quad\quad (\mathit{NotReliable}\ \mathit{ReliableArriveOne})\ \ldots) \\
\quad :\mathit{negative} \\
\quad\quad (\mathit{suggestion} \\
\quad\quad\quad :\mathit{guarantee\text{-}type} \\
\quad\quad\quad\quad (\mathit{ReliableArriveOne}\ \mathit{ReliableArriveSome})\ \ldots) \\
)
\end{array}
\tag{10}
$$

Fact (10) shows an inference result from intermediate knowledge. This result denotes when Agent A communicates with Agent B in platform P in machine X, *NotReliable* or *ReliableArriveOne* is feasible to communication scheme, but *ReliableArriveOne* and *ReliableArriveSome* are not feasible. Finally, the Inference Engine subtracts the : *negative* from the : *positive* of Fact (10), and the following fact led as a final inference result M.

$$
\begin{array}{l}
(M\ \ :\mathit{communication} \\
\quad :(\mathit{link}\ \ :\mathit{from}\ A\ \ :\mathit{to}\ B\ \ :\mathit{remote\text{-}address}\ X \\
\quad\quad\quad :\mathit{remote\text{-}platform}\ P) \\
\quad :\mathit{guarantee\text{-}type}\ \mathit{NotReliable} \\
\quad \ldots)
\end{array}
\tag{11}
$$

Fact (11) draws an inference result. This means to use not reliable communication scheme when Agent A communicates with Agent B in platform P in machine X.

4. Implementation

We have been developing the proposed mechanism based on the architecture described in Section 3. We employed DASH [8] for software infrastructure. In this implementation, we have incorporated the mechanism onto the inter-platform communication function in DASH framework that is a kind of agent-based middleware.

First, we enhanced the Network Interface in the prototype system to be able to connect with connection-oriented transport communication (TCP) and connectionless-oriented transport communication (UDP). In addition, we introduced a new function to the agent platform to provide the network resource information to the Network Resource Monitor. By using the characteristics of simple connection-oriented transport protocol (TCP), we added a function to a receiver to return an acknowledgement packet to a transmitter. The function detects whether the network congestion or not based on a delay time of acknowledgement packets. Furthermore, we added a re-connection function by utilizing this function. This function disconnects the communication path when the network congestion goes beyond the specific threshold, and then tries to reconnect the path after waiting for a fixed time interval. This threshold level and reconnection waiting time can be set dynamically from the agent application.

Second, we modified the Platform Control Interface such that the agents can specify an Inter-platform Communication Scheme M directly without any control of the Communication Scheme Selector.

Third, we implemented the Communication Scheme Selector. We constructed very simple production system to confirm the effectiveness of the proposal mechanism. Besides, the production system has only one rule that represents "All agents request a small delay communication scheme without any consideration of reliability." Thus, the Communication Scheme Selector can decide the M as connection-oriented communication when the network condition is stable, and it decides the M as connectionless-oriented communication when the network has congestion.

Finally, we implemented a Network Resource Monitor. This monitoring function enables for the Communication Scheme Selector to select the appropriate scheme based on the network status.

5. Experiment and Evaluation

5.1 Experimental Environment and Scenario

We performed experiments by using the prototype system to confirm the effectiveness of the proposed mechanism on a data streaming service with two types of access network. To compare to a traditional inter-platform communication mechanism, we implemented a single inter-platform communication scheme which can communicate by connection-oriented communication to the prototype system. The experimental environment consists of two PCs as shown in **Fig. 5**. The streaming transmission server consists of "Transmitter" agent and "Manager" agent. The Transmitter agent transmits a streaming data by UDP to "Receiver" agent in receiving terminal, and the Manager agent changes the streaming data rate. On the other hand, the receiving terminal includes "Receiver" agent and "Controller" agent. The Controller agent continuously sends a message for controlling transmission rate to the Manager agent by inter-agent communication. If the network resources are degraded, the QoS of the streaming service is degraded greatly because the network resources are shared by the streaming service and controlling its rate. Thus, the application system falls into uncontrollable or halts, and the QoS is degraded greatly.

We performed two experiments by using two types of access network of receiving terminal; one is on PHS access network (Exp.1) and another is on FOMA access network (Exp.2). On the both access networks, we compare the efficiency and stability of the inter-platform communication with the proposed mechanism and the traditional mechanism, in order to show how much the problem (P1) in Section 2.2 is resolved.

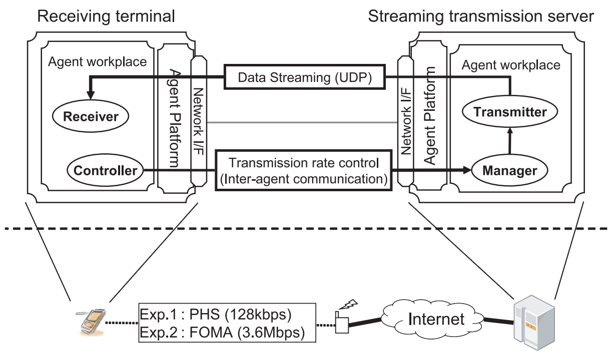


Fig. 5 Experimental environment.

We define the "Efficiency" of the inter-platform communication as the response performance of the data streaming service. This response performance is denoted as *ResponseTime*, and represented by the duration between the time Tc when the Controller agent sends a transmission rate control message and the time Ts when the Receiver agent receives a data which reflects the rate control message, as follows:

$$ResponseTime = Ts - Tc \text{ [sec.]}$$

Tc: The time when the Controller agent sends a transmission rate control message
Ts: The time when the Receiver agent receives a data which reflects the rate

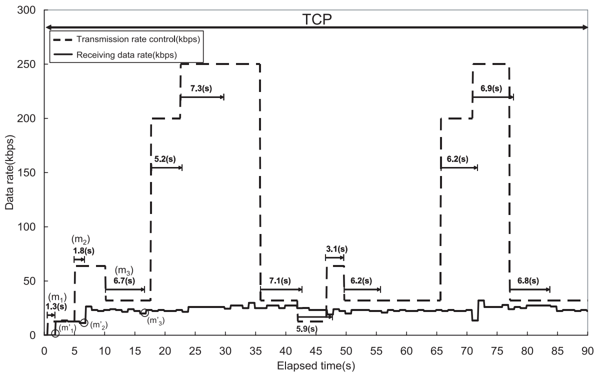
We also define the "Stability" of the inter-platform communication as the variance of the *ResponseTimes*. This stability can be measured by the standard deviation of a set of the *ResponseTimes*.

We performed the two experiments by the following scenarios. In the Exp.1, the Controller agent sends the transmission rate control messages in order of 12.8 kbps, 64 kbps, 32 kbps, 200 kbps, 250 kbps, and 32 kbps in every 5 seconds to the Manager agent. On the other hand, in Exp.2, the Controller agent sends the transmission rate control messages in order of 10 kbps, 90 kbps, 10 kbps, 180 kbps, 10 kbps, and 360 kbps in every 5 seconds to the Manager agent. In addition, we set the threshold value of round trip time of acknowledge packet to 2 seconds and the reconnecting waiting time to 30 seconds. Moreover, we measured the *ResponseTime* in 90 seconds on the receiving terminal.

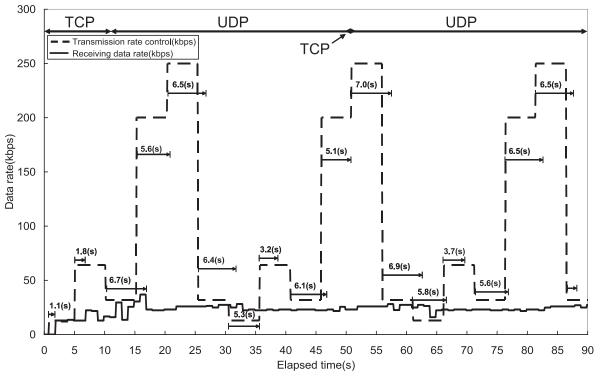
5.2 Experimental Results

The experimental results are shown in **Fig. 6** and **Fig. 7**. Figure 6 (a) and Fig. 7 (a) are the results of the traditional inter-platform communication scheme, and Fig. 6 (b) and Fig. 7 (b) are the results of the proposed mechanism.

In all results, the horizontal axis is measured time, and the vertical axis is data rate (kbps). The broken line drawn means the transmission data rate control, and a solid line means the received data rate. Circles on the solid line show the time when the received data rate is actually changed to the value specified by

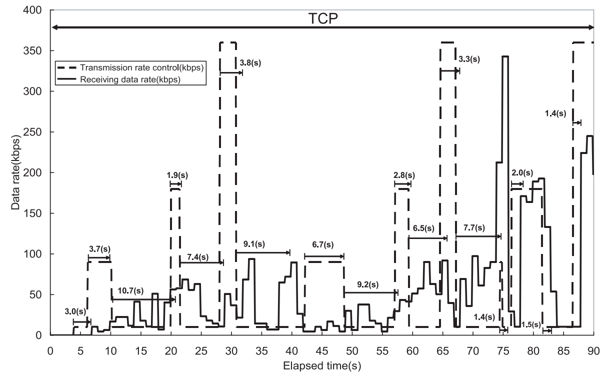


(a) A result by using the fixed communication scheme (Connection-oriented)

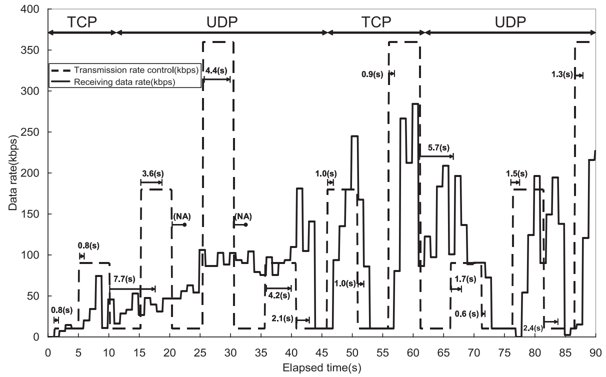


(b) A result by using the proposed mechanism

Fig. 6 Experimental results of Exp.1.



(a) A result by using the fixed communication scheme (Connection-oriented)



(b) A result by using the proposed mechanism

Fig. 7 Experimental result of Exp.2.

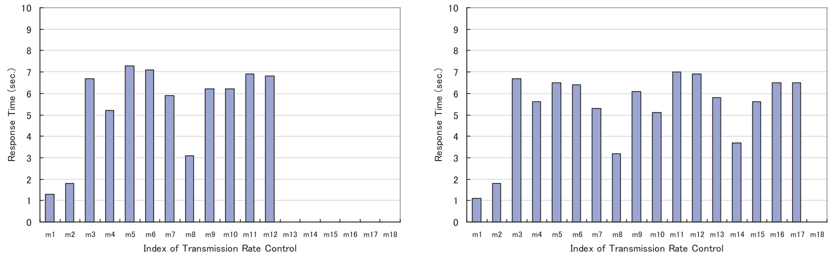
transmission rate control just before. For instance, the change of received data rate in the Fig. 6 (a) at point (m_1') is caused by the transmission rate control at point (m_1), and also change at (m_2') is caused by (m_2) and at (m_3') is caused by at (m_3), respectively. Moreover, the value such as 1.3 seconds or 1.8 seconds on Fig. 6 (a) represents the *ResponseTime*. "TCP" and "UDP" means the communication scheme used at the corresponding time slot.

First, we describe the result of traditional mechanism with PHS shown in Fig. 6 (a). The *ResponseTime* at 10 seconds was 6.7 seconds. At this time the situation of access network was overloaded and then the incoming streaming data to receiving terminal was delaying. This situation was continued for 37 seconds, but at 47 seconds the *ResponseTime* was decreased to 3.1 seconds, so the situation of network was turned around temporary at this time. However, at 50 seconds the *ResponseTime* was increasing and the network condition was degraded again. Additionally, the transmission intervals of transmission rate control message were delaying largely at 10 seconds, 23 seconds, and 50 seconds. This is occurred because the inter-platform communication scheme of traditional mechanism transmits an agent message after receiving acknowledgement packet. Therefore, the traditional mechanism leads degradation of QoS such as uncontrollable of data streaming service or decreasing frames per second (FPS) of video transmission service.

Second, we describe the result of proposed mechanism with PHS shown in Fig. 6 (b). At 10 seconds, the *ResponseTime* was 6.7 seconds and the network condition was degraded similarly at the time of Fig. 6 (a). In this case, the inter-platform communication scheme of proposed mechanism was switched from connection-oriented (TCP) to connectionless-oriented (UDP) communication scheme. Hereby we confirmed that the Network Resource Monitor function (described in Section 3.2) was running successfully, and therefore the *ResponseTime* at 20 seconds and 25 seconds was better than that of the traditional mechanism. In addition, at 52 seconds the inter-platform communication scheme switched to TCP because the reconnecting waiting time had passed, but the inter-platform communication scheme returned to UDP soon because the network congestion was continued. Therefore, the proposed mechanism can prevent degradation of QoS by improving the inter-agent communication by switching the inter-platform communication scheme according to the network situation.

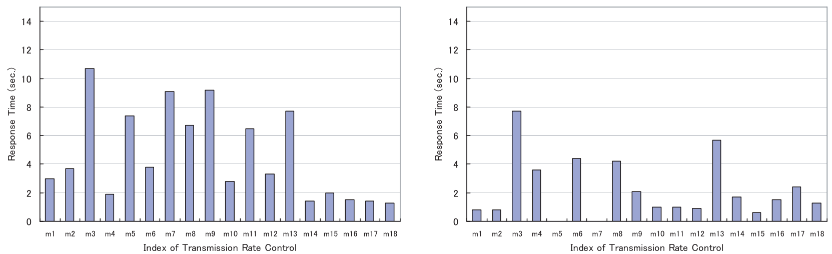
Third, we describe the result of traditional mechanism with FOMA shown in Fig. 7 (a). At 10 seconds the *ResponseTime* increased from 3.7 seconds to 10.7 seconds and the network condition was degraded. The nominal downstream bandwidth of FOMA is 3.6 Mbps but the receiving data rate was less than the 50 kbps. This means that the actual network situation can not determine beforehand. This is a good example of unexpected situation on the ubicomp environments. After this, the *ResponseTime* was not stable and also the receiving data rate was drastically changing. Hence, the traditional mechanism can not control the services stably, because the transmission of the inter-agent communication was delaying.

Fourth, we describe the result of proposed mechanism with FOMA shown in Fig. 7 (b). We confirmed that the mechanism switched inter-platform communication scheme from TCP to UDP at 10 seconds. As a result, the agent message was transmitting continuously. Additionally, at 20 seconds and 30 seconds, the rate control message was transmitted but missed to reflect to the receiving data. This



(a) A result by using the fixed communication scheme (b) A result by using the proposed mechanism

Fig. 8 Experimental result of Exp.1.



(a) A result by using the fixed communication scheme (b) A result by using the proposed mechanism

Fig. 9 Experimental result of Exp.2.

occurred because the system prioritized the real-time property rather than the reliability. Hence, the proposed mechanism controls QoS stably by improving the inter-agent communication by switching the inter-platform communication scheme. In addition, the inter-platform communication switched from UDP to TCP like a result by the proposed mechanism of Exp.1. Moreover, because the Network Resource Monitor function of proposed mechanism observed precisely what the network congestion improved at 45 seconds, 50 seconds, and 56 seconds, thereby the *ResponseTime* at the time was decreased.

Here, we show the *ResponseTime* of each Transmission Rate Control Message in both experiments, to make the effect of the proposed mechanism clear, in **Fig. 8** and **Fig. 9**. Figure 8 represents the result of Exp.1, and Fig. 9 is for the Exp.2. In

Table 3 Evaluation of the two experiments.

(a) Experimental results of Exp.1

	Traditional Mechanism	Proposed Mechanism
Average *ResponseTime*	5.4 (s)	5.3 (s)
SD of *ResponseTime*	2.0	1.7

(b) Experimental results of Exp.2

	Traditional Mechanism	Proposed Mechanism
Average *ResponseTime*	4.6 (s)	2.5 (s)
SD of *ResponseTime*	3.1	2.0

Table 4 Evaluation of 20 times experiments.

(a) Results of 20 times of Exp.1

	Traditional Mechanism	Proposed Mechanism
Average *ResponseTime*	5.1 (s)	5.5 (s)
SD of *ResponseTime*	5.2	3.7

(b) Results of 20 times of Exp.2

	Traditional Mechanism	Proposed Mechanism
Average *ResponseTime*	5.6 (s)	4.6 (s)
SD of *ResponseTime*	8.2	7.0

the graphs of Fig. 8 and Fig. 9, the horizontal axis represents the sequence number of the Transmission Rate Control Message in chronological order with index of m_i ($i = 1, 2, 3 \cdots$ n), and the vertical axis represents the *ResponseTime.* After the m_{13} in Fig. 8 (a) and after m_{18} in Fig. 8 (b) are not shown, because these messages are beyond the experimental duration of 90 seconds. From these results, by comparing the number of times of successful transmission rate control, proposed mechanism is greater than the traditional mechanism. This phenomenon presents the fact that the proposed mechanism can control the service surely, compare to the traditional mechanism. Moreover, in Exp.2 with FOMA access network, we can find that the *ResponseTime* of the proposed mechanism is considerably smaller that the traditional mechanism totally.

Subsequently, we summarize the average of *ResponseTime* and standard deviation (SD) in each experiment in **Table 3**. From the results of Exp.1 in Table 3 (a), the average *ResponseTime* of the proposed mechanism is increased and the SD of *ResponseTime* is also increased. The efficiency of the proposed mechanism is 2% better and the stability of the proposed mechanism is 15% better than that of traditional mechanism. In a similar way, the results of Exp.2 in Table 3 (b), the average *ResponseTime* and the SD of the *ResponseTime* is increased, and the efficiency of the proposed mechanism is 46% better and the stability of the proposed mechanism is 35% better than that of traditional mechanism. These results show that the proposed mechanism will have good performance on the access networks with broader bandwidth.

Finally, in **Table 4**, we show the efficiency and stability improvement in more general way. We performed 20 times experiments with the same situations of Exp.1 and Exp.2, and measured the total efficiency and reproducibility (i.e., stability) of the results. We show the standard deviation of the averages of *ResponseTime* in 20 times trials of Exp.1 and Exp.2. We also show the average of the averages of *ResponseTime* in 20 times trials of Exp.1 and Exp.2. Herewith, the average of *ResponseTime* of the proposed mechanism is 6% smaller than that of the traditional mechanism, and the SD of the averages of *ResponseTime* in proposed mechanism is 15% to 29% smaller than that of traditional mechanism. Therefore, from the experimental results, it is clear that the efficiency of the inter-platform communication is improved 6% and the stability is improved 22%, compare to those of the traditional mechanism, totally.

5.3 Discussion

The Controller agent sends a control message to the Manager agent to change the streaming data rate. This kind of service controlling messages influence greatly to the performance of the service. In this situation, the inter-platform communication is required "Reliable" communication, so the traditional agent-based middleware uses the connection-oriented communication scheme. However, our proposed mechanism selected connectionless-oriented communication scheme and it worked effectively.

This selection was inferred by the following knowledge which is embedded in the Communication Scheme Selector function of proposed mechanism:

(K1) "The Network is TCP/IP-based network."

(K2) "UDP can be transferred preferentially than TCP on the TCP/IP-based network."

(K3) "Use the Inter-platform Communication according to the actual network situation."

Moreover, the situation of the environment are following:

(S1) "The network state is congestive."

(S2) "Agents require reliable communication scheme."

Here, (S1) is obtained from the Network Resource Monitor function and (S2) from the Inter-agent Communication Monitor.

In this case, the Communication Scheme Selector inferred as follows:

(F1) "Use TCP" from (S1)

(F2) "Use UDP" from (K1) and (K2) and (S1)

(F3) "Use UDP" from (K3) and (F1) and (F2)

From this inference, UDP was selected for the inter-platform communication scheme in experiments that used proposed mechanism when the network resources are degraded. As a result, QoS of streaming service is improved without depending on the kind of access network in both experiments when used the proposed mechanism. We think the prototype system is suitable for the practical use because of the time cost for switching the inter-platform communication scheme is less than 1 millisecond.

Hence, we confirmed that the agent-based middleware can provide stable and high-quality services by switching adaptively the inter-platform communication scheme according to the situation of environment.

6. Conclusion

In this paper, we proposed an adaptive communication mechanism between agent platforms which can flexibly select communication schemes based on the properties of inter-agent communication, in order to achieve stable ubiquitous services by using agent-based middleware. We described the design of the Adaptive Inter-platform Communication Mechanism and implemented a prototype system which has a part of the proposed functions. We performed experiments using the prototype system under the situation which assumed ubicomp environment to evaluate the effectiveness of the proposed mechanism. In the network congestion, we confirmed that the proposed mechanism can improve QoS of ubiquitous services. In fact, from the experimental results, it is clear that the efficiency of the application is improved 5% and the stability of the application is improved 22%, compare to these of the traditional mechanism. We expect that our mechanism can improve the efficiency and stability of the ubiquitous applications, especially for the applications with real-time property.

In future work, we will continuously enhance the functions of the prototype system, such as the Inter-agent Communication Monitor, Communication Scheme Selector, and inference engine of the internal processing, to archive the middleware that can adapt to various inter-agent communications. Because the generality of this mechanism in various application domains is not deeply considered in this paper, we will apply this mechanism to the different applications, and will measure the effective range in efficiency and the stability in general ubiquitous applications. We will also apply this mechanism to the other practical applications to investigate the effects in the development, to have knowledge on design methodology of agent-based ubiquitous applications.

References

1) Weiser, M.: The Computer for the Twenty-first Century, *Sci. Am.*, Vol.265, No.3, pp.94–104 (1991).

2) Inoue, M., Mahmud, K., Murakami, H., Hasegawa, M. and Morikawa, H.: Context-Based Network and Application Management on Seamless Networking Platform, *Wireless Personal Communications*, Vol.35, No.1-2, pp.53–70 (2005).

3) Minoh, M. and Kamae, T.: Networked Appliance and Their Peer-to-Peer Architecture AMIDEN, *IEEE Comm. Magazine*, Vol.39, No.10, pp.80–84 (2001).

4) Gribble, S., Welsh, M., Behren, R., et al.: The Ninja architecture for robust Internet-scale systems and services, *Special Issue of Computer Networks on Pervasive Computing*, Vol.35, No.4, pp.473–497 (2001).

5) Masuoka, R., Parsia, B. and Labrou, Y.: Task Computing – The Semantic Web Meets Pervasive Computing, *Proc. 2nd International Semantic Web Conference* (*ISWC2003*), Vol.LNCS 2870, pp.866–881 (2003).

6) Grimm, R., Davis, J., Lamar, E., MacBeth, A., Swanson, S., Anderson, T., Bershad, B., Bborriello, G., Gribble, S. and Wetherall, D.: System Support for Pervasive Applications, *Proc. ACM Trans. Comput. Syst.*, Vol.22, No.4, pp.421–486 (2004).

7) Kawamura, T., Tahara, Y., Hasegawa, T., Ohsuga, A. and Honiden, S.: Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems, *IEICE Trans. Inf. Syst.*, Vol.J82-D-I, No.9, pp.1165–1180 (1999).

8) Fujita, S., Hara, H., Sugawara, K., Kinoshita, T. and Shiratori, N.: Agent-Based

Design Model of Adaptive Distributed Systems, *Applied Intelligence*, Vol.9, No.1, pp.57–70 (1998).
9) DASH: DASH, DASH – Distributed Agent System based on Hybrid architecture. http://www.agent-town.com/dash/index.html (accessed 2009-05-13).
10) Ajanta: AJANTA: Home Page, AJANTA – Mobile Agents Research Project. http://www.cs.umn.edu/Ajanta/ (accessed 2009-05-13).
11) FIPA-OS: SourceForge.net: FIPA-OS Agent Toolkit, FIPA-OS Agent Toolkit. http://fipa-os.sourceforge.net/ (accessed 2009-05-13).
12) Grasshopper: Grasshopper – A Universal Agent Platform Based on OMG MASIF and FIPA Standards. http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm (accessed 2009-05-13).
13) JADE: Jade – Java Agent DEvelopment Framework. http://jade.cselt.it/ (accessed 2009-05-13).
14) Zeus Agent Toolkit: SourceForge.net: Agent Tool Kit. http://sourceforge.net/projects/zeusagent/ (accessed 2009-05-13).
15) Ito, T., Imai, S., Takahashi, H., Suganuma, T. and Shiratori, N.: Adaptive Inter-platform Communication Mechanism in Agent-based Middleware for Ubiquitous Computing Environments, *Proc. 4th International Conference on Mobile Computing and Ubiquitous Networking* (*ICMU 2008*), pp.25–32 (2008).
16) Ito, T., Takahashi, H., Suganuma, T. and Shiratori, N.: Adaptive Inter-platform Communication Mechanism for Ubiquitous Multiagent System, *Proc. International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS 2009*), pp.193–199 (2009).
17) Ito, T., Takahashi, H., Suganuma, T. and Shiratori, N.: Design and Implementation of Adaptive Communication Mechanism for Ubiquitous Multiagent Systems, *Proc. Multimedia Communications and Distributed Processing Workshop* (*DPSWS 2008*), pp.187–192 (2008).
18) Nguyen, G.T., Dang, T.T., Hluchy, L., Laclavik, M., Balogh, Z. and Budinska, I.: Agent Platform Evaluation and Comparison, Pellucid EU 5FP IST-2001-34519 RTD, Technical Report, pp.1–11 (2002).

(Received May 18, 2009)
(Accepted March 5, 2010)
(Released June 9, 2010)

Editor's Recommendation

This paper proposes the innovative communication mechanism for the adaptive selection of the interaction method on the multi-agent communication environment. The environment is needed to support the adaptive selection of the interaction method when the characteristic of the network environment is dynamically changed. The paper significantly contributes to the research area of multimedia and distribute communication. It been selected as a recommended paper to IPSJ Journal by IPSJ SIG-DPS.

(Takayuki Kushida, Chair of IPSJ SIG-DPS)

Taishi Ito received his M.S. degree in 2009 from Tohoku University, Japan. Currently, he is pursuing his doctoral degree in Graduate School of Information Sciences (GSIS), Tohoku University. His research interests include agent-based framework and its application. He is a student member of IPSJ.

Hideyuki Takahashi is a research fellow of Research Institute of Electrical Communication of Tohoku University, Japan. He received his doctoral degree in Information Sciences from Tohoku University in 2008. His research interests include ubiquitous computing and agent-based computing. He is a member of IPSJ.

Takuo Suganuma is an associate Professor of Research Institute of Electrical Communication of Tohoku University, Japan. He received a Dr. Eng. degree from Chiba Institute of Technology. He received UIC-07 Outstanding Paper Award in 2007, etc. His research interests include agent-based computing, flexible network, and symbiotic computing. He is a member of IEICE, IPSJ and IEEE.

Tetsuo Kinoshita is a Professor of Research Institute of Electrical Communication and Graduate School of Information Science, Tohoku University, Japan. He received his Dr. Eng. degree in information engineering from Tohoku University in 1993. His research interests include agent engineering, knowledge engineering, agent-based systems and knowledge-based systems. He received the IPSJ Research Award, the IPSJ Best Paper Award and the IEICE Achievement Award in 1989, 1997 and 2001, respectively. Dr. Kinoshita is a member of IEEE, ACM, AAAI, IEICE, IPSJ, JSAI, and Society for Cognitive Science of Japan.

Norio Shiratori received his doctoral degree from Tohoku University, Japan in 1977. Presently he is a Professor of the Research Institute of Electrical Communication, Tohoku University. He has been engaged in research related to symbiotic computing paradigms between human and information technology. He was the recipient of the IPSJ Memorial Prize Winning Paper Award in 1985, the Telecommunication Advancement Foundation Incorporation Award in 1991, the Best Paper Award of ICOIN-9 in 1994, the IPSJ Best Paper Award in 1997, the IPSJ Contribution Award in 2007, and many others. He was the vice president of IPSJ in 2002 and now is the president of IPSJ. He is also a fellow of IEEE and IEICE.