IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2017, accepted December 29, 2017, date of publication January 23, 2018, date of current version April 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2796307

Agent-Based System Architecture Supporting Remote Collaboration via an Internet of Multimedia Things Approach

YUKI KAERI[1], (Member, IEEE) CLAUDE MOULIN[2], KENJI SUGAWARA[3], (Member, IEEE), AND YUSUKE MANABE[3], (Member, IEEE)

[1]Graduate School of Computer and Information Science, Chiba Institute of Technology, Narashino 275-0016, Japan
[2]Sorbonne Universités, Université de Technologie de Compiègne JR Unit - CNRS 7253, Heudiasyc, Centre de Recherche de Royallieu 60205 Compiègne Cedex, France
[3]Faculty of Information and Network Science, Chiba Institute of Technology, Narashino 275-0016, Japan

Corresponding author: Yuki Kaeri (yk.kaeri@gmail.com)

ABSTRACT In this paper, we propose an agent-based architecture for remote collaboration support systems that enables the exchange of synchronous and asynchronous multimedia streams at remote sites using an Internet of Multimedia Things (IoMT) approach. First, we design and implement Internet of Things (IoT) applications that contain simple sensors and actuators. These applications are modularized into agent-based subsystems that can be incorporated into an IoT application with agent operations. Our aim is to develop remote collaboration support applications composed of video, data, and document channels. Because the applications will exchange enormous multimedia streams between remote sites, we propose a novel IoMT system architecture composed of several channel types that consist of various resource and network components. Users can dynamically incorporate these channels into applications to update the IoMT system's effects. Finally, we demonstrate and discuss the experimental results of our application to validate its ability to rapidly supply multimedia resources via multi-agent collaboration. Adding to its novelty, the developed system is in practical use for collaboration between France and Japan.

INDEX TERMS Agent-based architecture, channel, Internet of multimedia things, multimedia resource, remote collaboration support.

I. INTRODUCTION

Designing tools to support collaboration between remote teams is a challenge. Systems integrating these tools are necessarily complex because they must allow different types of activities and capitalize on the documents and data produced or used during collaborative work sessions, and because the interactivity requirements of users, companies, and organizations have become highly demanding in the Internet age. In [1], we described the types of activities that may be performed during remote collaboration.

Communication between remote teams is also very important, and the use of a good video conference system is mandatory. Many solutions propose to integrate tools into closed videoconference systems. Section II-D describes some of these systems. Their drawback is that some integrated tools are useless or maladapted to real-world needs, and that other desirable tools may be missing. They are also often designed to connect several individual users rather than remote teams. They may provide some support for simple conversations between people; however, they are insufficient to facilitate true collaboration between teams.

Open solutions that include a videoconferencing system are better solutions, but they must also provide options for adding new tools based on a team's requirements. Videos recorded during work sessions are among the data produced by teams and, thus, can be reused. Moreover, these solutions link the data produced by disparate tools for better added value.

Recently, Alvi *et al.* [2] proposed the Internet of Multimedia Things (IoMT) as a ''novel paradigm in which smart heterogeneous multimedia things can interact and cooperate with one another and with other things connected to the Internet to facilitate multimedia-based services and applications that are globally available to the users.'' The open solution concepts that we recommend are completely in accord with this paradigm, which could come to define such systems.

In this context, the main characteristics of a videoconferencing tool as a component of a global solution for supporting the collaboration of remote teams can be summarized as follows:

- End-to-end delay: Network characteristics should ensure an acceptable delivery rate for multimedia content.
- Several points of view: Several multimedia devices (microphones, cameras, etc.) should be added in a plug-and-play manner to deliver different points of view to teams.
- Access: Multimedia content should be processed and stored for both synchronous and asynchronous access.

One of the most challenging research issues concerning collaboration support between remote teams lies in the design of an agent-based architecture model that meets users' needs according to different collaboration objectives, situations, and sizes. Components of support systems should be added and removed dynamically based on changes in the context of the collaborations.

To tackle this issue, we prototype several IoT applications that incorporate multimedia devices into programs working in a server. In these applications, sensors and actuators are modularized into agent-based subsystems to be dynamically incorporated into IoT applications. We formalize a five-layer model for IoT applications that is inspired by Alvi's IoMT architecture [2].

However, the architecture model of IoT applications cannot be applied to our remote collaboration model [1] because it cannot incorporate and control multimedia communication and cloud facilities in closed applications. Therefore, we introduce a novel concept of *channels* that incorporate, establish, and control multimedia communication and cloud resources into an IoMT system architecture. Based on this architecture, we develop a remote collaboration support system and conduct experiments to validate the rapid availability of multimedia resources on each site, as well as three methods for adding new compatible channels: adding agents, updating effects, and adapting mechanisms [3].

Section II presents some related work concerning domains in which such architectures must be considered. In Section III, we introduce the notion and structure of fast availability of multimedia resources in IoT applications stemming from previous studies. In Section IV, we define multimedia channels and synchronous and asynchronous activities in remote collaboration. In Section V, we design an agent-based architecture for remote collaboration support systems that include multimedia channels. Section VI discusses the results of experiments conducted between sites in Japan and France.

II. RELATED WORK

A. IoT ARCHITECTURES

As identified by Atzori *et al.* [4] and Razzaque *et al.* [5], the IoT consists of three components: the things themselves, an adapted middleware, and semantics. If we consider the first two components to be generic, then we must accept that the semantic aspect of the data exchanged through the middleware depends on the system's application [6].

In the Architectural Reference Model (see ARM[1] for details) defined in the IoT-A European research project [7], things consist of three types of devices—sensors, tags, and actuators—and can provide data to different external systems. The physical entities are the identifiable parts of the physical environment and can be almost any object or environment. The virtual entities are virtual counterparts of the physical objects, defining their functionalities, whereas things receive and provide data to fulfill these functionalities.

In [2], Alvi *et al.* proposed an architecture for IoT systems. We have reproduced the figure shown on page 91 of this work in Figure 1, keeping only the titles of the architecture layers for comparison with our own architecture, which is described in Section V-B.

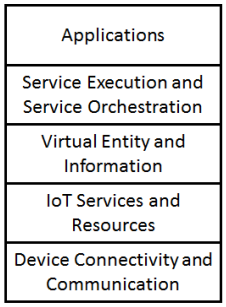


FIGURE 1. IoT architecture surveyed by Alvi *et al.* [2].

B. IoMT AND MULTIMEDIA RESOURCES

Alvi *et al.* [2] defined the IoMT as "the global network of interconnected multimedia things which are uniquely identifiable and addressable to acquire sensed multimedia data or trigger actions as well as possessing capability to interact and communicate with other multimedia and not multimedia devices and services, with or without direct human intervention." They also noted that "the services and resources layer is meant to provide the discovery and search functionalities, so that things are not restricted to provide data to a single specific vertical deployment but are available to external systems for the benefit of the connected things ecosystem."

Indeed, we also consider that multimedia resources must allow horizontal communication with other multimedia resources (e.g., when sending video streams), and vertical communication toward the core of the applications to which they belong. Alvi *et al.* [2] also reported that "virtual entities augment the things with additional functionalities implemented by the cloud." However, it is important that each resource component does not directly implement the interface needed to communicate with other entities situated in the cloud, such as services or agents, but uses an appropriate translator, which we call a resource connector [8], [9].

[1]Final Architectural Reference Model for the IoT: http://www.iot-a.eu/public/public-documents/d1.5/view

Botta *et al.* [10] introduced the role of cloud entities inside IoT architectures. They coined the term ''CloudIoT paradigm'' and provided a CloudIoT application landscape. This paradigm involves new types of applications, and new advances are required to reach this level of integration. Most current studies consider communication, storage, and computation to be advantages of cloud integration. However, the implementation of our platform mainly involves storage (of videos, documents, etc.) and computation. Communication for stream transfer is considered to be peer-to-peer communication.

C. MULTIMEDIA DEVICES FOR COLLABORATION SUPPORT

Our IoT vision implies a much wider category of IoT objects than the use of simple objects such as sensors. Any type of touchscreen device may potentially be an IoT object. However, their ability to serve as both input and output devices makes their integration into applications more difficult.

Multitouch displays are increasingly integrated into office environments [11] and other public areas. We must consider two types of displays: horizontal and vertical. Large vertical touch displays (boards) are easy to install and are mainly single-user displays because of their orientation and because they can show any application designed for simple screens.

Jones *et al.* [12] presented the TATIN-PIC project, the domain of which is preliminary project design. Project management activities gain added value when using large touch devices. Writing something on a tabletop requires opening a virtual keyboard, making it hard to edit even a short text [13]. Technological advances are still needed to more naturally render creative activities. Moreover, the use of mobile devices that interact with these surfaces to overcome their limitations is still necessary [14].

D. VIDEOCONFERENCE TOOLS

In this section, we give examples of communication tools (Skype, WebEx, etc.) and IoT platforms, and show why they do not fulfill our requirements.

Skype[2] is not intended for professional communication, but for personal communication. It links one or more individuals already registered to the platform. A chat and messaging system distributes comments and textual information and transfers documents. A main disadvantage of Skype is that these documents may be scattered across conversations, and no direct capitalization is available to benefit meetings.

WebEx[3] brought online meetings that include videoconferencing with screen-sharing viewing modes to regular people. Presenters can be switched, desktops can be shared, ideas can be sketched on a virtual whiteboard, and meetings can be recorded. WebEx is an interesting tool that achieves most of the requirements of our architecture, even if it is a closed system. However, remote collaboration support tools need several video streams from different cameras and synchronous data applications, making them much more sophisticated than a simple virtual whiteboard, in which each idea can be managed independently.

E. VIDEOCONFERENCING METRICS

During videoconferences, users must feel comfortable when communicating. They should have perfect speech intelligibility and perceive satisfactory lip synchronization when people are speaking. Hence, measuring the quality of videoconferencing applications is important for researchers.

Bartoli *et al.* [15] considered intrastream and interstream synchronization of speech and video in videoconferencing services over IP networks. They developed specific algorithms for preventive control of the speech stream and reactive control of the video stream.

Because users must feel comfortable when communicating via these applications, analyzing their performance and the quality of experience they provide is mandatory. Lu *et al.* [16] recommended studying the following aspects of videoconferencing applications: i) traffic load control and balancing to better use limited bandwidth resources and enable stable conversation and ii) stream re-encoding to limit overall traffic.

We find in [17] three levels of signals: i) standard, with 320×240 pixels of video resolution and a 15 frame-per-second (fps) video signal rate; ii) high quality, with 640×480 pixels of resolution and a frame rate of 30 fps; and iii) high definition (HD), with the highest resolution, 1280×720 pixels, and a frame rate of 30 fps. In our experiments, we consider high-quality signals.

III. AGENT-BASED ARCHITECTURE OF IoT APPLICATIONS FOR MODULARIZATION OF MULTIMEDIA RESOURCES

A. RESOURCE CONNECTOR FOR MODULARIZATION OF IoT RESOURCES

Figure 2 shows the generic structure of a module. It contains a multimedia device that is incorporated into an application, a resource (a plugin program) to control the device, and a resource connector that enables the resource to communicate with service execution agents for an application. The resource connector is an agent program that exchanges agent messages with the service execution agents and incorporates resources as plug-in programs. The resource connector has a knowledge base for saving specifications of a resource and a multimedia device as described by the resource's developer. A service execution agent can find a proper multimedia device using a cooperative agent protocol by broadcasting a request message that contains a requirement specification demanded by the application's user [8], [9].

For example, in Figure 2, an input module has a stream directed from a multimedia device (a microphone) toward a resource (a speech-to-text (STT) engine), and a resource connector composes an agent message that contains the recognized text in a format that is defined in the knowledge base of the resource connector. Alternatively, an output module

[2]Features of Skype: https://www.skype.com/en/features/
[3]https://www.webex.com/

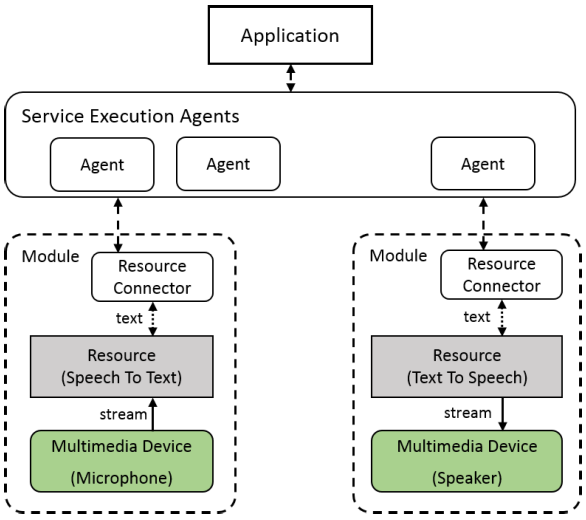


FIGURE 2. Generic structure of a module to incorporate resources into IoT applications via a resource connector and service execution agents.

produces a stream toward a multimedia device (a speaker) via a resource (a text-to-speech (TTS) engine) that is input by a resource connector that receives an agent message from a service execution agent that contains text (see the right side of Figure 2). That is, each resource connector translates an agent message from (or to) a service execution agent to (or from) a stream corresponding to characteristics of a multimedia device using its knowledge base.

A module composite application uses more than one module. Its core communicates with the agent space, and each of its modules communicates with agents via the resource connector (see Figure 2). The application can add and exchange devices dynamically thanks to service execution agents and resource connectors. A resource may, in some cases, also send commands and data to another resource.

B. AGENT LAYER TO INCORPORATE MODULES INTO IoT APPLICATIONS

Figure 2 details the lower layers of our system architecture, and Figure 3 shows the five-layer architecture of complete IoT applications that include a resource connector agent layer to incorporate modules into an application working inside the application layer. The agents communicate with applications via the IPC protocol and with resource connectors via an agent-based protocol, as shown in Figure 2.

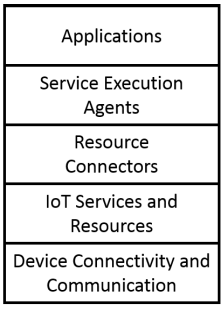


FIGURE 3. Agent-based architecture for the Internet of Things.

We now explain our five-layer architecture for IoT applications before proposing an agent-based architecture for IoMT systems. The device layer consists of the devices that interact with users and objects in physical space; the resource layer consists of programs that control devices and the data captured by these devices; and the resource connector layer consists of agent programs that communicate with the agents of the service execution agent layer.

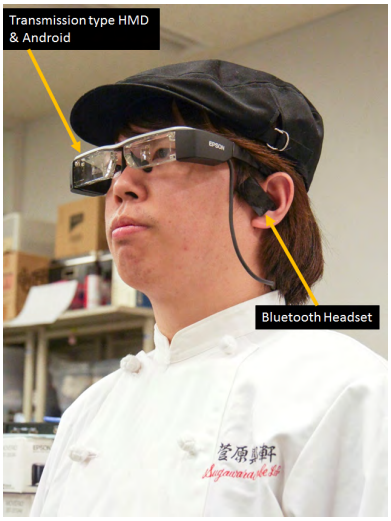
C. AGENT-BASED ARCHITECTURE FOR PROTOTYPING IoT APPLICATIONS TO STUDY RAPID RESOURCE AVAILABILITY

In this section, we introduce a notion of system expandability. A system is expandable if a new kind of resource required by a user can be managed by a new subsystem during operation.

To study methods for making a system expandable, we prototype several small applications built on the proposed agent-based architecture. Each application involves one or more modules linked to devices and to the service execution agent to fulfill its objectives.

Virtual-space instruction application (Figure 4a):

In this application, users must follow the instructions received and behave accordingly. Users wears



(a)

(b)

FIGURE 4. IoT applications developed based on the proposed agent-based architecture. (a) Virtual-space instruction application. (b) Local conference support application.

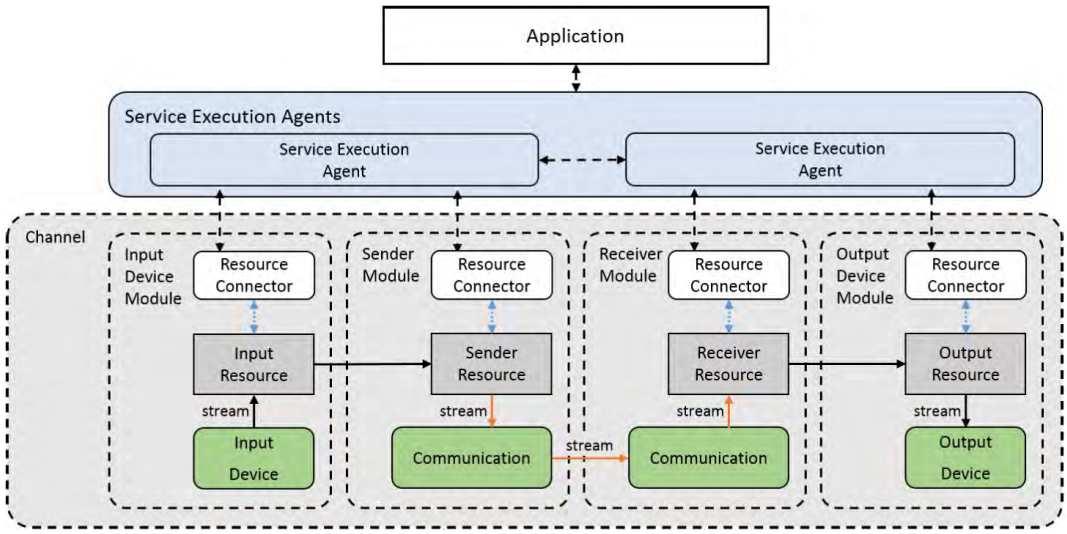


FIGURE 5. Structure of a channel composed of four modules.

head-mounted display glasses and a headset, and receive via a TTS component an instruction that remains visible using an augmented reality component of the glasses. Once a step is achieved, the user reports it to the system using an STT recognition component, and the system presents the next step.

Gesture recognition application:

A Kinect camera can detect the presence of an individual in a room and follow this presence. The detection of movements and hand and arm gestures produces events that are sent to modules based on personal agents. Small behaviors can then be interpreted from these events by artificial intelligence algorithms.

Posture detection application:

The position of a person sitting on a chair can be recognized and analyzed using chair pressure sensors. The application uses these data to measure the person's center of gravity and recognize how the individual is sitting. Furthermore, in response to these results, the application reproduces this state on a 3D virtual display.

Conference application:

A conference presenter using a slideshow displayed on a touch screen is filmed by a camera (Figure 4b). Attendees can send messages stored by the system, and the use of the slideshow (slide duration during the conference, sentences and words in the slide elements, etc.) is analyzed by agents. Then, after the conference, a video viewer allows review of the video together with the comments sent during the conference and the slide analysis data.

D. ON THE EXPANDABILITY OF RESOURCES

Each application described in the previous section involves several modules connected by resource connectors and service execution agents. We expand the first application into an integrated application by adding service execution agents from the second and third applications to it. If we expand the first application by adding these resources to the first application directly (manually), we should modify the first application to adjust its data format and protocol according to resource specifications. Using the proposed agent-based architecture, we could connect the first application with resource connectors by sending several agent messages.

Furthermore, for the fourth application, we develop resource connectors that control several sizes of multitouch displays and launch them before a meeting between two users. We assume that three other members join during the meeting. Then, the fourth application exchanges the previous small multitouch display for a bigger multitouch display by rapidly sending agent messages.

Therefore, we consider that the proposed agent-based architecture makes applications expandable. In the next section, we extend the proposed agent-based IoT application architecture to an IoMT application architecture.

IV. CHANNELS FOR THE INTERNET OF MULTIMEDIA THINGS

A. MULTIMEDIA FACILITIES FOR COLLABORATION SUPPORT

A collaboration support tool for remote teams requires different functionalities. The main facilities for synchronous activities between sites are as follows:

Communication and awareness facilities:

One or more cameras in one location must transmit video streams to the other location. One station must receive and display the video streams captured elsewhere. To collaborate between remote sites, these facilities must necessarily communicate their states each other (e.g., functions such as video chat consisting of this stream).

Data facilities:

Data created with a specific synchronized tool in one place must be accessible with the same tool in the other location.

Document facilities:

Documents of interest chosen in one place must be accessible in the other.

Figure 11a shows a part of the meeting viewer, which is an application we developed that shows corresponding camera views between locations. When several cameras are attached to the same computer, it is possible to open several meeting viewers on the same screen, providing better awareness of the atmosphere in the remote meeting location.

Other facilities concern the support of asynchronous activities. For example, someone not attending a meeting might be interested in viewing the video registered during a session. These types of facilities require two elements. First, the architecture must encompass an information system for storing videos and documents that have been exchanged. If these media are compliant with universal formats (such as PDF documents or JPEG images), they can be delivered without problems. Second, adapted viewers are required. In our system, videos are stored with annotations, which allows users to more easily search for notable events that occurred during the sessions.

B. IoMT SYSTEM CHANNELS

To fulfill all these facilitation requirements, we logically organize the architecture as a set of parallel, directed channels distributing streams from one place to another. A channel is defined by four modules—input device, sender, receiver, and output modules—as shown in Figure 5. A channel is organized by a service execution agent, according to application requirements, to send a stream to another application.

Streams can be continuous or discrete. Continuous streams include video streams, whereas discrete streams are streams in which data and events are exchanged. This organization is rapidly available in the sense that new video channels can be added according to the number of available cameras, and new instances of documents and data channels can be activated. It is possible to imagine that new types of channels could be added, but they should be developed based on this channel structure. The number of open channels may be even or odd, depending on the number of connected devices. Figure 6a shows simple video channels, and Figure 6c shows a more complex application in which seven channels are opened.

C. SYNCHRONIZED DATA CHANNEL FOR REMOTE COLLABORATION APPLICATIONS USING MULTITOUCH DISPLAYS

Figure 7 shows a pair of synchronized data sharing (SDS) applications that comprise a subsystem, as shown in Figure 6b, of a remote collaboration support system. An SDS

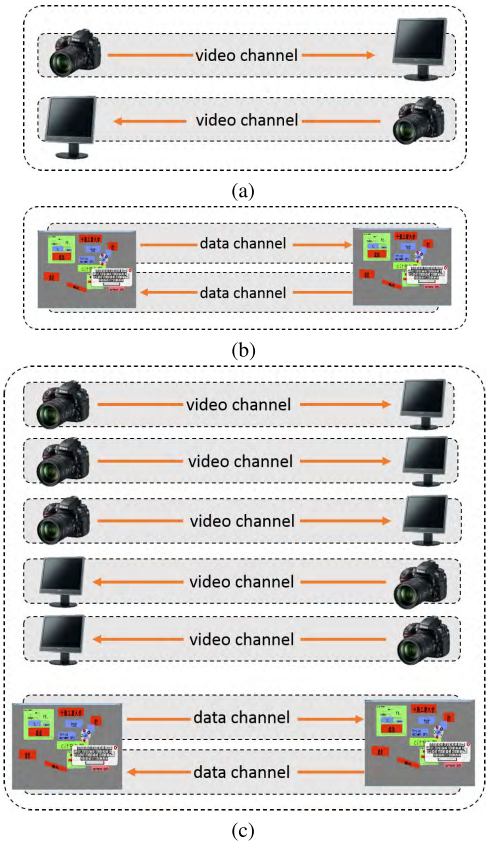


FIGURE 6. An agent-based collaboration tool: an IoMT system composed of video and data channels organized by agents. (a) Video chat application. (b) Data exchange application. (c) Remote meeting support system.

application working in a local site cooperates with input and output service execution agents. The input service execution agent cooperates with input and sender resource connectors; the output service execution agent cooperates with receiver and output resource connectors.

An input resource connector controls an input resource to capture events on a multitouch display and to send the event data to a sender resource that is controlled by a sender resource connector. A receiver resource connector controls a receiver resource to receive data streams and send them to an output resource to display events described in the stream that is controlled by an output resource connector.

The concept of a channel, as described in Section IV-B, is necessary for realizing remote collaboration. However, the architecture proposed in Section III cannot deal with expandable use of each channel. To implement such a channel, this architecture separates the flow of data between the service execution agent layer and the multimedia device and communication Layer. This data separation ensures that the layer above the resource connector does not need to handle the resource data stream. Therefore, this architecture can realize a structure that is adaptable to various forms of multimedia resource data.

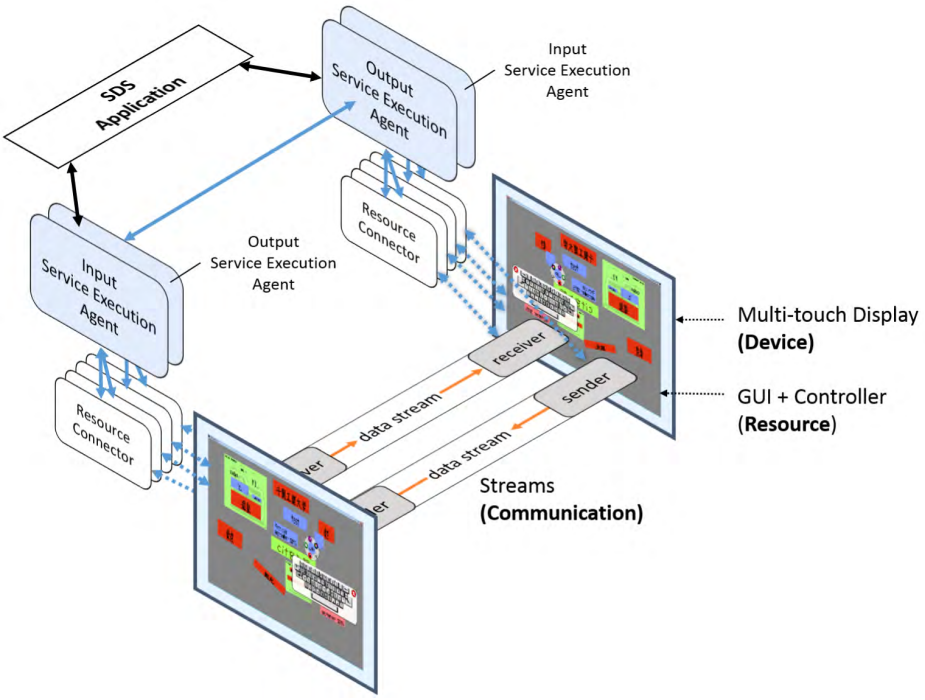


FIGURE 7. Synchronous data sharing subsystem composed of two data channels and multitouch displays.

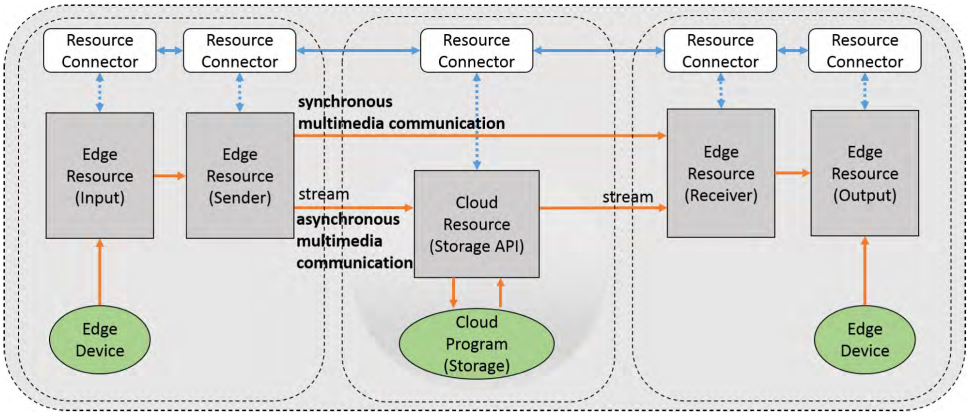


FIGURE 8. Structure of a composite channel for a synchronous multimedia communication and asynchronous multimedia communication.

V. COLLABORATION SUPPORT SYSTEM DESIGN BASED ON AN AGENT-BASED IoMT ARCHITECTURE

A. TYPES OF CHANNELS FOR SYNCHRONOUS AND ASYNCHRONOUS MULTIMEDIA COMMUNICATION

As shown in Figure 8, an edge resource directly controls a device that interacts with the environment and users (e.g., an interactive display) or a communication facility of the Internet and a local network. Therefore, edge resources perform real-time processing, and mainly operate synchronously. A cloud resource controls objects in the cloud (e.g., a multimedia database). Clouds resources mainly perform asynchronous operation. Modularization of resources and objects in the cloud is also configured by the resource connector described in Section III.

A system consists of different components and subsystems that need to exchange data. Some edge resources

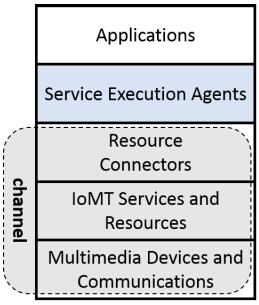


FIGURE 9. Proposed agent-based architecture for an IoMT system.

(e.g., senders, receivers) must be given the address of the corresponding end point for sending streams directly and synchronously (see Figure 8). This address is given to edge resources by the service execution agents. Other type of edge resources send data to a cloud resource to store in the cloud and read asynchronously. We designed a composite channel composed of synchronous communication to send streams directly to a corresponding edge resource, and asynchronous communication to send them to a cloud resource for storage, as shown in Figure 8. These types of channels can be chosen by applications according to performance requirements, costs, and sharing using execution agents and resource connectors.

B. PROPOSED AGENT-BASED ARCHITECTURE FOR AN INTERNET OF MULTIMEDIA THINGS APPROACH

We propose a novel IoMT architecture that introduces the agent-based architecture described in Section III and the channel concept described in Section IV to an IoMT architecture, as proposed by Alvi *et al.* The agent-based framework can turn a multimedia device and its control program into an agent-based module and let service execution agents organize them dynamically as subsystems, given application requirements. The channel components of the subsystems exchange data among modules controlled by resource connectors.

Figure 9 shows the proposed agent-based architecture for developing IoMT applications.

Multimedia device and communication layer:
: Different types of edge devices in Section V-A (e.g., cameras, displays, and touch and multitouch displays) are components of this layer. Touch devices can be used as both input start points and output end points of channels. Different multimedia communication facilities and protocols, e.g., wireless networks, MQTT, and WebRTC, are also components of this layer.

IoMT service and resource layer:
: Resources can communicate with things and other resources through channels and with the involvement of resource connectors in the applications. One special type of resource is the cloud resource, which manages data, knowledge, videos, and annotation bases.

Resource connector layer:
: Interfaces between resources and agents that translate a resource's internal data representation into a format compliant with the semantic model of the agents. Resource connectors are presented in more detail in [18]. They can communicate with agents in messages adapted to the agent language protocol.

Service execution agent layer:
: A service execution agent is a process that sends and receives agent messages to and from resource connectors to launch and operate modules according to requests from an application. Service execution agents provide APIs for applications to send requests and receive responses.

Application layer:
: Application cores ensure the logic of an application.

A triple of a resource connector, resource, and device or communication is called a module. A synchronous channel is a mechanism to convey streams between devices via some communication facility. An asynchronous channel is a mechanism that convey streams from (or to) a device to (or from) an object in the cloud. A composite channel combines both

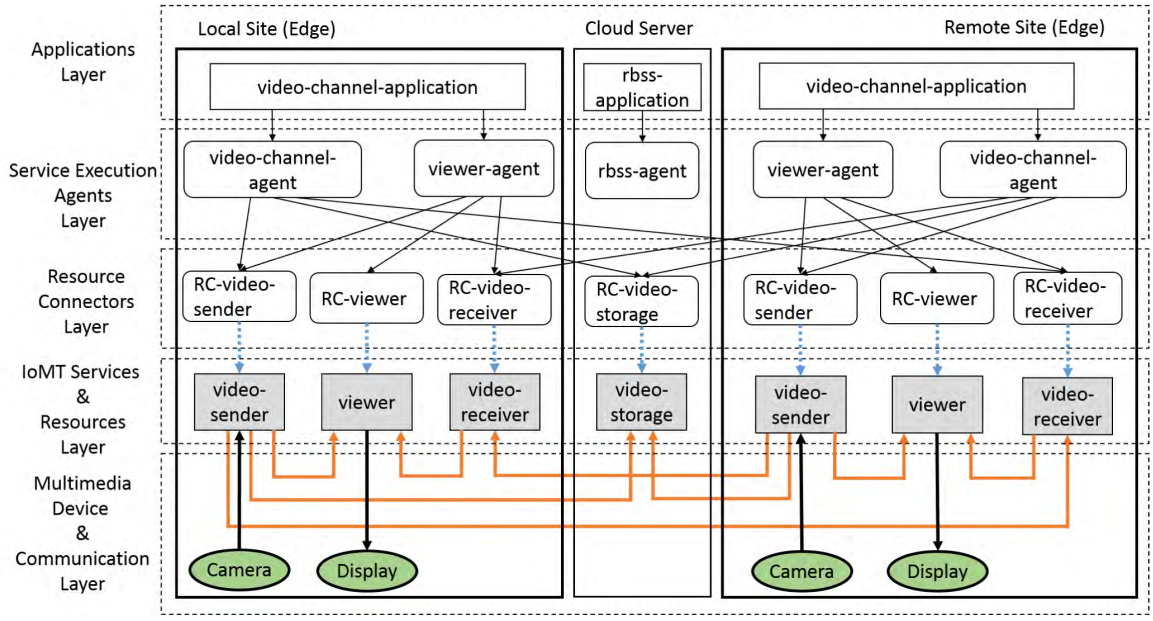


FIGURE 10. A video channel subsystem in a remote brainstorming support system.

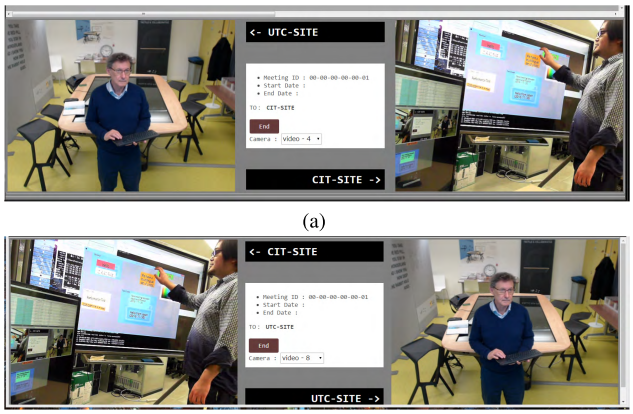


FIGURE 11. Video channel application based on the proposed IoMT architecture. (a) A meeting viewer on a display: an edge resource of a local team. (b) A meeting viewer on a display: an edge resource of a remote team.

channel types, as shown in Figure 8. These channels are dynamically composed of modules by requests from service execution agents.

C. VIDEO CHANNEL SUBSYSTEM DESIGN FOR REMOTE COLLABORATION SUPPORT

The proposed architecture comes in several representations based on applications and channels. Figure 10 shows a video channel subsystem of a collaboration support system that is composed of several modules, and both composite channel directions. The modules are launched by a service execution agent in response to requests from application components at two local sites and a cloud server.

A viewer, a multimedia resource opened at one team's location, displays both the local and remote teams (Figures 11a and 11b). The viewer receives a stream from a video sender (attached to a camera), and a stream from the video receiver is linked to a display. A video sender sends a video stream to a distant video receiver and video storage. The viewer agent and the viewer channel agent send commands to the video senders and receivers through adapted resource connectors. The video channel subsystem is one application involved in the remote collaboration support composite application.

The architecture may come in several different subsystem representations. Figure 12 shows the different synchronous application types and the launcher application. Each synchronous application must have two similar instances that are end points of a channel. The images presented in the figure at the device layer are the meeting viewer (for the video channel application), the data editor (for the data channel application), and the document viewer (for the data channel application). The launcher is used to open instances of the applications at each site.

VI. IMPLEMENTATION AND EXPERIMENTS

A. IMPLEMENTATION OF A REMOTE COLLABORATION SUPPORT SYSTEM

The system in Figure 12 is the result of a research project carried out at the Chiba Institute of Technology[4] (CIT) in Japan and the University of Compiègne[5] (UTC) in France over several years. These experiments have permitted us to elaborate the concepts presented here, to overcome obstacles due to network communication, and to account for the security features set up in universities. Moreover, as this project is itself a remote collaboration project, it has also served as a good benchmark.

[4]CIT site: http://www.it-chiba.ac.jp/english/
[5]UTC site: https://www.utc.fr/en.html

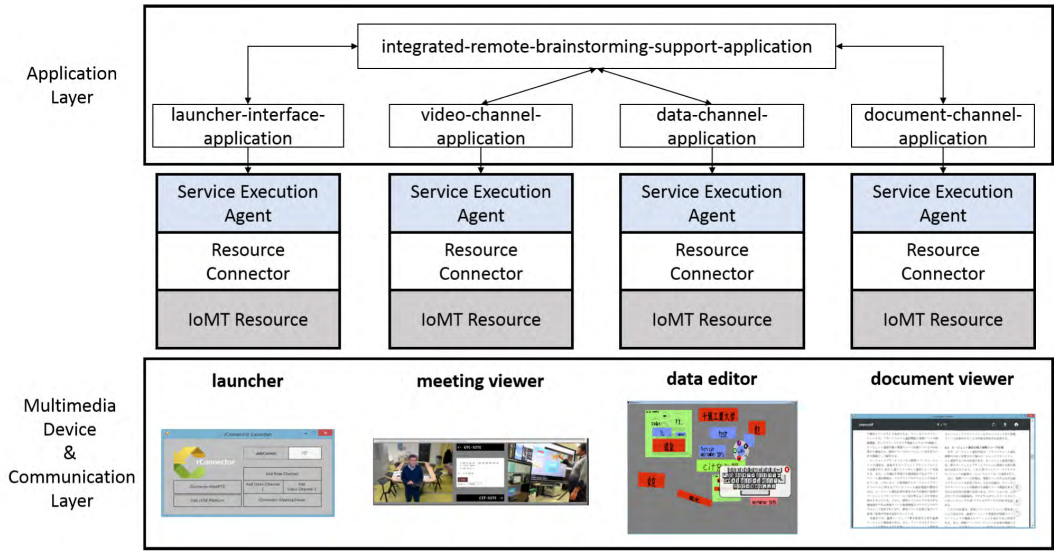


FIGURE 12. Integrated remote brainstorming support system application based on the proposed IoMT architecture.

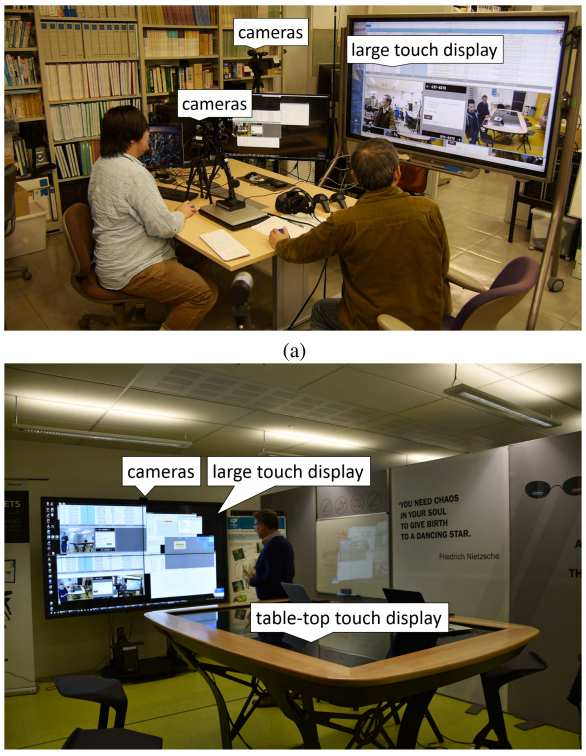


FIGURE 13. Experimental contexts of our remote collaborative supporting system. (a) Experimental context from the CIT site. (b) Experimental context from the UTC site.

Figure 13a and right side of Figure 11b shows the context of experiments at the CIT. At the Faculty of Information and Network Science Department, the team used a large touch board to display the meeting viewer and another touch screen to display the data editor. Three cameras allowed different views of the room. Figure 13b shows the context of the experiments at the UTC. Its Innovation Center[6] allowed us to benefit from a room equipped with a large touch board and a large touch table top.[7] Three cameras, attached to the board, proposed different axes and viewpoints in the room. Microphones were coupled with the cameras. The cloud server was managed by the CIT team.

We implemented the system as in the design in Section 8. Edge resources were implemented using C# (.NET Framework 4.5.1) and cloud resources were implemented using Python 3.6.

In our platform, stream communications were based on the WebRTC infrastructure,[8] which provides browsers, mobile devices, and IoT-device-based applications with real-time communication capabilities. WebRTC communicates streaming data, but also needs a mechanism to coordinate communications and to send control messages. It was therefore necessary to modify the event streams that synchronize two instances of the data editor. In this study, WebRTC was used as a network protocol of resources controlled from the network component and to implement the agent layer's protocol.

Our applications needed to traverse firewalls. Hence, we used a STUN server to get IP addresses and a TURN server as a relay. For the experiments, we employed a data channel server, a data repository, a video channel server consisting of a STUN/TURN server using Coturn,[9] a signaling server using PeerServer,[10] and a video repository we implemented using the PHP and Python languages and MariaDB. All servers were deployed on ConoHa,[11] an OpenStack cloud.

B. REMOTE COLLABORATION SUPPORT SYSTEM EXPERIMENTS

1) UTILITY OF A REMOTE COLLABORATION SUPPORT SYSTEM BASED ON THE PROPOSED IoMT ARCHITECTURE

We measured the delays in video and data channels between the CIT and UTC sites. Figure 14 shows the experimental system used to measure these delays. Local programs and devices at both sites access the cloud servers to construct two

[6]UTC Innovation Center: https://www.utc.fr/en/innovation.html
[7]These materials were designed and built by UbiKey: http://www.ubikey.fr/

[8]WebRTC site: https://webrtc.org/
[9]Coturn site: https://code.google.com/p/coturn/
[10]PeerServer: https://github.com/peers/peerjs-server
[11]ConoHa site: https://www.conoha.jp/en/

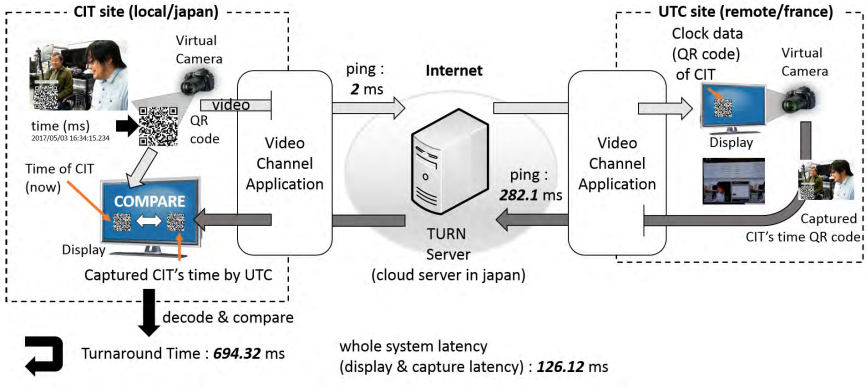


FIGURE 14. Measurement of the communication latency of the video channel application.

channels. The resolution of each camera was 640 × 480 and the frame rate was 30 fps.

First, we implemented a measurement application from Boyaci *et al.* [19] to measure accurate turnaround times. This application converts the current time into QR code and outputs the video in the video channel using a virtual camera. These functions were implemented with OpenCV[12] and ZXing.NET[13]; we used XSplit Broadcaster[14] for the virtual camera.

Next, we measured the delay with a ping command to calculate the system latency from the measured turnaround time. Figure 14 shows result of this measurement. The turnaround time was 694.32 ms (including Internet line delays for each site to the cloud server), and the system delay was 126.12 ms (including processing in the cloud server). Furthermore, the channel delay in one direction was measured as 63.06 ms. In this experiment, when evaluating Skype under the same conditions, the turnaround time was 1109.46 ms, and the system delay at the time of measurement was 541.26 ms. The delay on the Skype channel in one direction was 270.63 ms.

2) CHANGEABILITY OF THE REMOTE COLLABORATION SUPPORT SYSTEM BASED ON IoMT ARCHITECTURE

We conducted experiments to add video and data channels to a system that was opened by a user who managed the remote meeting to examine the system's changeability [3]. In these experiments, we tested the typical requirements shown in Figure 13. The initial system configuration consisted of two video channels comprising a camera and a display at each site, as shown in Figure 10.

After several minutes, the user at the CIT site operated the system with a control panel (launcher) at his local PC to add a data channel to the initial configuration. Agents at both sites sent messages to resource connectors at the CIT and UTC sites. In this experiment, we assumed the two agents knew the identifiers of the local and remote resource connectors that launch data channel components for controlling a multitouch display for collaboration, and network components for sending and receiving data channel streaming. Figure 15 shows a view of two video channels for the CIT site and a data channel that was added to the display by the user.

After several minutes, the user again operated the launcher to add two video channels, to add one more viewer to the display, and to see different pictures from different cameras, as shown in Figure 15. Then, because a new viewer was displayed over the data channel, the user moved the data channel viewer in the display on the screen in Figure 15. The requirements shown in Figure 13 were conducted successfully after those in Figure 15.

[12]OpenCV site: http://opencv.org/
[13]ZXing.NET site: https://zxingnet.codeplex.com/
[14]XSplit Broadcaster site: https://www.xsplit.com/

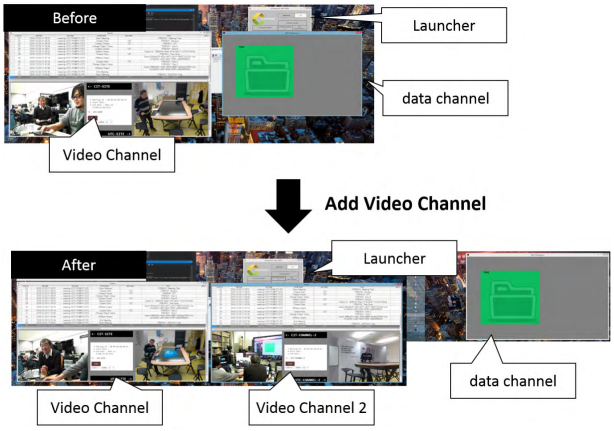


FIGURE 15. An add-channel operation on the remote collaborative supporting system.

C. DISCUSSION

First, to verify the expandability of the data and video channels, we conducted experiments using the implementation described in Section VI-A. Expandability was defined in Section III-C as a user's ability to add to or replace system resources during operation.

As shown in Figure 12, a video chat application in Figure 6a, composed of two video channels, and a data exchange application in Figure 6c, composed of two data channels (as shown in Figure 7) were simultaneously opened for the first stage of the tests. During the session, we supposed that participants required better awareness for remote collaboration using a launcher-interface application. Then, the request was given to two video channel applications in Figure 12 and three video channels were added to the initial system, as shown in Figure 6e. System operation was faster than operation by participants themselves for performing the same change. To treat various requirements for changing system properties, we should incorporate intelligence into the launch interface application and the integrated remote brainstorming support application in future work.

To investigate expandability criteria, we tested the quality of video and audio transmission and the swapping of cameras inside the meeting viewer application. Once it was completely finalized, we did not use other videoconferencing systems for our distant collaboration meetings. When we used an independent videoconferencing system, it seemed necessary to explain what we were doing with the data editor ("I am creating a new note on…" or "I am moving note …into cluster…"). With our own videoconferencing system and its good reception, the independent system was no longer useful. Moreover, with our own system, what people do locally is easily understood in the remote location.

The delay in our system seems to have necessary and sufficient performance for a conference support system. Our measurement tests demonstrated the complete usability of the meeting viewer. The rate of the transmission was completely satisfactory, and the participants could speak together and have discussions as if they were in the same room.

VII. CONCLUSION

In this paper, we proposed an innovative agent-based architecture for systems supporting remote collaboration based on an IoMT approach. Such applications involve basic IoT elements and devices such as cameras, microphones, sensors, and multimedia communication lines. Our architecture is divided into five layers: applications, service execution agents, resource connectors, IoMT services & resources, and IoMT devices & communications.

We introduced the concept of modules presented by a triple of the bottom three layers corresponding to each device and communication such as touch displays, synchronous and asynchronous communication lines, and cloud storage. The parallel channels require several graphical user interfaces opened at the same time and, more importantly, a display surface. Those channels can be incorporated into applications dynamically by users to enhance the expandability of the IoMT system.

This practical system supports conferences among multiple sites. In the future, we would like to provide this system to users as a service and obtain feedback from them.

REFERENCES

[1] C. Moulin, Y. Kaeri, K. Sugawara, and M.-H. Abel, "Capitalization of remote collaborative brainstorming activities," *Comput. Standards Interfaces*, vol. 48, pp. 217–224, Nov. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01396766

[2] S. Alvi, B. Afzal, G. Shah, L. Atzori, and W. Mahmood, "Internet of multimedia things: Vision and challenges," *Ad Hoc Netw.*, vol. 33, pp. 87–111, Oct. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870515000876

[3] A. M. Ross, D. H. Rhodes, and D. E. Hastings, "Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value," *Syst. Eng.*, vol. 11, no. 3, pp. 246–262, 2008. [Online]. Available: http://dx.doi.org/10.1002/sys.20098

[4] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT)—When social networks meet the Internet of Things: Concept, architecture and network characterization," *Comput. Netw.*, vol. 56, no. 16, pp. 3594–3608, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128612002654

[5] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7322178/

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[7] (2013). *IoT—A, Internet of Things Architecture, eu Integrated Project, 7th Framework*. [Online]. Available: http://www.iot-a.eu/

[8] Y. Kaeri, Y. Manabe, and K. Sugawara, "A development of an agent platform for connecting ubiquitous devices and digital resources to logical agents," (in Japanese), *Trans. Inf. Proces. Soc. Jpn.*, vol. 56, no. 1, pp. 284–294, Jan. 2015. [Online]. Available: http://ci.nii.ac.jp/naid/110009867111/

[9] Y. Kaeri, Y. Manabe, K. Sugawara, and C. Moulin, "An IoT application connecting edge resources and cloud resources using agents," *Int. J. Energy, Inf. Commun.*, vol. 8, no. 15, pp. 1–14, 2017.

[10] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Generat. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15003015

[11] R. Hardy and E. Rukzio, "Touch & interact: Touch-based interaction of mobile phones with displays," in *Proc. 10th Int. Conf. Human Comput. Interaction Mobile Devices Services (MobileHCI)*, 2008, pp. 245–254. [Online]. Available: http://doi.acm.org/10.1145/1409240.1409267

[12] A. Jones, A. Kendira, D. Lenne, T. Gidel, and C. Moulin, "The TATIN-PIC project: A multi-modal collaborative work environment for preliminary design," in *Proc. 15th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, Jun. 2011, pp. 154–161.

[13] C. Moulin, A. Jones, J.-P. Barthès, and D. Lenne, "Preliminary design on multi-touch surfaces managed by multi-agent systems," *Int. J. Energy, Inf. Commun.*, vol. 2, no. 4, pp. 195–209, 2011. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00849852

[14] R. Ballagas, M. Rohs, and J. G. Sheridan, "Sweep and point and shoot: Phonecam-based interactions for large public displays," in *Proc. CHI Extended Abstracts Hum. Factors Comput. Syst. (CHI EA)*, 2005, pp. 1200–1203. [Online]. Available: http://doi.acm.org/10.1145/1056808.1056876

[15] I. Bartoli, G. Iacovoni, and F. Ubaldi, "A synchronization control scheme for videoconferencing services," *J. Multimedia*, vol. 2, no. 4, pp. 1–9, 2007.

[16] Y. Lu, Y. Zhao, F. Kuipers, and P. Van Mieghem, "Measurement study of multi-party video conferencing," in *Proc. 9th IFIP TC 6 Int. Conf. Netw. (NETWORKING)*, 2010, pp. 96–108.

[17] W. Mazurczyk, M. Karaś, K. Szczypiorski, and A. Janicki, "YouSkyde: Information hiding for Skype video traffic," *Multimedia Tools Appl.*, vol. 75, no. 21, pp. 13521–13540, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11042-015-2740-0

[18] Y. Kaeri, K. Sugawara, Y. Manabe, and C. Moulin, "Prototyping a meeting support system using ubiquitous agents," in *Proc. 19th IEEE Int. Conf. Comput. Supported Cooperat. Work Design*, May 2015, pp. 18–23.

[19] O. Boyaci, A. Forte, S. A. Baset, and H. Schulzrinne, "vDelay: A tool to measure capture-to-display latency and frame rate," in *Proc. 11th IEEE Int. Symp. Multimedia (ISM)*, Dec. 2009, pp. 194–200.

YUKI KAERI (M'89) received the master's degree in computer and information science, and is currently pursuing the Ph.D. degree, from the Chiba Institute of Technology, Japan. His research interests include multi-agent systems, human–agent interaction, symbiotic computing, web recommendation systems, ubiquitous computing, and the Internet of Things.

CLAUDE MOULIN received the Ph.D. degree in computer sciences in 1998. He was with research centers in Italy, for four years. He is currently an Assistant Professor with the University of Technology of Compiègne, France, and a member of the HeuDiaSyc Laboratory (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 multimodal interfaces.

KENJI SUGAWARA (M'11) received the Ph.D. degree in engineering from Tohoku University, Japan, in 1983. He is currently a Professor with the Department of Information and Network Science, and the Dean of the Faculty of Information and Computer Science, Chiba Institute of Technology, 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 ACM and IPSJ.

YUSUKE MANABE (M'11) received the Ph.D. degree in software and information science from Iwate Prefectural University, Japan, in 2008. He is currently an Associate Professor with the Department of Information and Network Science, Chiba Institute of Technology, Japan. His research interests include intelligent informatics, cognitive science, chaotic time series analysis, and soft computing. He is especially interested in human skill analysis, context-aware computing, and symbol-grounding mechanisms. He is a member of JSAI, JCSS, SOFT, and IEICE in Japan.