International Journal of Informatics Society

17/09 Vol. 9 No.2 ISSN 1883-4566

Informatics Society

Aims and Scope

The purpose of this journal is to provide an open forum to publish high quality research papers in the areas of informatics and related fields to promote the exchange of research ideas, experiences and results.

Informatics is the systematic study of Information and the application of research methods to study Information systems and services. It deals primarily with human aspects of information, such as its qu ality and value as a resource. Informatics also referred to as Information science, studies t he structure, algorithms, behavior, and interactions of natural and a rtificial systems that store, process, access and communicate information. It also develops its own conceptual and theoretical foundations and utilizes foundations developed in other fields. The advent of computers, its ubiquity and ease to use has led to th e study of info rmatics that has computational, cognitive and social aspects, including study of the social impact of information technologies.

The characteristic of informatics' context is amalgamation of technologies. For creating an informatics product, it is necessary to integrate many technologies, such as mathematics, linguistics, engineering and other emerging new fields.

Guest Editor's Message

Tomoki Yoshihisa

Guest Editor of Twenty-Sixth Issue of International Journal of Informatics Society

We are delighted to have the twenty-sixth issue of the International Journal of Informatics Society (IJIS) published. This issue includes selected papers from the Tenth International Workshop on Informatics (IWIN2016), which was held at Riga, Latvia, Aug. 28-31, 2016. The workshop was the tenth event for the Informatics Society, and was intended to bring together researchers and practitioners to share and exchange their experiences, discuss challenges and present original ideas in all aspects of informatics and computer networks. In the workshop 26 papers were presented in eight technical sessions. The workshop was successfully finished with precious experiences provided to the participants. It highlighted the latest research results in the area of informatics and its applications that include networking, mobile ubiquitous systems, data analytics, business systems, education systems, design methodology, intelligent systems, groupware and social systems.

Each paper submitted IWIN2016 was reviewed in terms of technical content, scientific rigor, novelty, originality and quality of presentation by at least two reviewers. Through those reviews 15 papers were selected for publication candidates of IJIS Journal, and they were further reviewed as a Journal paper. This volume includes five papers among the accepted papers, which have been improved through the workshop discussion and the reviewers' comments.

We publish the journal in print as well as in an electronic form over the Internet. We hope that the issue would be of interest to many researchers as well as engineers and practitioners over the world.

Tomoki Yoshihisa received his Bachelor's, Master's, and Doctor's degrees from Osaka University, Osaka, Japan, in 2002, 2003, 2005, respectively. Since 2005 to 2007, he was an assistant professor at Kyoto University. In January 2008, he joined Cybermedia Center, Osaka University as a senior lecturer and in March 2009, he became an associate professor. In 2008, he was a visiting researcher at University of California, Irvine. His research interests include stream data distribution, video-on-demand, and webcasts.

He has made significant technical contributions in the field of stream data distribution with over 350 research papers including top-tier international publications. He has published actively in international, refereed journals and conferences, such as IEEE Transactions on Broadcasting, IEEE BigData, and ACM SAC. He has made pioneering research achievements along with significant contributions to conferences in this field. He received the Golden Paper Award of IEEE PACRIM, the Best Paper Awards of IPSJ ICMU (twice), and the Highly Commended Award of Emerald Publishing. Also, he received Telecom System Technology Award from Telecommunications Advancement Foundation, Yamashita SIG Research Award from IPSJ and Presidential Award from Osaka University. He was organizing members of international conferences/workshops such as IEEE AINA, IEEE SRDS, and SMDMS. Also, he managed some Japanese research groups as the organizing members (IPSJ SIG DCC, IPSJ SIG DBS, IEICE IA). He was the Program Chair of IWIN2016 and is managing IWIN2017 as the Program Chair.

A Design for Wireless Sensor Network Visualization Tools Based on Network Management Principles

Yuki Urata†, Takuya Yoshihiro‡, and Yutaka Kawahashi*

†Graduate School of Systems Engineering, Wakayama University, Japan
‡Faculty of Systems Engineering, Wakayama University, Japan
* Center for Information Science, Wakayama University, Japan
{s171009, tac, yutaka}@center.wakayama-u.ac.jp

Abstract - Wireless Sensor Networks (WSNs) are considered as a key technology of up-coming IoT applications in the world. Several studies have indicated the requirements on visualization tools to support administrators for practical operation and management of WSNs. Although they proposed various visualization system design as well as diagnosis methods, no firm principle has been suggested on designing administrative systems to monitor WSNs. We in this paper provide a new suggestion on designing WSNs management system to introduce the principle of the network management grown in long time with the experiment of the Internet. Note that we cannot apply query-based information acquisition such as SNMP since duty-cycled sensor devices cannot reply in real-time. So, under the assumption of deploying the passive management, in which the values required in WSNs management is continuously collected to the sink by piggy-backing them on data packets, we provide a new design of visualization tools that suffices the practical requirements on managing WSNs based on the three network management aspects: structure, failure, and performance managements. We developed a prototype of the proposed visualization system, and conducted an evaluation experiment. The results demonstrate that the proposed system design surely supports network operators to meets all requirements on WSNs management.

Keywords: Wireless Sensor Networks, Operation, Management, Visualization

1 INTRODUCTION

Wireless Sensor Networks (WSNs) are expected as a practically important technology to develop the up-coming IoT (Internet of Things) applications that will improve the quality of our lives. Many applications of WSNs including agriculture, medical cares, factory automations, environmental monitoring, etc. have been considered in many research work to improve each area of human activities. However, to realize practical WSNs, realizing reliable communications within very low power consumption are essentially important. For this challenge, various communication protocols as well as sensor-node hardware that consume extremely low power have been developed so far.

As a well-known communication standard for sensor networks, IEEE802.15.4[1] has been promoted, and many commodity devices have been developed. However, IEEE802.15.4 in fact consumes considerable power because it is designed to cover wide variety of practical scenarios, and also to include many functions to enhance flexibility of communications. Thus, to achieve a minimum power consumption level to collect sensed values to sink nodes, lots of low-power MAC protocols for WSNs have been proposed [2]-[5].

In these MAC protocols, each node takes as much sleep time as possible to save its power consumption, by efficiently synchronizing the communication timing between a sender and a receiver nodes within a limited awaking time. From the viewpoint of synchronization mechanism, they are classified into two types of MAC protocols, sender-initiated and receiver-initiated protocols. Sender-initiated MAC protocols such as B-MAC[2] and X-MAC[3] synchronize wake-up timing using actions of sender nodes. For example, B-MAC transmits a long preamble that is longer than the wake-up interval to enable receivers being awake state when a transmitter transmits a frame. Receiver-initiated MAC protocols such as RI-MAC[4] and RC-MAC[5] synchronize wake-up timing based on the action of receiver nodes. Typically, like RI-MAC, receivers periodically transmit beacons when they are ready to receive frames, while a sender that has a data frame keeps waking-up and waits for beacons to transmit the frame. Recently, receiver-initiated MAC protocols are regarded as one of the promising approaches for practical low-power WSN systems so that many proposals have appeared in the literature [6][7].

On the other hand, similar to the networks connected to the Internet, we have to manage and operate WSNs in practice to keep WSNs work correctly to gather sensed data values. To this end, We need a system with which we can watch the state of WSNs and find the trouble as soon as it occurs so that we can take effective measures against the troubles to maintain WSNs to work continuously and correctly. There are several tools and methods developed for this purpose. In early days, query-based methods to collect information of WSNs has been tried, which is a similar approach to SNMP [8] in the Internet management. However, in low-power duty-cycle MAC protocols, such queries require significantly long delay due to long sleeping time of each node, and thus it is not a possible approach in WSNs. Alternatively, one of the basic approaches is to collect required values to sink nodes by piggy-backing them on data packets to find anomalies of WSNs. There are several proposals [9] [10] from this approach.

These methods carefully model the relationship among possible events to specify the essential causes of troubles. However, since the events that are considered in them are limited to

the ones that are supposed in advance, new phenomena cannot be treated. As the Internet management process poses, the troubles in network management have too large variety, so that those proposals cannot cover all possible cases. Also, finding a small sign of trouble before it occurs is an important issue for prevention of troubles in managing networks. Thus, in addition to the predefined diagnose systems, we need a system that visualizes the state of WSNs to help finding troubles or their prior signs in managing and operating WSNs.

In the literature, several systems that visualize WSNs exist [11]. However, they are not designed from the viewpoint of managing networks to utilize the experiment of Internet management, nor evaluated from the viewpoint.

In this paper, we propose a new design of systems that visualize the state of WSNs and help managing them. Our system design is based on the principle of Internet management, i.e., we designed to perform (a) structure management, (b) failure management, and (c) performance management, in order to keep stable operation of WSNs. We implemented and evaluated our system using simulation traces of a WSN to confirm that the system enables users to find important signs from the viewpoint of management principles (a)-(c).

This paper is organized as follows. In Sec. 2, we present related work to manage WSNs. In Sec. 3, we describe the design policy of WSN management systems. Especially, we discuss on the principal of Internet management and requirements for WSN management systems. In Sec. 4, we present the proposed system, and we evaluate the system in Sec.5. Finally in Sec.6, we conclude the work.

2 RELATED WORK

There are several related studies and systems that visualize and help managing WSNs. In this section, we describe these previous contributions.

It has been pointed out from the early stage of WSN studies that the behavior as well as the root causes of troubles in WSNs are hard to look out. Thus, several studies have been proposed to diagnose WSNs in face of troubles to find out what is going on in WSNs. Note that the cause of troubles can be inside the sensor node, i.e., software bugs that happen only in multiple-node-related scenarios are hard to eliminate in the development phase, and require a tool to help debugging. Consequently, both realfield operation and development testing can be the target of those diagnosis methods and tools.

As specific proposals, Ramanathan et al. proposed a diagnosis method called Sympathy [9] that construct a decision diagram from passively collected data at sink nodes to find out the root cause of the trouble. Liu et al. proposed a method called PAD [10] that uses causal diagram to find out the root cause. Liu et al. also proposed a method TinyD2 in which multiple nodes cooperatively work to find out the root cause of trouble [12]. Khan et al. proposed a debugging method DustMiner [13] that utilizes a set of event logs in WSNs to find the sequences of events that lead to reveal bugs and troubles. However, those methods make diagnosis from predefined causes and troubles, so that they are not possible to treat new phenomena and troubles. Unfortunately, in the real Internet management, system administrators often meet new kind of troubles so that flexible responses are required.

As another approach, several methods learn the normal state of WSNs and detect abnormal states, i.e., anomalies, based on the distance between the current state and the normal state. For example of this class of methods, Li et al. proposed a method VN2 that learns the normal state using 43 metrics of WSNs from event history logs and apply Non-negative Matrix Factorization (NMF) model to detect exceptional state of WSNs [14]. Miao et al. proposed a method called Agnostic Diagnosis (AD) that computes correlations between 22 metrics to detect anomalies from the history logs. Those systems are useful to reduce labor and time of administrators to detect anomalies, but in managing WSNs [15], the detail and the root causes must be carefully examined anyway using some visualization tools to clarify the occuring phenomena and take a proper countermeasure.

As a visualization tools for WSNs, there are several studies and systems such as Octopus [16], SRNET[17], etc. Also, several visualization tools have been provided as bundled software in WSNs devices or implementations; they are concisely introduced in a survey paper [11]. However, they all are not designed based on the principle of network management, nor evaluated from that viewpoint. Thus, the efficacy of those tools in WSN management is not sufficiently clarified. This paper is the first study that presents the WSN administration tool designed based on the principle of the Internet Management.

3 REQUIREMENTS AND THE DESIGN

3.1 Requirements for WSN Management Systems

In this study, we suppose wireless sensor networks (WSNs) that deploy low-energy MAC protocols such as X-MAC and RI-MAC, so that any query-based management protocol such as SNMP cannot be used to manage them. The basic way to manage this kind of WSNs is to apply passive strategy as presented in [9][10], where administrative data values are collected to sinks piggy-backed on data packets, and network states/problems are visualized/inferred to manage networks properly.

In managing WSNs, we are required to carefully watch the networks to keep them correctly working for collecting sensed data values. For this purpose, not only detecting the problems occurring in the networks, but also finding implicit signs of future problems is important, to prevent degradation of communication performance or extra power consumption due to redundant network behaviors. One of our primary goals is to make network administrators possible to find these signs surely and speedily by visualizing the values collected at sinks. Note that it is difficult to manage multi-hop WSNs in real time as long as it is managed in the passive way, i.e., managed through the values collected at sink nodes. In other words, we have to allow a certain level of delay on finding failures or troubles in the network management tasks. If we have a requirement on the delay performance, they should be covered by the deployed MAC and routing protocols. Accordingly, we in this paper focus on just inquiring the causes

of the failures without caring the time delay that takes to find them.

On the other side, in the Internet, network management has been a critically important issue to provide stable and secure services to end users since the Internet now has a role of social Infrastructure. In this context, the area of network management has naturally been grown in the past decade and formed a rough but firm consensus on how to manage networks. Such consensus is issued as some documents, e.g., reference [18] describes the required knowledge and administrative operations that should be performed in managing networks connected to the Internet. The document says that the network management consists of 5 specific aspects of management domains shown in the following.

Structure Management
Structure management is the task that maintains physical and logical elements in networks. Network structure is the basis of all management task, so it is siginificantly important to grasp the latest state of the network structure. In the Internet management, the network structure consists of all the physical elements such as network boxes, the logical elements such as virtual functions and configurations, and also their connections. In contrast, the structure of current WSNs is quite simple in which sensor devices, their configurations, neighbor relationship, and the paths from each node to sinks are included.

Failure Management
Failure management is the task (1) that defines the event regarded as failure, (2) considers the detecting strategy, countermeasures and preventive measures for each failure, (3) and executes them. We have two types of the failure detection approaches: active detection based on queries sent to each device, and passive detection based on the reports coming from each device. In case of this study, since we suppose WSNs which deploy low-power MAC protocols, we have no choice but taking passive detection approaches. Also, in WSNs, we mainly considers node and link failures.

Performance Management
Performance management is the task that maintains WSNs to keep a constant level of communication performance. Generally, the performance of networks include such as throughput, packet loss ratio, latency, jitter, retransmission count, congestion frequency, CPU usage, etc. In WSNs in this study, we expect each packet to reach a sink reliably within a certain latency. Thus, especially packet loss ratio and latency are the important measure of the performance.

Resource Management
Resource management is the task that maintains the system resources required in the operation of systems. The resource includes every elements that consists of the system such as hardware, software, cables, etc. Note that resource management includes CPU, memory, and network capacity management that prevents shortage of those dynamically used resources.

Security Management
Security management is the task that protects the system and the contents from the threat outside of the network. In the Internet, security is a very important issue since there are several threat such as viruses and attacks from outside the network. In WSNs in contrast, the main concern is the information leaking, which can be prevented by deploying some encryption facility.

In this study, we only consider three management aspects, i.e., structure, failure, and performance management, and omit considering resource and security management. Although resource management includes CPU, memory, and communication capacity management, they are not the matter in WSNs; WSNs use small amount of CPU and memory resources so that they are not important to manage in most cases. Communication capacity of links is an important issue in WSNs, but in wireless networks, communication capacity management is tightly connected to the performance management, since communication performance gradually degrades as communication amount approaches the capacity. As above, in WSNs, three management issues, i.e., structure, failure, and performance management, are essentially important.

3.2 The System Design

For structure, failure, and performance management, the system must collect the required administrative information from WSNs and visualize them appropreately to help operators manage WSNs properly. Collecting information is done such that each node measures several administrative values and include them in the packet that the node generates. By piggy-backing the values required for management on packets, passive management using the values gathered on the sink nodes is enabled. Note that the administrative values are not added at each node in the collection paths, but added only at the originated node of the packet. Therefore, the overhead incurred from the administrative values does not change depending on the size of WSNs.

User interface is designed in order for operators to easily find events related with the three management aspects. For structure management, we prepare the *delivery tree view* to see the network state intuitively. Every node is placed at the right position and the set of next-hop links at an arbitrary time point are shown to form the delivery tree. We can overlap two delivery trees at different time points, which enables us to see the transition of the delivery tree.

In failure management of WSNs, we find node or link failure. When the packets from a node do not arrive at sinks, we can regard that the node fails, and we can detect link failure in the similar way. Thus, we prepare the *alert table* in which possible node failure events as well as other alerts are listed up. When operators found possible node failure in the *alert table*, they usually have to check whether the failure really occurs or not. To do this, we prepare the *administrative values table* in which all the data values collected in sinks are seen

per source node, and also prepare the *line graph view* that visualize the data values per node and per item to see the values intuitively.

Finally, for performance management, we mainly watch residual power, packet loss ratio, and delivery delay at each node. By listing the events in the *alert table* when these values exceed a threshold, the operators easily find the performance degradation. After the operator is notified of the performance anomalies, they usually explore for the level of the degradation and specify what is the root cause of this trouble. We can use the *administrative values table* and the *line graph view* again for this purpose.

With the basic design policy described above, our system has the following characteristics that differentiate our system from the others.

(1) Visualizing Delivery Tree Transition
The function of overlaid display of multiple delivery trees obtained from different time points is unique to our system, which enables operators to grasp the transition of delivery paths easily and intuitively. Since this view provides an intuitive overview information, this view plays a role of the 'base' page from which we can explore several detailed data values.

(2) Alert Table to be Aware of Administrative Events
We prepare the *alert table* that makes administrators keep aware of important administrative events occurred in WSNs. By simply applying thresholds to several carefully-selected administrative data items that represent network performance, administrators can watch WSNs via *alert table* to find important events.

(3) Intuitive Operation
From the base *delivery-tree view*, we can intuitively transit to other views by clicking entities nodes and buttons. In our user interface design, the administrators are possible to access the related data values to explore for the state of WSNs.

4 THE PROPOSED SYSTEM

4.1 System Structure

The proposed system visualize the network state from the administrative values collected to sinks. We show the system structure in Fig. 1. A sink node collects the sensed values generated periodically at every node. Note that there may be multiple sink nodes in a WSN, but the server collects values from all sink nodes. The server provides the function of web servers so that the administrators access to the server via Web browsers to visualize the state of WSNs.

4.2 Attaching Administrative Values to Packets

In our framework, we piggy-back the administrative values on packets to collect them to the sinks in WSNs. As the administrative values, we used the following 18 items that are typically used in administrating networks. Note that every item is measured at each node by itself.

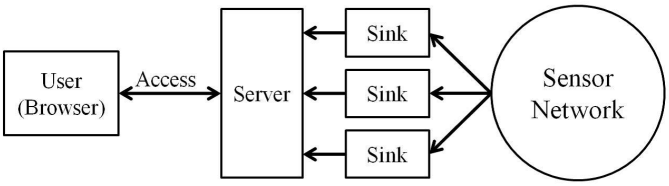


Figure 1: System Structure

(a) Reception Time at Sink

(b) Generated Time of Sensed Values

(c) Sink Node ID

(d) Source Node ID

(e) Sequence ID

(f) Parent Node ID (Next-hop of Source Node)

(g) Number of Transmitted Frames per Unit Time

(h) Number of Received Frames per Unit Time

(i) Number of Transmitted ACKs per Unit Time

(j) Number of Received ACKs per Unit Time

(k) Number of Transmitted Control Messages per Unit Time

(l) Number of Received Control Messages per Unit Time

(m) Accumulated Awaking Time per Unit Time

(n) Accumulated Sleep Time per Unit Time

(o) Residual Power

(p) Sensor Coordinate (If device is with GPS)

Most items above are well-defined but we would add an explanation for several items. Note that these administrative values are added to data packets only when a sensor value is measured and the corresponding data packets are generated, and not added at the relay nodes.

Item (e) is a value uniquely assigned to each packet by a node, which is used to check the loss of data packets. Items (g)-(l) are the values measured per unit time, where typically counting is done after previous sensed-value generation since we assume sensing is done periodically at each node. Note that the number of transmissions (g)(i)(k) include not only the frame generated at the node but also the frame that the node relays to sinks. Item (o) represents residual power at each node when the packet is generated.

Sensor location (p) may be collected at sink, but we have to consider that many sensor-node devices do not have GPS due to large power consumption, and also due to relatively large errors in computing positions.

Next, we describe the reasons for our choosing these values, and the necessity. Items (a)-(e) are the most basic values for understanding the flow of each packets, and are essential in network management. Item (f) is necessary to grasp delivery tree of the packet. Items (g)-(j) are also absolutely necessary to grasp communication state in each node in every unit

time, such as how much communication is successful. Items (k), (l) are important index to confirm that operation of routing protocols is correct. If abnormality of protocols are suspected, a success rate of transmission and reception of these control messages will be important information for making a decision. Items (m), (n), (o) are necessary for grasping state and detecting troubles about power of sensor nodes. In WSN that the remaining battery is essential measure, these items are high importance, because these are closely related to power consumption. Item (p) is required, when we collect positional information by GPS attached to sensor nodes. As explained above, minimal information that is required to grasp state of WSN are contained in the 16 administrative values which are treated by this system.

From the items (a)-(p) above, we compute several values useful for managing WSNs shown as follows.

(q) Packet Loss Ratio of the Next-hop Link per Unit Time

(r) Delivery Delay

(s) Elapsed Time after Last Frame Reception from Each Node at Sink

Item (q) represents a quality of next-hop link that is computed from the transmission count (g) and the ACK reception count (j) using the formula (q)= $\{((g)-(j))/(g)\}\times 100[\%]$. Item (r) is the average time of packets taken to travel to sinks from the packet is generated. This value is computed as an average of (r)=(a)−(b) for each packet. Item (s) implies a potential anomaly if it is far larger than the sensing time interval. This value is computed from item (a): if we let a_n be the arrival time of a packet with sequence number n, (s)= $a_n - a_{n-1}$.

Note that those three values (o), (q)-(s) are especially important in managing WSNs since worse values immediately imply performance degradation of WSNs. Thus, in our system, we set a threshold value for each of (o), (q), (r), and (s) so that the system can notify administrators of the abnormal state through the *alert table*.

4.3 User Interface

4.3.1 Transition of Views

The transition of user's view in our system is shown in Fig. 2. In the top view, the delivery tree is displayed to show the overview of the current state of the WSN. The administrative values table and the alert table are aside of delivery tree to show the specific data values and important alerts to notice. In the delivery tree view, the delivery tree of a specific time point is displayed with several important information items. In the administrative values table, list of administrative values received at sinks are displayed to check the specific values and states of WSNs. In the alert table, to help being aware of important events and states of WSNs, list of automatically detected events are displayed.

At the top of each item in administrative values table, we placed a button that pop-up a new window and display the

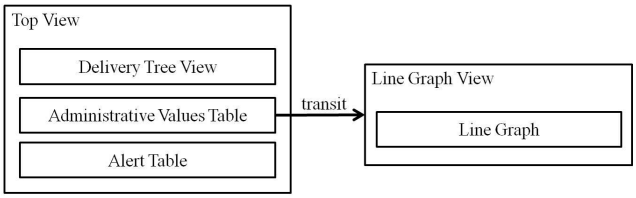


Figure 2: User Interface

line graph of the specified data item. Line graphs enable administrators to understand the tendency of value transition intuitively. In combination of those four components, administrators can watch the state of WSNs and explore the detailed behavior to find what is going on under troubles.

4.3.2 Delivery Tree View

Delivery tree view displays the delivery tree at the specific time point. Normally it would display the latest tree, but user can specify arbitrary time to see the tree at that time point. Delivery tree basically consists of a set of nodes which are placed at the right position and the set of next-hops to which each node forwards packets destined to sinks. In managing WSNs, we heve to know the place of each node, so we assume that the coordinate of each node is known by some mean, for example, GPS equipped to each sensor node, or static map that include the coordinate of each node. To show the transition of the delivery tree in time, our view has a function to overlay the past or the future delivery tree over the tree of the current time point. See Fig. 3 for this overlaid view. The current delivery tree is shown with solid blue lines whereas the past tree with dotted pink lines. The past tree consists of the previous next-hop links that are different from the current one and are reported not before the time $c - t$ where c is the current time and t is a predetermined threshold. Overlaying the future delivery tree is done in the similar way. By labeling links with the time reported (i.e., the generation time of reported packets), administrators can understand the transition of trees with exact reported time.

4.3.3 Administrative Values Table

The administrative values table shows all the administrative values collected at sinks per node. We show an example of the administrative values table in Fig.4. By clicking a node in the delivery tree, the administrative values table is updated to show the values sent from the clicked node around the time specified in the delivery tree view. With the administrative values table, administrators can refer the required administrative values intuitively and efficiently from large amount of data. Also, the button on top of each column invokes the line-graph window to see the data values intuitively.

4.3.4 Alert Table

The alert table has a role to keep administrators being aware of the important signs of WSN state changes. Especially, performance changes are essentially important to notify since they are invisible in the delivery tree so that not easy to notice

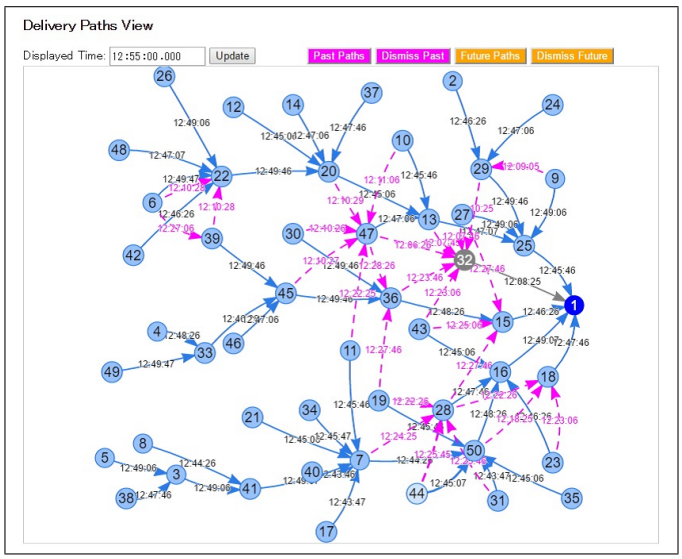


Figure 3: Delivery Tree View (When Two Time-points are Overlayed)

for administrators. To thid end, as mentioned in Sec.4.3.2, we display alert messages on communication performance by applying a threshold for each data items (o), (q), (r), and (s). An example of the alert table is shown in Fig. 5.

4.3.5 Line Graph View

The line graph shows the values listed in the administrative values table in the line graph fashion to enable administrators to grasp the trend of values intuitively. An example of the line graph view is shown in Fig. 6. As shown in this example, administrative values are plotted in time series where the horizontal axis is the reported time of the value.

5 EVALUATION

5.1 Implementation

The proposed system is implemented as a web application implemented using javaScript and AJAX. We use a library vis.js[19] for graph visualization, and highcharts[20] for drawing line graphs. In our system implementation, our application runs on Apache ver.2.4.6[21].

5.2 The Evaluation Method

We run simulation of WSNs to obtain a trace from a self-designed scenario, and visualize it using the proposed system. For the simulation, we implement a receiver-initiated MAC protocol combined with a low-power routing protocol proposed in [6]. We implement these protocols on Contiki[22], which is an OS for sensor network devices, and simulated it over simulator Cooja included in Contiki OS package. As a simulation scenario, we placed 50 nodes at a random coordinate in a 200[m]×200[m] rectangular field, and we placed a sink node on the central position of the left side of the field. Each node generates a sensed value in 5[min] interval, and

Rcv .Time	Seq .No	Next hop	#Data Rcv	#Data Tmt	#Ack Tmt	#Ack Rcv	#Ctl Tmt	#Ctl Rcv	#Data Rcv (Direct)	#Ack Rcv (Direct)	Data Gen.Time	Awake Time	Sleep Time	Residual Power(J)	Packet Loss(%)	Latency (sec)
11:07:002	0	7	0	0	0	0	8	52	0	0	8:13.	5:20.	2:28.	26993.16	0[%]	2:54
14:26:306	1	7	1	0	0	1	8	53	0	1	13:13.	5:21.	7:46.	26993.04	0[%]	1:13
20:26:427	2	7	2	0	0	2	8	54	0	2	18:13.	5:23.	12:25.	26992.9	0[%]	2:13
29:06:750	4	7	4	0	0	4	16	107	0	4	28:13.	9:59.	17:49.	26986.83	0[%]	53
34:26:330	6	7	5	0	0	5	23	162	0	6	33:13.	15:19.	17:49.	26979.95	0[%]	1:13
39:47:494	6	7	6	0	0	6	25	163	0	6	38:13.	17:19.	20:29.	26977.32	0[%]	1:34
43:46:967	7	7	7	0	0	7	25	164	0	7	43:13.	17:20.	25:47.	26977.2	0[%]	33

Figure 4: Administrative Values Table

Reception time	Generated time	Source Node ID	Next-hop Node ID	Status	Misc.
12:10:29:071	9:29.	20	47	Unstable Links	Packet Loss Ratio:20%
12:29:06:517	24:29.	20	13	Unstable Links	Packet Loss Ratio:11.5%
12:45:06:654	44:29.	20	13	Low Residual Battery Power	Residual Battery:26968.6[J]
12:26:25:914	25:18.	25	1	Unstable Links	Packet Loss Ratio:11.1%
12:30:26:334	30:18.	25	1	Unstable Links	Packet Loss Ratio:10%
12:35:06:376	34:14.	29	25	Unstable Links	Packet Loss Ratio:10%
12:08:25:700	8:5.	32	1	Long Time Without	Interval:51min35sec
12:14:26:909	12:50.	41	7	Unstable Links	Packet Loss Ratio:12.5%

Figure 5: Alert Table

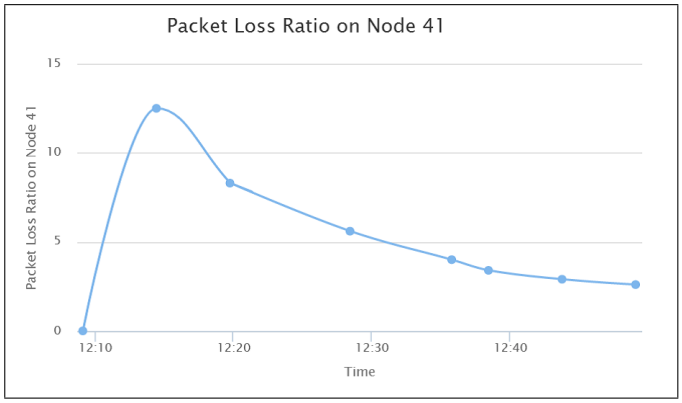


Figure 6: Line Graph View

sends periodical beacons in 40[sec] interval. We run the simulation in 60[min], but we intentionally invoke a node failure about the middle of the simulation time. Other parameters related to the simulation is shown in Table 1. As a result of simulation, we obtained a set of trace data set, i.e., the set of administrative values collected at the sink.

As for the parameters on the proposed system, we used the threshold values to generate alert messages in Table 2. These values are determined through previously performed test runs.

Evaluation process is in the following. We prepare the proposed system that loaded the above trace data set. We asked five subject, who are students who have studied both the Internet management and sonsor network protocols, to use the system for 20 minutes and also asked to list up the events that is important from the viewpoint of three management aspects, i.e., structure, failure, and performance management. After the above operations, we carefully examined the trace data set to find all the events that administrators should be aware of. We confirmed whether the subject can find all the events or not.

5.3 Results

By examining the events that the subjects found, we evaluate the practical efficacy of the proposed system. First of all, we say that the subject found all the events that we judged administrators should be aware of. This result shows that the proposed system works well and useful in managing WSNs.

Table 1: WSN Environment

Simulation Start Time	12:00
Simulation Time	60[min]
Field Size	200[m]×200[m]
Number of Nodes	50
Communication Range	Circle with Radius 50[m]
Beacon Interval	40[sec]
Sensing Interval	5[min]
Battery Capacity	27000[J]

Table 2: Threshold for Alert Table

Packet Loss Ratio	10%
Delivery Delay	60[sec]
Elapsed Time after Last Frame Reception	30[min]

In the following, we see the detailed description seen from the three aspects of network management, i.e., structure, failure, and performance management.

5.3.1 On Node Failure (Failure Management)

All the five subjects indicated the failure of node 32, which we intentionally did in the middle of the simulation time. From hearing from the subjects, this event was found by watching the delivery tree view. Figure. 7 shows the delivery tree at some time point where the node failure is clearly seen; since packets from node 32 has not been reached the sink for a long time, child nodes of node 32 no longer exist and the color of node 32 changes. Simultaneously, this event is displayed in the alert table. In addition, by watching administrative values table of node 32, all the subjects confirmed that only one sensed packet of the node 32 has arrived at the sink node during the evaluation test.

In this way, all the subjects actually found failure of node 32 in our evaluation test using the alert table, the delivery tree view, and the administrative values table. From the above, the system provides several mechanisms to help administrators find node failure. Therefore we confirmed that the failure management was well performed using our system.

5.3.2 On Path Transition (Structure Management)

All the five subjects explained how and why delivery tree changes as time passes. This is also possible using delivery tree view. When the subjects saw the delivery tree just after failure of node 32, which is shown in Fig. 8, they found that all nodes whose next-hop was node 32 changed their next-hop one after one. Also, the subjects found that, node 20 observes far larger number of control frames than usual just after node 32 fails. Note that the deployed MAC and routing protocol [6] transmits far larger number of control messages in face of topology changes to speed up the tree reconstruction. The subjects observed this behavior of nodes, which also supports that the root cause that changes next-hops is failure of node 32.

Also, two of the subjects confirmed that nodes around node 28 changed their next-hops to avoid node 28 in the similar

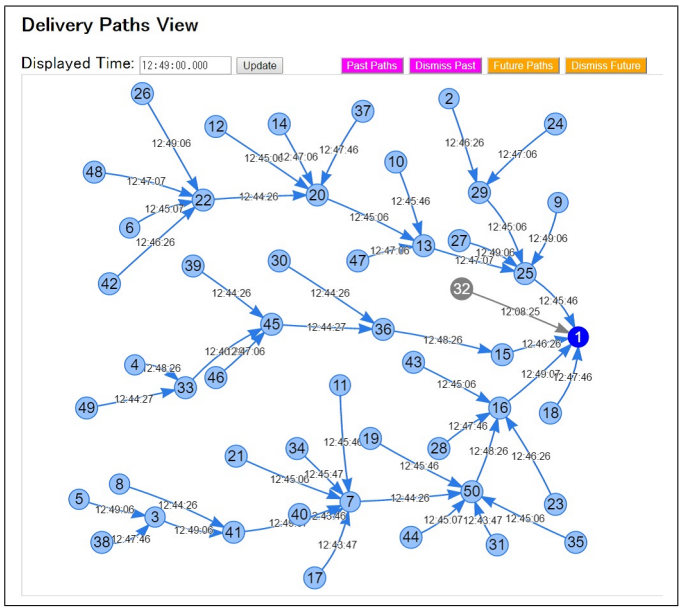


Figure 7: Delivery Tree View in Finding Node Failure

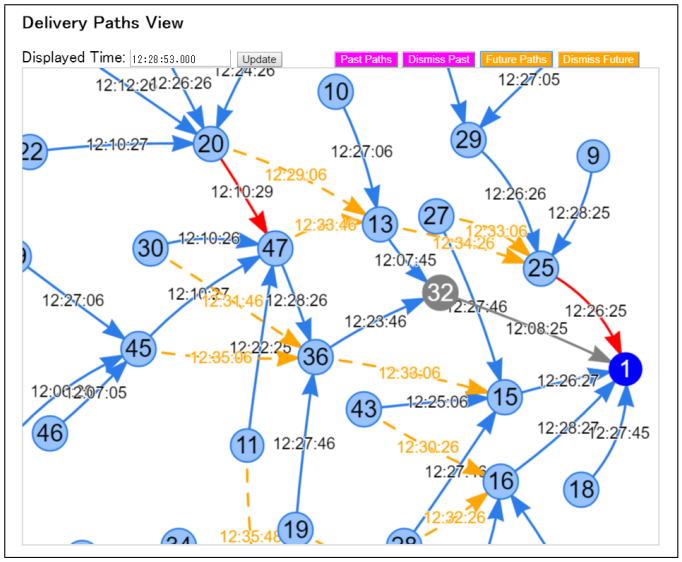


Figure 8: Delivery Tree View in Tree Transition

way to the case of node 32. In this case, however, node 28 did not fail. Instead, the subjects found that node 7, which is a child of node 28, has high packet loss ratio on the link to node 28. In this way, subjects could identify the cause of the delivery tree by means of referring the administrative values table. As above, we confirmed that the structure management was well performed.

5.3.3 On Performance Degradation (Performance Management)

As for the performance of the network, all the five subjects indicated that there are several nodes that had high frame loss ratio, which is found by the alert table and the delivery tree view. When the subjects examined the network state around node 20, several nodes adjacent to node 20 transmit control frames more frequently. So, the subjects naturally judges that

those control messages are the cause of the congestion and the frame loss.

Also, three subjects found that all children of the sink node had high packet loss ratio. It seems that the cause could be the failure of node 32 because node 32, which failed, was a child of the sink. However, the subjects found that the main cause was different from it. They found the packet loss ratio of node 18, which was also a child of the sink, was as high as 33% because the number of children of node 18, as well as its traffic, was large. They also found that, the number of children of node 18 decreases after that, as the children of node 18 change their next-hops. As a result, the collision among the children of the sink decreased, and the packet loss ratio improved accordingly. This indicates that the main cause was the concentration of traffic at node 18. As above, we confirmed that performance management was also well performed.

In summary, through the evaluation test, we found that the subjects performed all the tasks of failure, structure, and performance management. Consequently, we confirmed that the proposed system works well in a practical scenario of WSN management.

Additionally, note that our result implies that four basic components (i.e., delivery tree, administrative values table, line graphs, and alert table) are sufficient for a management tool of WSNs. Several visualization tools that have complicated views have been presented so far. However, from the viewpoint of functional design, the propose system can offer a new principle of WSN management tools.

6 DISCUSSION

In this section, we discuss two important issues over the proposed approach; one is (A) how the system can help administrators surely recover WSNs after detection of disorders, and the other is (B) on the overhead introduced by the administrative values attached to every packet.

First, we discuss (A), the function for failure recovery. Requirement to this system for managing WSNs is to make administrators surely notice disorders of networks quickly, and simultaneously to provide them the information that is essential in recovering the networks. We have various possible causes of disorders in WSNs such as node failure, communication failure due to severe interference or obstacles, and incorrect behavior of nodes due to buggy software. To enable administrators to recover those disorders, it is necessary for WSN management systems to provide the information that enables administrators to identify and specify the parts of the system that cause the disorders. However, because there are too many types of disorders that possibly occur, it is generally impossible to provide the information that guarantees to identify all cases and specifies the root causes of disorders. On the other side, in our evaluation scenario, it is shown that administrators could detect performance degradations due to node failure and congestion, and also that they could grasp the network state that the administrator should be aware of in order to understand what happens in the WSN. Although this evaluation results do not guarantee the ability of our system to deal with all kinds of disorders, we actually showed that the proposed system has ability to handle several typical cases of disorders that occur in WSNs.

Next, we explain about (B) how much the overhead of administrative values influences on communication performance. Since WSNs are generally discussed without thinking of the concept of management operations, packets sent to sink are usually supposed to include only the minimum information such as the value sensed at each sensor, meaning that the packet size is considered very small. However, to enable administrators to manage WSNs properly so that they can specify the cause of most of troubles, we need at least 16 administrative values as discussed in section 4.2. Therefore, we discuss in the following whether the overhead needed for the management affects the behavior of WSNs. First, as a typical example of the packet sizes of the 16 items listed in section 4.2, we assume that items (a), (b) spends 8 bytes, (m), (n) do 4 bytes, (p) does 8 bytes, and the others do 2 bytes. The total size of them is 54 bytes even when we require positional information obtained from GPS attached to sensor nodes. On the other side, in recent years, the typical transmission speed in WSNs is generally around 250 Kbps, as it is also proposed in several standards such as IEEE802.15.4. Consequently, the overhead calculated as $54 \times 8 = 432$ bits is approximately 0.17 % of the transmission speed. Although nodes near the sink needs to transfer a large number of packets, it is surely expected that the overhead would hardly increase congestion as well as worsen communication performance, because sensor nodes in most cases would have sufficient sleep time to save their battery power to work as members of the WSN. On the other hand, from the viewpoint of power consumption, we cannot deny the negative influence by forcing additional 54 bytes, which may increase packet size by several times as the original size, even though we require the overhead of beacons and control messages to compute forwarding paths with a routing protocol. However, we would again emphasize that we must accept the administrative values as overhead required for management operations in WSNs.

7 CONCLUSION

In this paper, we proposed a system to manage WSNs that deploy low-power MAC protocols. We designed the system from the viewpoint of the network management in the Internet, i.e., we aim at executing structure, failure, and performance management. We evaluated the proposed system using a simulation trace data set. As a result, all the events that should be noticed are listed up by the subject, while all three management aspects are well managed, showing that the proposed system possibly works effectively in the real WSN management operations.

To apply the system to the real environment instead of simulation is one of the important task for the future.

ACKNOWLEDGMENT

This work is supported by JSPS KAKENHI (15H02691).

REFERENCES

[1] IEEE802.15.4, http://www.ieee802.org/15/pub/TG4.html (referred November 21, 2015).

[2] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," In Proc. of SenSys'04, pp.95–107, (2004).

[3] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks," In Proc. of SenSys'06, (2006).

[4] Y. Sum, O. Gurewits, and D. B. Johnson, "RI-MAC: A Receiver-initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks," In Proc. of SenSys'08, pp.1-14, (2008).

[5] P. Huang, C. Wang, and L. Xiao, "RC-MAC: A Receiver-Centric MAC Protocol for Event-Driven Wireless Sensor Networks," In Proc. of IWQoS'10, (2010).

[6] S. Kojima, T. Yoshihiro, "A Low Management Cost Wireless Sensor Network Based on Receiver Initiated MAC Protocols," Journal of Information Processing, Vol.57, No.2, pp.480-493, (2016) (In Japanese).

[7] X. Fafoutis, A.D. Mauro, M.D. Vithanage, and N. Dragoni, "Receiver-initiated medium access control protocols for wireless sensor networks," Computer Networks, Volume 76, Pages 5574, (2015).

[8] SNMP Standard, http://www.snmp.com/ (referred December 6, 2015).

[9] N.Ramanathan, K.Chang, R.Kapur, L.Girod, E.Kohler, and D.Estrin "Sympathy for the Sensor Network Debugger," In Proc. Sensys'05, (2005).

[10] K. Liu, M. Li, and Y. Liu, "Passive Diagnosis for Wireless Sensor Networks," IEEE/ACM Transactions on Networking, Vol.18, Issue 4, pp.1132–1144, (2010).

[11] B.Parbat, A.K.Dwivedi, and O.P.Vyas, "Data Visualization Tools for WSNs: A Glimpse," International Journal of Computer Applications, Vol.2, No.1, pp.14–20, (2010).

[12] K. Liu, Q. Ma, Xibin Zhao, and Yunhao Liu, "Self-Diagnosis for Large Scale Wireless Sensor Networks," In Proc. INFOCOM'11, (2011).

[13] M. Khan, H. Le, H. Ahmadi, T. Abdelzaher, and J. Han, "Dustminer: Troubleshooting Interactive Complexity Bugs in Sensor Networks," In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp.99-112, (2008).

[14] X. Li, Q. Ma, Z. Zhicao, K. Liu, and Y. Liu, "Enhancing Visibility of Network Performance in Large-scale Sensor Networks," In Proc. of 34th International Conference on Distributed Computing Systems (ICDCS'14), (2014).

[15] X. Miao, K. Liu, Y. He, D. Papadias, Q. Ma, and Y. Liu, "Agnostic Diagnosis: Discovering Silent Failures in Wireless Sensor Networks," IEEE Transactions on Wireless Communications, (2013).

[16] R.Jurdak, A.G. Ruzzelli, A. Barbirato, and S. Boivineau, "Octopus: Monitoring, Visualization and Control of Sensor Networks" Wireless Communications and Mobile Computing, Vol. 11, Issue 8, pp. 1073–1091, (2011).

[17] E. Karapistoli, P. Sarigiannidis, and A.A. Economides, "SRNET: A Real-time Cross-based Anomaly Detection and Visualization System for Wireless Sensor Networks," In Proc VisSec'13, (2013).

[18] Information-technology Promotion Agency, Japan (IPA), "Knowledge for Network Management I," Open Source Software Model Curriculum Version 1, https://jinzaiipedia.ipa.go.jp/wp-content/uploads/oss/basic_Guidance_12.pdf (referred December 5, 2015).

[19] vis.js, http://visjs.org/ (referred January 15, 2016).

[20] Highcharts, http://www.highcharts.com/ (referred January 15, 2016).

[21] The Apache Software Foundation, http://www.apache.org/ (referred December 10, 2015).

[22] Contiki, http://www.contiki-os.org/ (referred February 1, 2016).

(Received October 20, 2016)
(Revised February 3, 2017)

Yuki Urata received the B.E. degree from Wakayama University in 2016. He is currently a Master-course student in Wakayama University.

Takuya Yoshihiro received his B.E., M.I. and Ph.D. degrees from Kyoto University in 1998, 2000 and 2003, respectively. He was an assistant professor in Wakayama University from 2003 to 2009. He has been an associate professor in Wakayama University from 2009. He is currently interested in the graph theory, distributed algorithms, computer networks, medial applications, and bioinformatics, and so on. He is a member of IEEE, IEICE, and IPSJ.

Yutaka Kawahashi received his the B.S. degrees from Wakayama University, Japan in 1993, and received the M.S. degrees from Nara Institute Science and Technology, Japan in 1995, and received the Ph.D. degrees from Graduate School of Engineering, Osaka Prefecture University, Japan in 2013. He is an Assistant Professor at Center for Information Science, Wakayama University, Japan. His research interests include Internet architecture, network management and network security. He is a member of WIDE Project.

Scalability Analysis of Exploration with Micro-Robots for Search in Rubble

Yuki Koizumi, Minoru Harada, and Toru Hasegawa

Graduate School of Information Science and Technology
{ykoizumi, m-harada, t-hasegawa}@ist.osaka-u.ac.jp

Abstract - To utilize a huge number of small robots, called *micro-robots*, for survivor searches in rubble in disaster areas is considered as a promising approach because of their smallness. How many micro-robots should be deployed is one of crucial research issues for survivor searches with micro-robots. In this paper, we derive theoretical lower bounds of the number of deployed micro-robots for accomplishing their search mission by modeling rubble as a graph and drawing orbits of micro-robots as paths on the modeled graph. As the first step to analyze difficulties in search in rubble with micro-robots, we focus on two relations: one is the relation between the number of deployed micro-robots and the sizes of rubble and the other is relation between the number of deployed micro-robots and abilities of their locomotion. Comparisons between the theoretical lower bounds and simulated results of the number of deployed micro-robots imply that searches in rubble with micro-robots becomes difficult as the rubble size in vertical directions becomes large.

Keywords: Search in Rubble, Micro-Robot, Graph-based Analysis

1 INTRODUCTION

As various robots have been developed [1]–[4], attempts to use robots for survivor searches in rubble in disaster areas are studied [5]. These robots can enter dangerous areas for humans and perform their tasks. Moreover, using robots may reduce risks of further endangering the survivors and rescuers due to secondary disasters. For these reasons, robots are expected as one of promising means for search and rescue in disaster areas. Since smallness of robots allows them to break into small gaps in rubble, it is more likely to find survivors buried inside the rubble. Therefore, to utilize smaller robots, which are often called *micro-robots* [6]–[8], for survivor searches in disaster areas is considered [9]–[11].

Searches in rubble with micro-robots are generally performed on the basis of deployment of enormous micro-robots because of the following two reasons: one is a simple structure of micro-robots and the other is a complex structure of rubble. Since it is difficult to install many functions on small micro-robots, they have minimum capabilities for searching, such as moving and detecting obstacles and survivors [9]. Thus, one approach for achieving a fast completion of the search is deploying enormous micro-robots and searching many possible spaces simultaneously [10]. As another point of view about the difficulty in searches in rubble with micro-robots, Cho and Arnold [11] pointed out a possibility that micro-robots may fall into holes surrounded with debris inside rubble and they cannot escape from the holes. Therefore, enormous micro-robots have to be deployed so that they accomplish the search in rubble even if a certain amount of them fall into such holes.

To determine how many micro-robots should be deployed to complete a search in rubble with micro-robots is one of crucial issues since the searches are performed with a huge number of micro-robots. However, few researches analyze the number of deployed micro-robots to complete the search in rubble. We simply refer to the number of deployed micro-robots to complete the search in rubble as the number of deployed micro-robots, hereafter. In [10], a method to reduce the number of deployed micro-robots is introduced but the target of the research is two-dimensional spaces with several obstacles. However, the number of deployed micro-robots will strongly depend on complexity of three-dimensional structures of rubble since many holes exist inside rubble of collapsed buildings, as Cho and Arnold pointed out in [11]. In this paper, we use the number of deployed micro-robots for indicating one of the difficulties in search in rubble with micro-robots and analyze the number of required micro-robots in a three-dimensional structure of rubble.

As the first step to understand difficulty in searches in rubble with micro-robots, we theoretically derive the *minimum number of deployed micro-robots* to complete the searches under an ideal case where all micro-robots move optimally in the rubble on the basis of the complete information about the internal structure of the rubble. We model a pile of rubble as a graph and get the minimum number of micro-robots by deriving the minimum number of paths, which correspond to orbits of the micro-robots, to cover all vertices in the modeled graph. Then, we conduct several simulation experiments assuming that currently developed micro-robots having capabilities of moving horizontally and detecting humans are used. By comparing the theoretical and simulation results, we discuss how searches in rubble with the currently developed micro-robots is difficult.

The rest of this paper is organized as follows. In Section 2, we define the problem of searches in rubble with micro-robots. Then, a method to model a pile of rubble with a graph and a method to derive the theoretical minimum number of deployed micro-robots are described in Section 3 and 4, respectively. We compare the number of deployed micro-robots in the case of the primitive micro-robots with the theoretical results in Section 5. Finally, we conclude this paper in Section 6.

2 PROBLEM FORMULATION OF SEARCH IN RUBBLE WITH MICRO-ROBOTS

In this section, we define the problem of searches in rubble with micro-robots.

2.1 Micro-robots and Searches in Rubble

This subsection describes micro-robots and searches in rubble.

1) Micro-robots: We assume that low-cost and small, cm-scale, micro-robots are used for searches in rubble in this paper. The micro-robot is equipped with 1) a sensor device to detect humans, 2) a moving mechanism to move inside rubble and 3) a device for notifying findings of survivors. Regarding the human detection in rubble, we assume that micro-robots use a temperature sensor or carbon a dioxide sensor to detect humans. Regarding the moving capability of micro-robots, we assume that they have the capability of moving in horizontal directions since many currently developed millimeter-scale micro-robots have the capability of crawling ground [6]–[8]. That is, we do not assume that micro-robots have the moving capability toward the upward direction in rubble. Finally, regarding the notifications of the survivor detection, we assume that micro-robots have wireless communication functionality like Wi-Fi or Bluetooth and they can notify the finds of survivors from anywhere inside the rubble to outside the rubble. Hence, we do not make further consideration about notifications of the survivor detection, hereafter.

2) Searches in Rubble: A micro-robot finds a survivor if and only if it detects the survivor by using the installed sensor device, i.e., the micro-robots need to approach close enough to the survivor. To find all survivors buried in the rubble, micro-robots need to arrive at all possible *spaces*, which are defined as places without any pieces of rubble, at least once, since no knowledge about locations of survivors is available in advance of the search.

Micro-robots have only the horizontal movement capability. Therefore, once they drop vertically into the lower space, they can never go back to the upper spaces. Since the micro-robots have no way to avoid dropping into *holes*, which are defined as spaces surrounded with walls of debris, micro-robots that drop into holes no longer continue their search missions. That is, it is practically impossible for one micro-robot to reach all spaces in rubble and find all survivors buried there. Hence, for search in rubble with micro-robots, it is necessary to deploy multiple micro-robots and to search all possible spaces simultaneously.

2.2 Building Artificial Rubble

In this section, we explain a method for building rubble to analyze the difficulty in searches in rubble with micro-robots. We refer to rubble built with our method as *artificial rubble*.

We model rubble by piling small and equal-sized cuboids, which are the minimum units of artificial rubble, in a three-dimensional Euclidean space. We divide the three dimensional Euclidean space into small sections. We refer to the three-dimensional Euclidean space and the sections there as *field* and *cells*, respectively. These cuboids represent small blocks of rubble. That is, the cuboids are building blocks of the artificial rubble and we build the complex structure of rubble by piling the cuboids. In this paper, we use a cuboid with sides of 10, 10, and 5 cm for constructing the artificial rubble. The sizes of cells and cuboids are the same. We construct artificial rubble by placing the cuboids into cells. We refer to cells filled with the cuboids, which represent objects of rubble, as *rubble cell* and other cells except for rubble cells as *empty cells*. That is, empty cells represent spaces inside rubble. We handle rubble as a field of given *width*, *depth*, and *height*. To simplify notations, we refer to the lengths in the x, y, and z axes directions in the Euclidean space as width, depth, and height, respectively. Micro-robots discussed above can stay in empty cells having a rubble cell underneath them. We refer to these cells, where micro-robots can stand as *plane cells*. They can move horizontally on plane cells that are adjacent each other on the same height of the field. We refer to a set of adjacent plane cells with the same height as a *plane*.

Figure 1 depicts the overview of constructing the artificial rubble. At the initial step, we fill a field of given width, depth, and height with empty cells. That is, the field is empty at the initial step. Then, we fill the field with rubble cells from the bottom toward the top step by step. At each layer of the field, rubble cells are placed until a given ratio, which we call *rubble ratio*, of cells are filled with rubble cells, as shown in Fig. 1(a). At each layer, one cell is selected randomly among empty cells. Then, we make a cuboid of $i \times j$ cells starting from the selected cell and fill with rubble cells in the cuboids. The integer numbers i and j are uniformly distributed random numbers between 1 and 10. Note that we make one constraint where at least one cells underneath the cuboid must be a rubble cell to avoid the situation where rubble cells float. We construct a complex structure by combining small rubble cells in this way, as shown in Fig. 1(b).

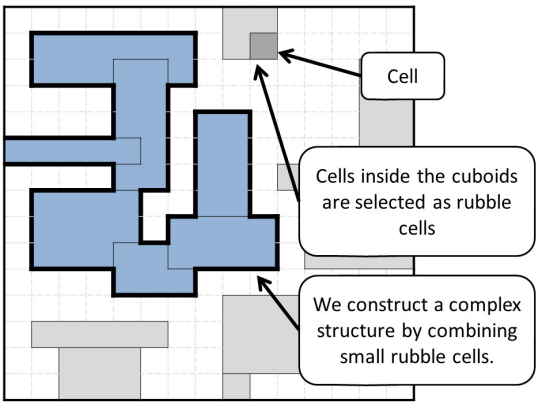
2.3 Definition of Searches in Rubble

This section defines the problem of searches in rubble with micro-robots. We define the completion of a search in rubble with micro-robots as the situation where all plane cells in artificial rubble are *covered* by micro-robots. A plane cell is covered when at least one micro-robot reaches the cell. Covering all plane cells means that micro-robots will find all survivors in the rubble. To cover all plane cells, a search in rubble with micro-robots are performed according to the following procedures. First, all micro-robots are deployed on top of the highest rubble cells. Each micro-robot moves independently from other micro-robots. Then, they move to one of adjacent empty cells of the same height in x or y axes directions per unit time. If the cell where micro-robots arrived at is an empty cell, they fall vertically to a plane cell right under the empty cell. Thus, if a plane is horizontally surrounded with rubble cells, they cannot escape from the plane. We refer to such planes as *holes*. We assume that micro-robots do not collide each other. Finally, we assume that micro-robots have enough battery capacity and therefore they do not stop moving till the completion of the search.

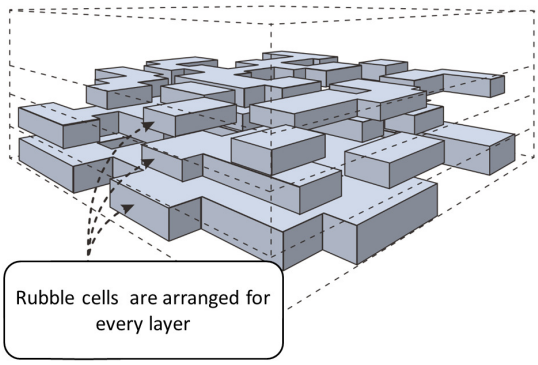
2.4 Difficulty in Searches in Rubble

This section explains a metric to indicate difficulty in a search in rubble with micro-robots and discusses a method for analyzing the relation between the size of rubble and the difficulty.

Since a micro-robot in a hole keeps staying there, all plane



(a) Decision of selection of rubble cells



(b) Example of artificial rubble

Figure 1: The overview of constructing artificial rubble

cells are not able to be covered by one micro-robot. Thus, we have to deploy multiple micro-robots to cover all plane cells. We expect that many micro-robots have to be deployed since many holes exist inside rubble of collapsed buildings. Therefore, the number of deployed micro-robots to complete the search is one of metrics to know how the search is difficult.

To measure the difficulty in the search in rubble with the number of deployed micro-robots, we compare results obtained by simulation with the theoretical lower bounds derived theoretically. In the simulation experiments, we assume that above-mentioned autonomous micro-robots are used, that is, micro-robots having only the horizontally moving capability move autonomously without any knowledge about rubble. We use the *minimum number of deployed micro-robots* to cover all plane cells as a metric to indicate the difficulty in the search in rubble. We simplify refer to the minimum number of deployed micro-robots as the minimum number of micro-robots, hereafter. That is, we use the minimum number of micro-robots as guidelines to measure the difficulty in the search.

To analyze the difficulty, we use a graph that represents a structure of the artificial rubble and derive the minimum number of micro-robots. We define the graph and a method to derive the minimum number of micro-robots from the graph in the following sections.

3 METHOD OF MODELING RUBBLE WITH A GRAPH

3.1 Overview

In this section, we define a search in rubble as a problem traversing a graph which corresponds to the artificial rubble constructed in the previous section. That is, we model the artificial rubble as a directed graph, of which vertices and directed edges correspond to places where micro-robots can be and paths where micro-robots can move, respectively. Then, we derive the minimum number of paths, which correspond to orbits of the micro-robots, to traverse all the vertices in the graph.

We model the artificial rubble as a directed graph according to the following two steps: 1) We simply express all possible places and paths where micro-robots can be and move as a directed graph and then 2) we simplify the graph by summarizing horizontal movements of micro-robots. More precisely, we first derive a graph G_1 by converting all empty cells, where micro-robots can be, as vertices and placing directed edges between all the possible vertices where micro-robots can move horizontally or fall vertically. That is, micro-robots can move from one vertex to another along directed edges in G_1. However, G_1 has many redundant information to know the minimum number of paths to cover all vertices, such as loops and unreachable vertices. Thus, we derive G_2 by summarizing several vertices in G_1 where micro-robots can move each other with horizontal movements to one vertex and removing unreachable vertices. That is, G_2 contains only directed edges that correspond to vertical movements of micro-robots. Consequently, the problem of a search in rubble is equivalent to covering all vertices in G_2 since all plane cell are contained in vertices in G_2 and the minimum number of micro-robots is equivalent to the minimum number of paths to cover all vertices in G_2. The following section mathematical defines the directed graphs and develops several heuristic algorithms to build the graphs.

3.2 Graph Formation

3.2.1 Formulating Rubble as a Graph

Before defining a directed graph, symbols used in this section are defined. The location of the cell in the artificial rubble is expressed by (i, j, k) $(i, j, k \in \mathbb{Z})$, which indicates i-th, j-th, and k-th cell in x, y, and z axis directions, respectively. The cell at (i, j, k) is represented as $c_{(i,j,k)}$. The function $C(i, j, k)$ is defined, which returns 0 if $c_{(i,j,k)}$ is an empty cell, return 1 if $c_{(i,j,k)}$ is a plane cell. X, Y, and Z are the width, depth, and height of the artificial rubble.

Let $G_1 = (V_1, E_1)$ be the directed graph, where V_1 and E_1 are the sets of vertices and edges. V_1 includes all plane cells in the artificial rubble and defined as follows:

$$V_1 = \{v_{(i,j,k)} \mid (i,j,k) \in I \times J \times K \ \wedge \ C(i,j,k) = 0 \\ \wedge \ (k = 1 \vee (k \neq 1 \wedge \ C(i,j,k-1) = 1)) \\ \vee (i,j,k) = (0,0,Z+1)\}, \quad (1)$$

where I, J, and K represent ranges of x, y, and z axes, respectively. That is, I is represented as $I = \{x \in \mathbb{N} \mid 1 \leq$

$x \leq X\}$ and J and K are represented similarly. The condition $C(i,j,k) = 0 \wedge (k = 1 \vee (k \neq 1 \wedge C(i,j,k-1) = 1)$ represents that $c_{(i,j,k)}$ is plane cell. Therefore, V_1 contains all the plane cells in the artificial rubble. Micro-robots are deployed on top of the rubble. Thus, we add another vertex to express the deployment of micro-robots, *root vertex* v_0, and the root vertex is represented by the constraint $(i,j,k) = (0,0,Z+1)$.

Next, we define the set of directed edges E_1 in G_1. Directed edges express the movement of micro-robots from one plane cell to another plane cell with horizontal movements and free falls and are defined as follows:

$$\begin{aligned} E_1 = \Big\{ \big(v_{(i,j,k)}, v_{(l,m,n)}\big) \mid \\ (l,m,n) = (i \pm 1, j, k) \ \vee (l,m,n) = (i, j \pm 1, k) \\ \vee \ (((l = i \pm 1 \ \wedge \ m = j) \ \vee \ (l = i \ \wedge \ m = j \pm 1)) \\ \wedge \ \textstyle\sum_{h=n}^{k} C(l,m,h) = 0 \ \wedge \ k > n) \\ \vee \left((i,j,k) = (0,0,Z+1) \wedge \sum_{h=n}^{Z} C(l,m,h) = 0 \right) \Big\}. \end{aligned} \quad (2)$$

The condition $(l,m,n) = (i \pm 1, j, k)$ and $(l,m,n) = (i, j \pm 1, k)$ represents the horizontal movements and $(((l = i \pm 1 \ \wedge \ m = j) \ \vee \ (l = i \ \wedge \ m = j \pm 1)) \ \wedge \ \sum_{h=n}^{k} C(l,m,h) = 0 \ \wedge \ k > n)$ represents movements from one plane cell to another in a free fall through empty cells. To express the deployment of micro-robots, E_1 contains the edge that is link from the root vertex v_0 to vertices of plane cells of the top of the rubble and this is represented by the last condition in Eq. (2).

Next, we present a heuristic algorithm to derive the directed graph G_1 from given artificial rubble in Algorithm 1. First, the sets of vertices and edges V_1 and E_1 are initialized with empty sets at lines 1 and 2. From line 3 to 11, G_2 is constructed from the given artificial rubble according to the definitions in Eqs. (1) and (2). The sub-function SET_EDGE defined at lines 12 to 15 is the function to set edges between vertices.

Algorithm 1 Constructing $G_1 = (V_1, E_1)$

Input: The artificial rubble, $c_{(i,j,k)}$
Output: The directed graph, $G_1 = (V_1, E_1)$

```
1:  V1 ← ∅
2:  E1 ← ∅
3:  for k ← 1 to Z do
4:      for j ← 1 to Y do
5:          for i ← 1 to X do
6:              if C(i,j,k) = 0 ∧ C(i,j,k−1) = 1 then
7:                  V1 ← V1 ∪ {v(i,j,k)}
8:              end if
9:          end for
10:     end for
11: end for
12: for all v(i,j,k) ∈ V1 do
13:     SET_EDGE(v(i,j,k), c(i±1,j,k))
14:     SET_EDGE(v(i,j,k), c(i,j±1,k))
15: end for

16: function SET_EDGE(v(i,j,k), c(l,m,n))
17:     if C(l,m,n) = 0 then
18:         if n = 1 ∨ C(l,m,n−1) = 1 then
19:             E1 ← E1 ∪ {(v(i,j,k), v(l,m,n))}
20:         else
21:             z ← n
22:             while C(l,m,z) = 0 do
23:                 z ← z − 1
24:             end while
25:             E1 ← E1 ∪ {(v(i,j,k), v(l,m,z+1))}
26:         end if
27:     end if
28: end function
```

3.2.2 Summarizing a Graph

Next, we derive the graph $G_2 = (V_2, E_2)$ by summarizing several vertices in G_1 where micro-robots can move each other with horizontal movements to one vertex. That is, vertices in V_2 correspond to planes in the artificial rubble. A plane is denoted by $A_{(i,j,k)}$, which contains the plane cell $c_{(i,j,k)}$, and the constraint to aggregate plane cells and constructing a plane is expressed as

$$A_{(i,j,k)} = \{c_{(i,j,k)} \mid c_{(i,j,k)} \in A_{(i,j,k)s} \ \wedge \ A_{(i,j,k)s} = A_{(i,j,k)s-1}\}, \quad (3)$$

where $A_{(i,j,k)\,s}$ is defined using the following constraints.

$$A_{(i,j,k)\,1} = \{c_{(i,j,k)}\} \quad (4)$$

$$\begin{aligned} A_{(i,j,k)\,s} = \{c_{(l,m,n)} \mid c_{(l,m,n)} \in A_{(i,j,k)\,s-1} \\ \vee \ (c_{(l,m,n)} \in Q \\ \wedge \ \exists c_{(s,t,u)} \in A_{(i,j,k)\,s-1} \ ((l,m,n) = (s \pm 1, t, u) \\ \vee \ (l,m,n) = (s, t \pm 1, u)))\} \end{aligned} \quad (5)$$

$$\begin{aligned} Q = \{c_{(i,j,k)} \mid (i,j,k) \in I \times J \times K \ \wedge \ C(i,j,k) = 0 \\ \wedge \ (k = 1 \vee (k \neq 1 \wedge \ C(i,j,k-1) = 1))\} \end{aligned} \quad (6)$$

To express the relationship between a plane and a vertex v_a that constitutes the plane a, we introduce S_{v_a}, where S_{v_a} is a set of plane cells that constitute the plane a. S_{v_a} satisfies $\forall c_{(i,j,k)} \in S(v_a) \, \forall c_{(l,m,n)} \in S(v_a) \ (A_{(i,j,k)} \in A_{(l,m,n)})$. S_{v_0} is an empty set.

Next, we explain how a set of directed edges E_2 of directed graph G_2 is constructed. Edges in E_2 represent the vertical movements in free falls from one plane to the next plane. The edge from v_a to v_b must satisfy $\exists \, c_{(i,j,k)} \in S(v_a) \ \exists \, c_{(l,m,n)} \in S(v_b) \, (((l = i \pm 1 \ \wedge \ m = j) \ \vee \ (l = i \ \wedge \ m = j \pm 1)) \ \wedge \ k > n)$. Since micro-robots are deployed via the root vertex v_0, we connect the root vertex to all other planes which can reach from the top of the artificial rubble, i.e., directed edges are placed from v_0 to all v_a that satisfies the constraint $\exists \, v'_{(i,j,k)} \in S(v_a) \ \sum_{h=n}^{Z} C(l,m,h) = 0$.

Then, we present a heuristic algorithm to derive the directed graph G_2 from the given rubble and the directed graph G_1 built with Algorithm 1. In The algorithm consists of three parts: initializing variables from lines 1 to 3, constructing vertices from lines 4 to 14, and constructing edges from lines 15 to 25. To distinguish which plane vertices belong to, we assign identifiers to all plane and the identifier of plane where $c_{(i,j,k)}$ belongs to is stored to $c_{(i,j,k)}.id$. The initial value of $c_{(i,j,k)}.id$ is zero. From lines 4 to 14, each plane in the given artificial rubble is converted to a vertex. That is, all adjacent

Algorithm 2 Constructing $G_2 = (V_2, E_2)$

Input: $G_1 = (V_1, E_1)$ and $c_{(i,j,k)}$
Output: $G_2 = (V_2, E_2)$

1: $V_2 \leftarrow \emptyset$
2: $E_2 \leftarrow \emptyset$
3: $id \leftarrow 1$
4: **for** $k \leftarrow 1, Z$ **do**
5: **for** $j \leftarrow 1, Y$ **do**
6: **for** $i \leftarrow 1, X$ **do**
7: **if** $C(i,j,k) = 0 \wedge C(i,j,k-1) = 1$
8: a $\wedge c_{(i,j,k)}.id = 0$ **then**
9: $V_2 \leftarrow V_2 \cup \{v_{id}\}$
10: CLUSTER$(id, c_{(i,j,k)})$
11: id $\leftarrow id + 1$
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: **for** $k \leftarrow 1, Z$ **do**
17: **for** $j \leftarrow 1, Y$ **do**
18: **for** $i \leftarrow 1, X$ **do**
19: **if** $C(i,j,k) = 0 \wedge C(i,j,k-1) = 1$ **then**
20: $id \leftarrow c_{(i,j,k)}.id$
21: SET_EDGE2$(v_{id}, c_{(i\pm1,j,k)})$
22: SET_EDGE2$(v_{id}, c_{(i,j\pm1,k)})$
23: **end if**
24: **end for**
25: **end for**
26: **end for**

Algorithm 3 Sub-functions for constructing $G_2 = (V_2, E_2)$

1: **function** SET_EDGE2$(v_{id}, c_{(i,j,k)})$
2: $z \leftarrow k$
3: **if** $C(i,j,z) = 0 \ \wedge \ C(i,j,z-1) = 0$ **then**
4: **while** $C(i,j,z) = 0$ **do**
5: $z \leftarrow z - 1$
6: **end while**
7: $id2 \leftarrow c_{(i,j,k)}.id$
8: $E_2 \leftarrow E_2 \cup \{(v_{id}, v_{id2})\}$
9: **end if**
10: **end function**

11: **function** CLUSTER$(id, c_{(i,j,k)})$
12: **if** $C(i,j,k) = 0 \wedge C(i,j,k-1) = 1 \wedge c_{(i,j,k)}.id = 0$ **then**
13: $c_{(i,j,k)}.id \leftarrow id$
14: CLUSTER$(id, i \pm 1, j, k)$
15: CLUSTER$(id, i, j \pm 1, k)$
16: **end if**
17: **end function**

plane cells of the same height are aggregated toe one plane. To construct planes, we use sub-function CLUSTER, which aggregates plane cells recursively assign the identifier to the plane cells. From lines 15 to 25, directed edges are placed from one plane to another where micro-robots can move vertically in free falls. Finally, we construct G_2' by removing vertices that are unreachable from v_0 and edges to the removed vertices since micro-robots cannot reach such spaces in the rubble.

4 THE MINIMUM NUMBER OF MICRO-ROBOTS

The theoretical minimum number of micro-robots is equivalent to the number of paths in the minimum path cover of the modeled graph G_2'. A path cover is a set of directed paths such that every vertex in the graph belongs to at least one of the path. If the path cover consists of the minimum number of paths, the path cover is referred to as the minimum path cover. In this section, we first explain the reason why the minimum number of micro-robots is equivalent to the number of paths in the minimum path cover and then how to derive the minimum path cover of G_2'.

A search in the rubble with micro-robots is equivalent to traverses of vertices leaving from the root vertex along with directed edges on the directed graph derived from the rubble. Specifically, we have modeled a given pile of artificial rubble as a directed graph. Passing through a vertex in the graph is equivalent to surveying the space in the rubble that correspond to the vertex. In the similar way, moving along with a directed edge is equivalent to moving from one space to another in the rubble. Therefore, orbits of micro-robots in the rubble can be expressed as paths starting from the root vertex v_0 in the directed graph G_2'. The minimum number of deployed micro-robots is equivalent to the minimum number of paths, which are originating from v_0, to cover all vertices in G_2'.

Next, we derive the minimum number of paths originating from the root vertex v_0 for covering all the vertices in G_2'. A path P is defined as an ordered set of vertices. We denote a set of paths using $\mathcal{P}$, hereafter. The set of paths to cover all vertices is also called as *path cover* and it is defined as follows: $\mathcal{P}$ is a path cover of $G = (V, E)$ if $\mathcal{P}$ is a set of paths of G such that every $v \in V$ is included at least one path $P \in \mathcal{P}$ [12]. That is, a path cover must satisfy the following condition:

$$\forall v \in V \ \exists P \in \mathcal{P} \ \ v \in P. \tag{7}$$

The minimum path cover is a path cover $\mathcal{P}$ such that $|\mathcal{P}|$ is minimum.

The orbits of micro-robots can be expressed by paths but the paths must start with v_0 since micro-robots are deployed on top of the rubble. That is, the paths of micro-robots must satisfy the following constraint:

$$\forall P \in \mathcal{P} \ \ (v_0 \in \mathcal{P}) \ \wedge \ \forall \ v \in V \ \exists P \in \mathcal{P} \ \ (v \in P) \tag{8}$$

Such a path cover is called a single starting point path cover. Therefore, the constraints of paths of the path cover problem and the search in rubble with micro-robots are slightly different, as shown in Eqs. (7) and (8). The minimum single starting point path cover is a single starting path cover $\mathcal{P}$ such that $|\mathcal{P}|$ is minimum. Though algorithms to deriving the minimum path cover have been already developed but no algorithms to derive the minimum single starting point path cover.

If the number of elements of the minimum single starting point path cover is equal to that of the minimum path cover, the minimum number of micro-robots can be derived by solving the problem of the minimum path cover of G_2'. We prove

that the number of elements of the minimum single starting point path cover is equal to that of the minimum path cover as follows: Since for all vertices $v \in V_2'$ in G_2' at least one path that originates from v_0 and reaches to v exists, a set of paths $\mathcal{P}$ exists such that $\mathcal{P}$ satisfies the following condition, $|C| = |\mathcal{P}| \wedge \forall P \in \mathcal{P}\ (v_0 \in P) \wedge \forall P' \in C\ \exists P \in \mathcal{P}\ (P' \subseteqq P)$, where C is the minimum path cover of G_2'. That is, the set of paths $\mathcal{P}$ is the minimum path cover and the all paths in $\mathcal{P}$ start with v_0. The number of elements of the minimum path cover is equal to that of the minimum single starting point path cover. Hence, the minimum number of micro-robots can be derived by solving the problem of the number of elements of the minimum path cover of G_2'.

Finally, we explain a method to derive the minimum path cover in G_2'. G_2' is a directed acyclic graph (DAG) [12]. It is a well-known fact that the minimum vertex-disjoint path cover in DAG can be derived by solving the maximum matching problem by converting the DAG into a bipartite graph [13]. The minimum vertex-disjoint path cover is a set of the minimum number of elements in vertex-disjoint path covers. The vertex-disjoint path cover is a set of paths $\mathcal{P}$ such that for every $v \in V$ in G there exists at exactly one path $P \in \mathcal{P}$ including v. The vertex-disjoint path cover must satisfy another constraint that no paths in the set cannot share vertices in addition to the minimum path cover. However, if the target graph G is a DAG, the minimum path cover of G is equal to the minimum vertex-disjoint path cover of the transitive closure of the graph G_{clo} [12]. Therefore, we can have the minimum path cover of G_2' by deriving the minimum vertex-disjoint path cover of transitive closure of G_2'. In this way, the minimum number of micro-robots deployed to complete the search in the rubble can be derived.

5 DIFFICULTY IN SEARCH IN RUBBLE WITH MICRO-ROBOTS

In this section, we analyze the difficulty in searches rubble with micro-robots. First, we describe our simulation environments. Then, we analyze the number of deployed micro-robots and the minimum number of deployed micro-robots by change the width/depth and height of the artificial rubble.

5.1 Simulation Conditions

We set parameters of the artificial rubble as follows: To express complicated shape of rubble, we set the size of one cell to $10 \times 10 \times 5$ cm. Since we suppose that micro-robots search inside highly dense rubble, where a large robot cannot enter, we set the percentage of rubble cells in the artificial rubble to reasonably high, i.e., 0.65. In our simulation experiments, micro-robots are deployed randomly on top of the artificial rubble. In the rubble, the micro-robots move independently to each other according to the Lévy Flight mobility model [14], [15], which is know as an efficient mobility pattern to search for targets. Lévy Flight is a random walk where the step lengths during the walk are described by a heavy-tailed probability distribution. In the simulation experiments, the Lévy distribution is used to describe the step lengths of the walk. The micro-robots move horizontally on planes and they fall in free falls when they reach an empty cell. In this paper, we ignore situations where micro-robots stop working during the search due to several issues, such as failures of locomotion or sensor devices. We define that a plane cell is surveyed if at least one micro-robot enters the plane cell. The search will be finished within 24 hours since the probability of humans under rubble being alive decreases rapidly 24 hours after they are buried. We compute the number of deployed micro-robots to cover 90% of plane cells in the artificial rubble within 24 hours. We compute the average of results obtained from 30 simulation trials. In the following section, we investigate how the search in rubble is difficult by evaluating the number of deployed micro-robots by changing the size of the artificial rubble.

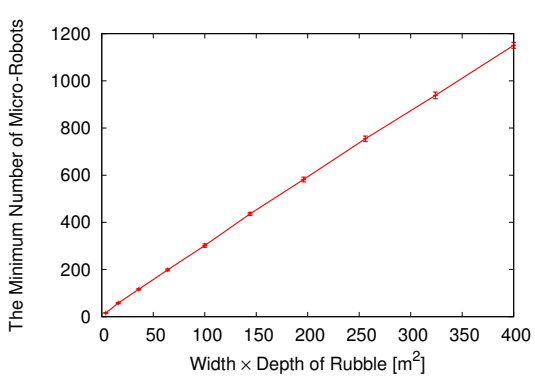
5.2 The Number of Deployed Micro-robots

First, we observe the effects of the area size, i.e., the width multiplied by the depth, of the artificial rubble on the number of deployed micro-robots. Figure 2 shows relations between the number of deployed micro-robots and the area size of the rubble. In this simulation, the height of the rubble is set to 1.5 meters. The horizontal axes of the figures are the area size, which is defined as the width multiplied by the depth. The minimum number of micro-robots derived theoretically is shown in Fig. 2(a) and the number of deployed micro-robots obtained through simulations is shown in Fig. 2(b). The error bars in Fig. 2(b) indicate the 95% confidence intervals. Note that comparing the absolute values of the analytical and simulation results is nonsense. We compare the tendency of the results in this paper. Both the minimum number of micro-robots derived theoretically and the number of deployed micro-robots obtained through simulations increase almost proportionally to the area size.

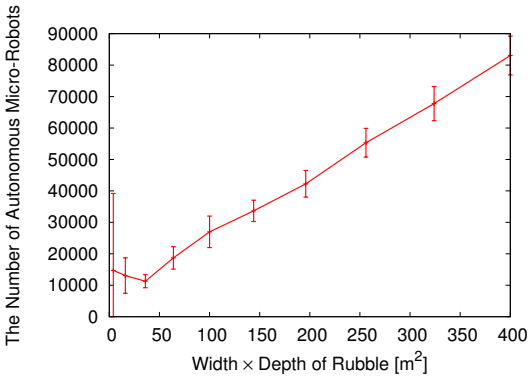
Next, we observe the effects of the height of the artificial rubble on the number of deployed micro-robots in Fig. 3. The horizontal axes indicate the height of the artificial rubble. The vertical axes are the same as those in Fig. 2. The minimum number of micro-robots derived theoretically is shown in Fig. 3(a) and the number of deployed micro-robots obtained through simulations is shown in Fig. 3(b). In this simulation, both the width and depth of the artificial rubble is set to 10 meters. In contrast to the minimum number of micro-robots derived theoretically, which increases almost proportionally to the height, the number of deployed micro-robots obtained through simulations increases more sharply.

5.3 Effects of Locomotion in Vertical Directions

One of the critical reasons why such huge number of micro-robots must be deployed for accomplishing searches in rubble is that micro-robots may fall into holes surrounded with debris inside rubble and they cannot escape from the holes, as Cho and Arnold have pointed out in [11]. One solution to resolve this issue is installing an ability of locomotion in vertical directions, such as jumping locomotion [16], on micro-robots. This section investigates how an ability of locomotion in vertical directions relaxes the difficulty in searches in rubble with

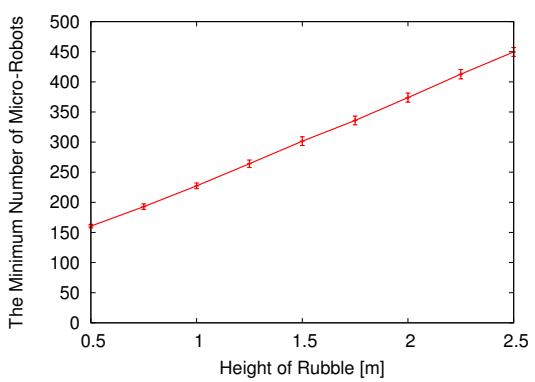


(a) The minimum number of micro-robots derived theoretically

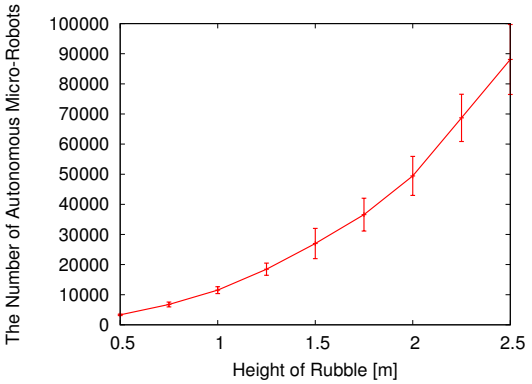


(b) The number of deployed micro-robots derived by simulations

Figure 2: The minimum number of micro-robots and the number of deployed micro-robots in the case that the area size of the rubble is changed



(a) The minimum number of micro-robots derived theoretically



(b) The number of deployed micro-robots derived by simulations

Figure 3: The minimum number of micro-robots and the number of deployed micro-robots in the case that the height of the rubble is changed

micro-robots.

As an ability of locomotion in vertical directions, we introduce *jumping locomotion*, which is proposed in several studies [9], [16]. Let us assume that micro-robots can jump h m high. That is, they can go to cells in h m higher places. We refer to the height of jumping locomotion as *jumping height*, in this section. We use the following settings regarding the behavior of micro-robots: when micro-robots arrive at a wall, where they cannot move without the jumping locomotion, they jump with the probability of 0.1; otherwise they continue to move in the opposite direction with the horizontal locomotion.

In this simulation, both the width and depth of the artificial rubble is set to 3 meters, which is smaller than the settings of simulations for Fig. 3, to focus on the effect of the jumping locomotion. Regarding other simulation settings, we use the same ones as those used in the previous section. Figure 4 shows the number of deployed micro-robots in the case that the height of the rubble is changed. To investigate effects of the jumping locomotion, we compare the number of deployed micro-robots without the jumping locomotion and that with jumping locomotion. We set the jumping height, h, to 0.05 m, in this simulation. Installing the jumping locomotion drastically reduces the number of deployed micro-robots. To investigate effects of the jumping height, we evaluate the number of deployed micro-robots with changing the jumping height, and the results are shown in Fig. 5. The results indicate that increasing the jumping height does not impact on the reduction in the number of deployed micro-robots. Finally, we compare the number of deployed micro-robots with the jumping locomotion derived via simulations (Fig. 6) with the minimum number of micro-robots derived theoretically (Fig. 3(a)). In contrast to the theoretical minimum number of micro-robots, which increases almost proportionally to the height, the number of deployed micro-robots with the jumping locomotion increases more sharply in the same way as micro-robots without the jumping locomotion (Fig. 3(b)).

5.4 Implications

Observations found through the evaluations are summarized as follows. The number of deployed micro-robots is mostly proportional to the size in the horizontal direction of each rubble, as shown in Fig. 2. In contrast, it steeply increases as the size in the vertical direction of rubble increases, as shown in Fig. 3. Installing the jumping locomotion function

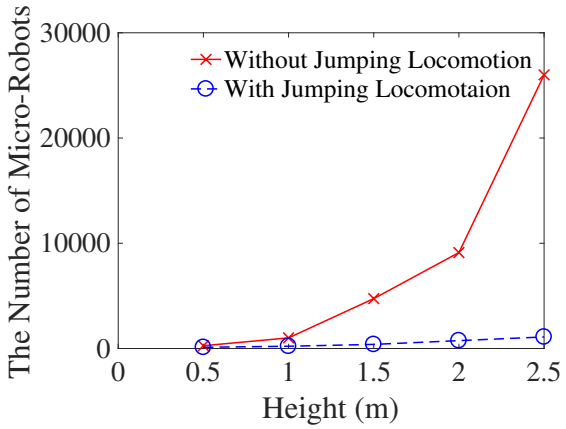


Figure 4: The number of deployed micro-robots in the case of micro-robots with and without the jumping locomotion

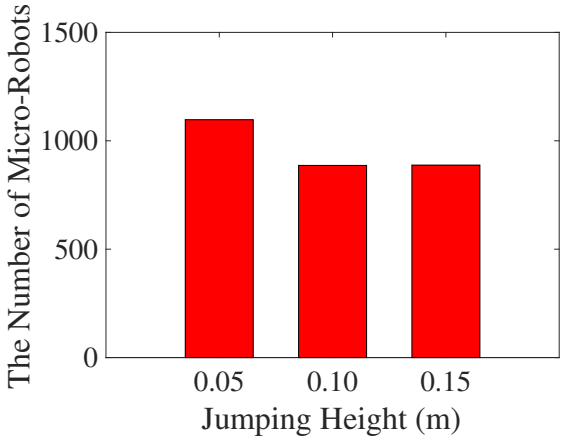


Figure 5: The effects of the jumping height on the number of deployed micro-robots

is a candidate to relax the difficulty in the searches in rubble with micro-robots, as shown in Fig. 4, which shows that the jumping locomotion function drastically reduces the number of deployed micro-robots. However, the results also indicate that the jumping locomotion function does not resolve the root cause of the difficulty fundamentally due to the fact that the number of deployed micro-robots with the jumping locomotion increases more sharply than the theoretical minimum number of micro-robots, which is almost proportional to the height of rubble.

Through simulation experiments, we have learned the following implications: First, searches in rubble with primitive micro-robots which have only the horizontal movement function as their locomotion function are very difficult since many micro-robots are required to accomplish the searches. Second, the search in rubble using micro-robots with the jumping locomotion function might be useful for searches in rubble if it is not very high. Nevertheless, using such micro-robots may not be a feasible solution in the case that the rubble is high, such as high buildings. Finally, we can roughly estimate the number of deployed micro-robots to accomplish a search in rubble of an ordinary dwelling house in Japan, using the observation found through the simulation, i.e., the number of deployed micro-robots is mostly proportional to the size in the horizontal

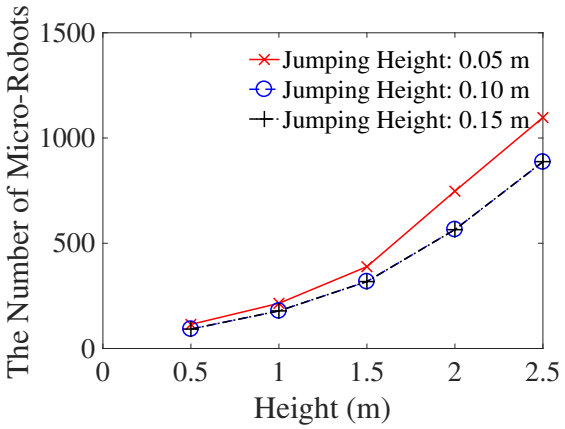


Figure 6: The number of deployed micro-robots in the case of micro-robots with the jumping locomotion

direction of each rubble, as shown in Fig. 2. Assuming that the average total floor size per a dwelling in Japan is 94.42 m^2 in 2013 [17] and the height of the rubble is about 2.5 m, the number of deployed micro-robots is roughly estimated at $9298.3 = 886.3 \times 94.42/9.0$ according to the result that 886.3 micro-robots are required for the search for 9 $m^2 \times 2.5$ m cuboid, as shown in Fig. 6. This estimation implies that micro-robots with the jumping locomotion function can be used for searches rubble of low dwelling houses in Japan.

6 CONCLUSION

In this paper, we analyze the number of deployed micro-robots to complete the search in the rubble. To investigate the difficulty in the search, we derive the theoretical lower bounds of the number of deployed micro-robots, which can be used as guideline to measure the difficulty in the search in the rubble. As the first step to analyze the difficulty in the search in the rubble, we compare the number of deployed primitive micro-robots, which have only the horizontal movement function as their locomotion function, with the minimum number of micro-robots derived theoretically by changing the area size and the height of the rubble. In contrast to the theoretical minimum number of micro-robots, which increases almost proportionally to the height, the number of deployed micro-robots obtained through simulations increases more sharply. Although installing the jumping locomotion to micro-robots drastically reduces the number of deployed micro-robots, the number of deployed micro-robots with the jumping locomotion also increases more sharply than the theoretical minimum number of micro-robots. This fact suggests that the jumping locomotion may not essentially resolve the root cause of the difficulty in searches in rubble, and hence searches in rubble with primitive micro-robots get difficult as the rubble gets large in vertical directions.

REFERENCES

[1] K. Osuka and H. Kitajima, “Development of mobile inspection robot for rescue activities: Moira,” in *Proceedings of the IEEE/RSJ International Conference on*

Intelligent Robots and Systems, vol. 4, pp. 3373–3377, (2003).

[2] M. Arai, Y. Tanaka, S. Hirose, H. Kuwahara, and S. Tsukui, "Development of "Souryu-IV" and "Souryu-V:" Serially connected crawler vehicles for in-rubble searching operations," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 31–65, (2008).

[3] J. Casper and R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, vol. 33, no. 3, pp. 367–385, (2003).

[4] B. Yamauchi, "Packbot: A versatile platform for military robotics," in *Proceedings of SPIE*, vol. 5422, pp. 228–237, (2004).

[5] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen, "Search and rescue robotics," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), ch. 50, pp. 1151–1173, Berlin, Heidelberg: Springer, (2008).

[6] E. Edqvist, N. Snis, R. C. Mohr, O. Scholz, P. Corradi, J. Gao, A. Dieguez, N. Wyrsch, and S. Johansson, "Evaluation of building technology for mass producible millimetre-sized robots using flexible printed circuit boards," *Journal of Micromechanics and Microengineering*, vol. 19, (2009).

[7] B. R. Donald, C. G. Levey, C. D. Mcgray, I. Paprotny, and D. Rus, "An untethered, electrostatic, globally controllable MEMS micro-robot," *Microelectromechanical Systems*, vol. 15, pp. 1–15, (2006).

[8] T. Ebefors, J. U. Mattsson, E. Kalvesten, and G. Stemme, "A walking silicon micro-robot," in *Proceedings of the 10th International Conference on Solid-State Sensors and Actuators*, pp. 1202–1205, (1999).

[9] S. Dubowsky, J. S. Plante, and P. Boston, "Low cost micro exploration robots for search and rescue in rough terrain," in *Proceedings of IEEE International Workshop on Safety Security and Rescue Robotics*, (2006).

[10] D. K. Sutantyo, S. Kernbach, P. Levi, and V. A. Nepomnyashchikh, "Multi-robot searching algorithm using lévy flight and artificial potential field," in *Proceedings of IEEE International Workshop on Safety Security and Rescue Robotics*, pp. 1–6, (2010).

[11] J. H. Cho and M. G. Arnold, "Survivor search using a quasi-2D-parallax algorithm with massive microrobot swarms," in *Proceedings of the 14th WSEAS International Conference on Systems*, pp. 522–525, (2010).

[12] M. Kowaluk, A. Lingas, and J. Nowak, "A path cover technique for LCAs in dags," in *Algorithm Theory-SWAT 2008*, pp. 222–233, Springer, (2008).

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (Instructor's Manual Second Edition)*, ch. 26. CreateSpace Independent Publishing Platform, (2014).

[14] G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev, S. Havlin, M. G. E. D. Luz, E. P. Raposo, and H. E. Stanley, "Lévy flights in random searches," *Journal of Physica A: Statistical Mechanics and its Applications*, vol. 282, no. 1, pp. 1–12, (2000).

[15] G. M. Viswanathan, S. V. Buldyrev, S. Havlin, M. G. D. Luz, E. P. Raposo, and H. E. Stanley, "Optimizing the success of random searches," *Nature*, vol. 401, no. 6756, pp. 911–914, (1999).

[16] S. Bergbreiter and K. S. J. Pister, "Design of an autonomous jumping microrobot," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 447–453, (2007).

[17] Statistics Bureau, Ministry of Internal Affairs and Communications, Japan, "Japan statistical yearbook 2017," http://www.stat.go.jp/data/nenkan/index1.htm, (January 25, 2017).

(Received October 31, 2016)
(Revised January 26, 2017)

Yuki Koizumi received his Master of information science and technology and Ph.D. in information science and technology degrees from Osaka University, Japan, in 2006 and 2009, respectively. He is currently an assistant professor at Graduate School of Information Science and Technology, Osaka University, Japan. His research interests include information centric networking and mobile networking. He is a member of ACM, IEEE, and IEICE.

Minoru Harada received his Bachelor and Master of information science and technology degrees from Osaka University in 2012 and 2014, respectively. He currently works at Nissay Information Technology Co., Ltd. His research interests include mobile ad-hoc networking, wireless networks, and self-organized networking.

Toru Hasegawa is a professor of Graduate school of Information and Science, Osaka University. He received the B.E., the M.E. and Dr. Informatics degrees in information engineering from Kyoto University, Japan, in 1982, 1984 and 2000, respectively. After receiving the master degree, he worked as a research engineer at KDDI R&D labs. (former KDD R&D labs.) for 29 years and moved to Osaka University. His current interests are future Internet, Information Centric Networking, mobile computing and so on. He has published over 100 papers in peer-reviewed journals and international conference proceedings including MobiCom, ICNP, IEEE/ACM Transactions on Networking, Computer Communications. He has served on the program or organization committees of several networking conferences such as ICNP, P2P, ICN, CloudNet, ICC, Globecom etc, and as TPC co-chair of Testcom/Fates 2008, ICNP 2010, P2P 2011 and Global Internet Symposium 2014. He received the Meritorious Award on Radio of ARIB in 2003, the best tutorial paper award in 2014 from IEICE and the best paper award in 2015 from IEICE. He is a fellow of IPSJ and IEICE.

Design and Implementation of a Multimedia Control and Processing Framework for IoT Application Development

Daijiro Komaki*, Shunsuke Yamaguchi*, Masako Shinohara*, Kenichi Horio*, Masahiko Murakami*, and Kazuki Matsui*

* Fujitsu Laboratories Ltd., Japan
{komaki.daijiro, yamaguchi.shun, m-shinohara, horio, mul, kmatsui}@jp.fujitsu.com

Abstract - When creating Internet-of-Things (IoT) applications, it is difficult to deal with multimedia data captured from cameras and microphones installed at field sites since it requires a wide variety of knowledge of topics such as codecs, protocols and image processing. To solve this problem, therefore, we propose a framework that makes it easy to deal with multimedia stream data in IoT application development. Our framework has three main features as follows: (1) Virtualization of multimedia input/output devices; (2) Distributed execution of multimedia processing pipeline between gateways and a cloud; and (3) Simple service description using a graphical flow editor. In this paper, we present some prototype applications we created and discuss the effectiveness of our framework from the perspective of complexity, productivity and ease of trial and error.

Keywords: Internet-of-Things, Multimedia, Framework, Web API

1 INTRODUCTION

We have entered the era of the *Internet-of-Things* (IoT), where not only computers but also physical objects (i.e., *things*) such as vehicles and home appliances are connected to the internet and interact with each other, or with systems, services and people. When creating such IoT applications, it is important to make devices such as sensors and actuators already installed at field sites (e.g., classrooms, concert halls, building entrances) available for various applications, rather than to install devices at a field site for a specific purpose [2, 5].

In addition, not only sensory data (e.g., temperature and acceleration) but also multimedia data (i.e., audio and video) captured from devices such as cameras and microphones installed at field sites are important for IoT application development, since we can offer many beneficial applications that utilize multimedia data as sensory data by using computer vision technologies (e.g., *detecting a suspicious person at a building entrance*), or that utilize sensory data as an input to process multimedia data (e.g., *adding effects to a live video stream of a concert event according to the mood of audiences there*).

On the other hand, there are some multimedia frameworks such as Kurento[1] [4] and Skylink[2] that make it easy to create applications utilizing multimedia data. These frameworks provide multimedia server programs and client libraries, and developers can easily create their applications without worrying about the differences in codecs and formats of audio/video contents by using the provided libraries. For example, developers can easily create applications equipped with multimedia features (e.g., VoIP, augmented reality) simply by connecting multimedia processing blocks as a pipeline. However, even when using these frameworks, problems remain when considering the characteristics of IoT application development, as follows:

- Since there may be many different types of devices at field sites, developers need to know the detailed specifications (e.g., an interface to start/stop capturing media data, to establish a media session between a device and a media server) in advance, and need to create applications according to the specifications.
- If all multimedia stream data generated at field sites are continuously transferred to a media server, they consume large amounts of network bandwidth.

To solve these problems, we designed and implemented a framework to simplify the process of multimedia IoT application development. Our framework has three main features as follows:

Virtualization of Multimedia Input/Output Devices

Our framework provides a mechanism to virtualize multimedia input/output devices (e.g., cameras and speakers) to obscure the differences in heterogeneous device specifications. This mechanism means that developers no longer need to consider the details of devices already installed at field sites, and applications once created can be adapted for another field site.

Distributed Execution of Multimedia Processing Pipeline between Gateways and a Cloud

Our framework provides a mechanism to distributedly process single multimedia stream data by coordinating multiple multimedia servers running independently on gateways (installed at field sites) and on a cloud, respectively. This mechanism enables developers to easily create multimedia IoT applications that can save network bandwidth usage and can serve immediate detection and response to field sites.

Simple Service Description using a Graphical Flow Editor

Our framework provides a web-based graphical flow editor tool to simply define a distributed multimedia processing

[1] Kurento: https://www.kurento.org/
[2] Skylink: http://skylink.io/

pipeline by connecting input/output device blocks and filter blocks. This tool enables developers to easily use trial and error by replacing and relocating each block.

In this paper, we present some of the prototype applications we created and discuss the effectiveness of our framework from the perspective of complexity, productivity, and ease of using trial and error.

2 RELATED WORK

2.1 Multimedia Frameworks

There are several frameworks that make it easy to create applications that utilize multimedia stream data. GStreamer [3] is an open-source framework for creating multimedia applications that handle audio, video and any kind of data flow in a modular way. The basic idea of GStreamer is to link together various plug-in elements (e.g., sinks/sources, encoders/decoders, filters) on provided pipeline architecture to obtain a stream that meets the desired requirements. This seems to be effective for developers who are not familiar with multimedia processing or multimedia networking. However, even when using GStreamer, developers are required to know which type of devices, which protocols, and which codecs to use in advance, in order to define a pipeline.

Kurento Media Server is an open-source multimedia server based on GStreamer that supports WebRTC. Developers can easily create web-based multimedia streaming applications (e.g., VoIP, video conference, augmented reality) using provided APIs. Since Kurento Media Server provides the mechanism to absorb the differences in media codecs and formats, even developers unfamiliar with multimedia processing (e.g., web application developer) can create multimedia streaming applications by linking media processing modules (e.g., image processing, event detection) via the provided web APIs. In addition, this framework provides a way to implement a new media processing module using OpenCV [4](Open-Source Computer Vison). Computer vision experts can create their new modules independently from the application development process.

Although it becomes easy to deal with multimedia stream data by using such framework technologies, these frameworks are not necessarily suitable for creating IoT applications (that make use of devices already installed at field sites) since developers need to create their applications according to device type, protocols, codecs and so on.

2.2 IoT Application Development Platforms

On the other hand, there are several cloud-based platforms for creating and deploying IoT applications that utilize multiple sensors and actuators installed at field sites. Kii Cloud[5], a *Backend-as-a-Service* for IoT application development, provides the functionalities to virtualize devices on the cloud. IoT application developers can create their IoT applications by combining multiple virtualized device functionalities by using provided APIs, so they need not be concerned about the differences in detailed specifications such as communication protocols.

IBM Bluemix[6] also provides a way to create IoT applications using virtualized device functionalities on the cloud. Bluemix provides a graphical flow editor (called Node-RED[7]) to create interactive, near real-time IoT applications by simply connecting things and services. Blackstock [3] focused on the fact that many IoT scenarios require the coordination of computing resources across networks: on servers, gateways, and devices, and extended Node-RED in order to create distributed IoT applications that can be portioned between servers and gateways. MyThings [8] (provided by Yahoo! Japan) enables users to create IoT applications that link various devices to various web services by simple IF-THEN rules.

Owing to such platforms, it becomes easy to create IoT applications that connect multiple devices and services to each other. However, if developers attempt to create an IoT application that deals with multimedia streams generated from field sites, they have to use multimedia frameworks such as those mentioned above.

There have been a few efforts to simplify the creation of IoT applications that can handle both sensory data and multimedia data in combination. ThingStore [1] provides the mechanisms to virtualize any type of device as a thing that generates Boolean data (i.e., Boolean value represents whether a certain event occurs or not). Owing to this abstraction, application developers can deal with media input devices as sensor devices and can simply create IoT applications that coordinate both sensory and multimedia stream data. However, since ThingStore abstracts multimedia stream data as Boolean data, it is not suitable for dealing with end-to-end multimedia stream data transferred from a device to another device.

2.3 Kurento Media Server

In this section, we focus on Kurento Media Server, which is the basis of our framework implementation. Kurento Media Server is an open-source software media server that makes it simple to create web applications equipped with multimedia features (e.g., VoIP, video conference, augmented reality). Kurento Media Server provides endpoint modules (i.e., elements to input or output the multimedia stream) and filter modules (i.e., elements affecting the media stream or detecting events from the media stream) shown in Table 1. Application developers are simply required to take the modules needed for an application and to connect them, without worrying about differences in codecs and formats of audio/video data. Figure 1 shows an application example: the video stream captured by the web browser is sent to the media server, then *FaceOverlayFilter* detects faces from frames of the video stream and puts a specified image on top of them, and finally the face-overlaid video stream is sent

[3] GStreamer: https://gstreamer.freedesktop.org/
[4] OpenCV: http://opencv.org/
[5] IoT Cloud Platform Kii: https://en.kii.com/
[6] IBM Bluemix: https://www.ibm.com/developerworks/cloud/bluemix/
[7] Node-RED: http://nodered.org/
[8] myThings: http://mythings.yahoo.co.jp/

back to the web browser, while recording it on the media server.

Kurento Media Server provides Java and JavaScript client libraries. Developers can use the functionalities Kurento Media Server offers on their applications. Figure 2 shows the required procedures to construct the pipeline shown in Fig. 1. Firstly, a web browser-side script creates a Session Description Protocol[9] (SDP) offer and sends it to the web application server. Then, a server-side script creates the pipeline by using the provided client library, makes *WebRtcEndpoint* process the SDP offer in order to get an SDP answer, and sends the SDP answer back to the client. Finally, the client processes the received SDP answer to establish a WebRTC session with the *WebRTCEndpoint* on the media server. When the client starts sending the video stream, the processed video stream data is sent back to the client. Our framework also uses the provided client library to control Kurento Media Server in the same manner as described.

The filters and endpoints that Kurento Media Server provides have their own methods for clients to change inner parameters. Moreover, events raised by filters are subscribable by client-side scripts. For example, in Fig. 3, an application can change the image path to overlay and can subscribe an event that a barcode is detected in a frame of video stream. Our framework makes use of such features of Kurento Media Server and implemented some additional functionalities from the perspective of the IoT scenario.

3 DESIGN OF THE FRAMEWORK FOR IOT APPLICATION DEVELOPMENT

3.1 Target

We aim to make it easier to deal with not only sensory data but also multimedia stream data among devices in such cloud-based IoT application development platforms. Our framework focuses on IoT applications where sensory stream data and multimedia stream data affect each other interactively. Typical scenario cases we assume as multimedia IoT applications are as follows.

[Case 1] *Capture live video stream from a certain camera at a field site and transfer it to a screen installed at the same place.*

[Case 2] *Obtain text segment data from live audio stream captured from a certain microphone by speech recognition, overlay the text on live stream video captured from a certain camera, and project the text-overlaid video stream on a nearby screen.*

[Case 3] *Count the number of people from live video stream captured by a certain camera, and detect an event according to the change of that number.*

[9] RFC 4566 - SDP: Session Description Protocol: http://tools.ietf.org/html/rfc4566.html

Table 1: Modules provided by Kurento Media Serer

Endpoints (Inputs/Outputs)	
WebRtcEndpoint	Send and receive WebRTC media flow
RtpEndpoint	Send and receive RTP media flow
PlayerEndpoint	Read media from a file or URL
RecorderEndpoint	Store media flow to a file or URL
Filters (Processing/Detecting…)	
FaceOverlayFilter	Recognize face areas and overlay picture on that area.
ZBarFilter	Detect barcode and QR code
GStreamerFilter	Use filters of GStreamer

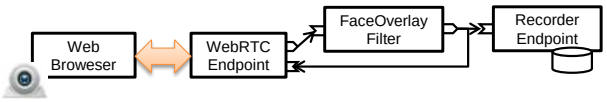


Figure 1: Example of connecting endpoints and filters

[Case 4] *Record live stream video captured from a certain camera installed at a field site only when the temperature there is higher than a threshold value.*

3.2 Functional Requirements

In order to make it easy to create such above multimedia IoT applications, we extract the requirements of functionalities that the framework should provide as follows:

(1) Virtualization of Multimedia Input/Output Devices

Considering the above scenarios, it is desirable to deploy an IoT application once created to many various field sites rather than to create an IoT application for a specific field site. However, there may be different types of devices (e.g., IP camera that supports RTSP, USB camera) at field sites and they may communicate using different protocols (e.g., WebRTC, RTP, HTTP). Such heterogeneity does not become a problem when creating conventional web applications since developers already know which type of device to use (i.e., devices are virtualized on the HTML5 layer on the browser side). However, from the perspective of IoT scenarios, it is assumed that many IoT applications utilize the same devices already installed at field sites together. Therefore, the framework should obscure such device heterogeneity so that developers do not need to consider it.

(2) Distributed Execution of Multimedia Processing Pipeline between Gateways and the Cloud

Since multimedia stream data is far larger than sensory data, it consumes a large amount of network bandwidth if transferring all multimedia stream data generated at field sites to the cloud. Considering the case where the results of event detection from video stream data captured from a field site are fed back to the same field site (Case 2), or the case where an IoT application needs only metadata extracted from multimedia streams (Case 3), Processing multimedia stream data on a computational resource near the field site is

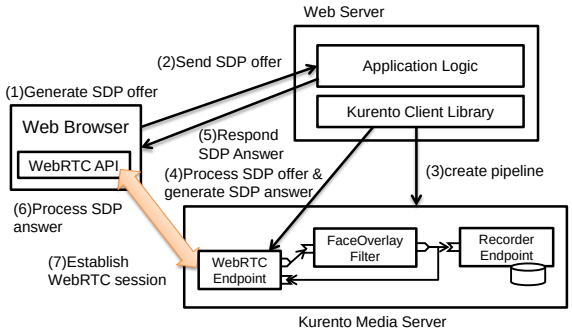


Figure 2: Procedure to establish media session

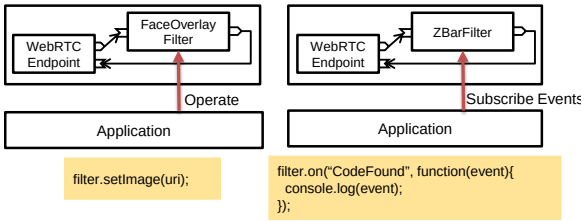


Figure 3: Interaction between application and filters

effective in saving network bandwidth usage and responding to the field site quickly.

To realize this, it is effective to process a single multimedia stream distributedly between a gateway (installed at the field site) and a cloud. However, in order to do that, it requires laborious procedures such as opening ports for sending/receiving multimedia stream on multiple media servers and establishing a media session between them. Therefore, the framework should enable the processing of single multimedia stream distributedly without considering media session establishment.

(3) Cooperation with External System

We assume not only the case where the framework control and process end-to-end multimedia stream are sent from one device to another (Case 1 and 2) but also the case where the framework detects an event using time series metadata extracted from a multimedia stream (Case 3) and the case where the framework controls the media stream according to the changes in sensory data (e.g., temperature) (Case 4). To do that, the framework should provide a way to easily cooperate with an existing IoT platform that provides the functionalities of time series data analysis or complex event processing.

(4) Simple Description of Media Processing Pipeline

By using media framework technologies such as Kurento Media Server, developers can easily create multimedia web applications by connecting multiple endpoints and filters as a pipeline, and can easily use trial and error by replacing or reconnecting each block. The framework should inherit this feature to simply implement a media processing pipeline, while satisfying the above three requirements ((1)-(3)).

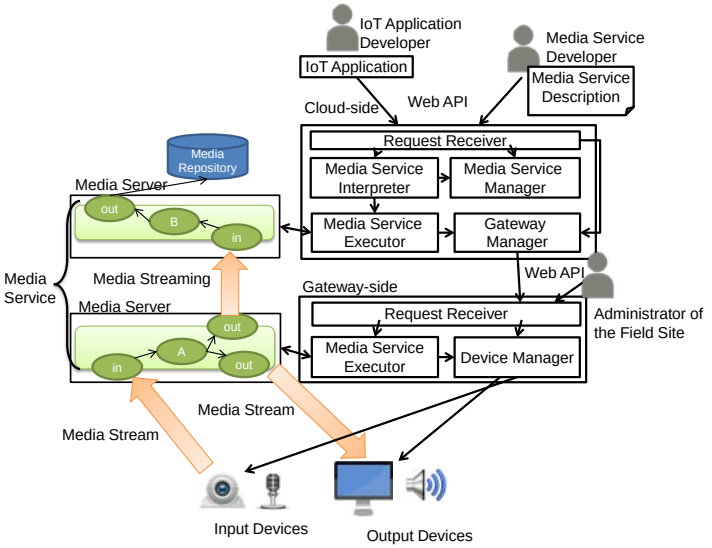


Figure 4: Architecture of our framework

4 IMPLEMENTATION OF THE FRAMEWORK

4.1 Architecture

In order to meet the above requirements, we designed the architecture of our framework (Fig. 4). Here, we define the term media service as a set of an entity to process multimedia stream data running on media servers, and an IoT application as an entity to use media service(s) by using the APIs that our framework provides. Our framework uses multiple media servers running independently on the gateway(s) and the cloud, respectively, and deploys corresponding modules to cooperate with multiple media servers. The framework forms star topology, where the cloud-side module aggregates all gateway-side modules. Noted that we adopted Kurento Media Server as the media server, but other media servers are adaptable to realize such architecture. In the following, we describe the behavior of each component.

[Request Receiver]

This component receives the requests from the clients (e.g., registration of devices, gateway, media services, and operation of media services) via web APIs (shown in Table 2). The person who installs the devices uses the web APIs on the gateway side, while media service developer and IoT application developer use those on the cloud side.

[Media Service Manager]

This component manages the media service descriptions written in JavaScript Object Notation (JSON) format. Since media services to be executed on the gateway side are deployed at the time of execution, media service descriptions are centrally managed on the cloud side.

[Media Service Interpreter]

Table 2: Web APIs

URI	Method		parameters
/service	GET	Get a list of registered media service descriptions	
			id
			service
/service/:id	GET	Get the media service description specified by id	id
			id
			params
/pipeline	GET	Get a list of executed media service instances	
/pipeline	POST	Create media service instance	service:
/pipeline/:id	GET	Get the media service instance specified by ID	id
			id
			method
/gw	GET	Get a list of registered gateways	
			key
			uri
/device	GET	Get a list of registered devices	
			key
			uri

This component converts media service descriptions into executable ones for the **Media Service Executor**; this component divides media service description into cloud-side and gateway-side media services.

[Media Service Executor]

Based on converted media service descriptions (described above), this component initializes and controls media processing modules on the corresponding media server. In addition, this component establishes media sessions between devices and gateways and between gateways and the cloud. The cloud-side **Media Service Executor** cooperates with the **Gateway Manager** in order to deploy a media service to the specified gateway and establish the session between media services, while the gateway-side one cooperates with the **Device Manager** in order to establish a media session between the device and the endpoint on the gateway-side media server.

[Gateway Manager]

This component manages the relationships between ID of each gateway (i.e., keyword to specify a field site) and their URLs and provides an interface to control gateways. When receiving the requests from **Media Service Executor**, this component forwards it to the specified gateway-side module.

[Device Manager]

This component manages the relationships between ID of each devices and connection information (e.g., URL, socket ID in the case of using HTTP, WebSocket, respectively) and provides an interface to negotiate SDP and to start/stop and sending/receiving multimedia stream data.

In the following, we describe the procedure to create a multimedia IoT application using our framework functionalities. The main players in this scenario are *Field Site Administrator*, *Media Service Developer*, and *IoT Application Developer*. These players may be either the different persons respectively or the same person.

(1) Registration of the Gateway

The *Field Site Administrator* edits the configuration file in the gateway module to define the field site ID. When starting up the gateway-side module, a request for registering this gateway is automatically sent to the cloud.

(2) Registration of the Devices

The *Field Site Administrator* registers devices to the gateway-side module via using gateway-side APIs by specifying the device ID and device type. Device IDs needs to be identifiable only in the same field site since they are managed by each gateway.

(3) Registration of the Media Service

The *Media Service Developer* writes the media service description (such as shown in Fig. 5) and registers it to the cloud-side module using cloud-side APIs.

(4) Execution of the Media Service

The *IoT Application Developer* connects his/her application to the specified media service using server-side APIs to operate media services.

4.2 Details of Functionalities

In this section, we describe the detailed behaviors of the functionalities of each component above.

Interpretation/Execution of Media Services

The *Media Service Developers* describe their services in JSON format (shown in Fig. 5). This example shows a media service where the video stream captured from a specified camera at a specified field site is processed to put a specified image on the face area on the gateway side and sent to cloud side to be recorded. Here, **type** is used to specify the type of filter/endpoint, **place** is used to specify the execution place (i.e., gateway or cloud), **front_id** is used to specify the field

```
[
 {
    id: 0,
  //type of  device
  type: “DeviceEndpoint”,
  //specify the device id to use
  key:“camera001”,
  //specify a place to execute media service
  place: “gw”,
  //specify the field site id ($0 is variable)
  front_id: “$0”,
  out:[0]
 },
 {
    id: 1,
  type: "FaceOverlayFilter",
  //filter’s own property (specify image url to overlay)
  img: “http://XXXX/hat.jpg”,
  place: "gw",
  in:[0],
  out:[1]
 },
 {
    id: 2,
  type: "RecorderEndpoint",
  place: "cloud",
  in:[1]
 }
]
```

Figure 5: Example of media service description

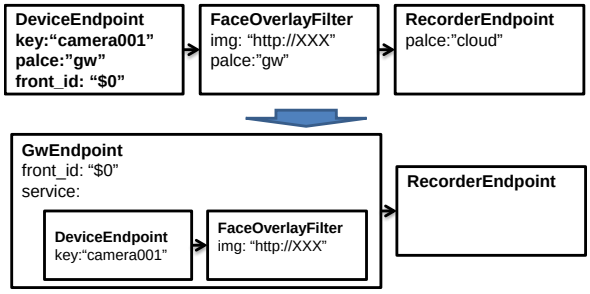


Figure 6: Transformation of media service description

site where the media service is applied, and **in/out** is used to specify the relationship between elements.

The **Media Service Interpreter** divides the received media service description and creates a *GwEndpoint* that includes a partial media service description that should be executed on the gateway side (shown in Fig. 6). Based on this converted media service description, the **Media Service Executor** initializes filters and endpoints on the media server and connects them.

In addition, the media service description can accept variable definition. For example, in Fig. 5, **front_id** (i.e., the ID that specifies where the device is installed) is defined as a variable (**“$0”**) so that it can be set when this media service is executed.

Virtualization of Media Devices

As an endpoint of multimedia stream data via a network, Kurento Media Server has three different types of endpoints: *RTSPEndpoint*, *WebRTCEndpoint* and *RTPEndpoint*. When using a camera that supports RTSP, it is required to simply specify the resource URL to establish a media session between the device and a media server, while it is necessary to

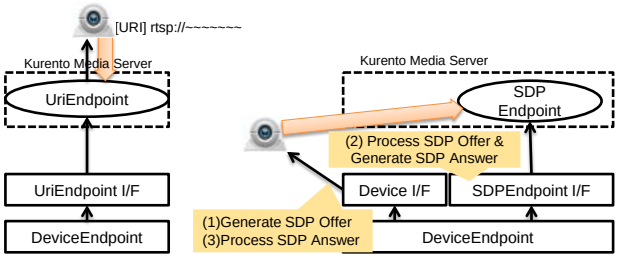


Figure 7: Obscuring the initialization procedure that varies according to device type

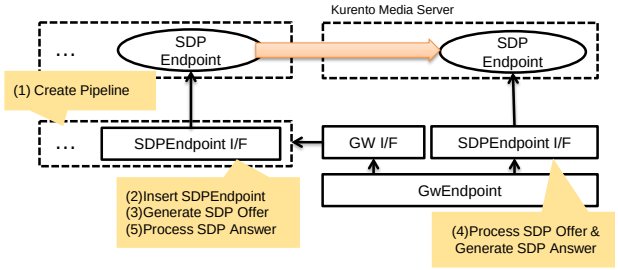


Figure 8: Session establishment between gateway and cloud

manually negotiate SDP when using a camera that supports WebRTC or RTP. Moreover, since each device may provide its own interface to operate (start, stop), developers need to take care of how to establish a session and how to operate devices that vary according to device type.

Therefore, the framework provides a set of classes, each of which implements required procedures to establish a session according to the corresponding device type (Fig. 7). *Field Site Administrators* are required to specify the device type when registering a new device. Owing to this, *Media Service Developers* do not need to be concerned about such differences in devices. A media session is automatically established when executing the media service.

Cooperation between Gateways and Cloud

To execute a media service distributedly on gateways and a cloud, it is necessary to establish a media session between divided partial media services. Therefore, the framework automatically inserts an *SDPEndpoint* (i.e., either *RTPEndpoint* or *WebRTCEndpoint*) at the end of the gateway-side *media service* description. The framework also creates an *SDPEndpoint* on the cloud-side media server and establishes a media session between gateway-side and cloud-side *SDPEndpoints* (as shown in Fig. 8). Thereafter, when the cloud-side module receives the request to operate a *media service*, it propagates this request to the corresponding gateway-side module.

Management of Events

The framework enables developers to describe event subscription between two filters in a media service description. As shown in Fig. 9, three attributes are required to define an event subscription: **target** (to specify which filter or endpoint publishes the event), **event** (to specify which type of

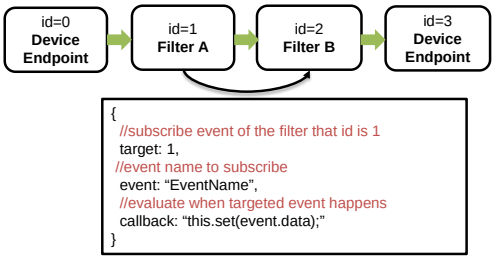


Figure 9: Event description between elements

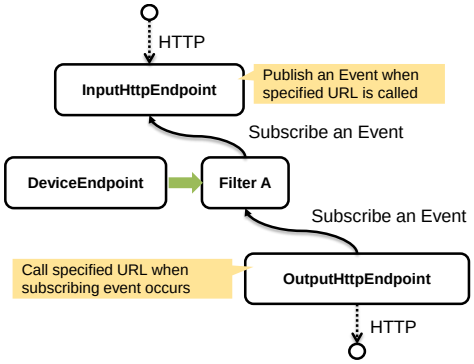


Figure 10: Input/output endpoint to external systems

event to subscribe), and **callback** (to describe the callback function that is evaluated when the specified event occurs). In the callback function, variable **this** is bound as the subscriber element itself.

Additionally, as shown in Fig. 10, the framework provides two endpoints in order to cooperate with external systems: *InputHttpEndpoint* publishes the event when a specified URL is called by an external system and *OutputHttpEndpoint* subscribes inner events and calls the external URL (specified in advance) when a specified event occurs. By using these endpoints, developers can easily create a media service that can process media stream data according to environmental changes or can store time series of metadata extracted from media stream data into external databases.

Graphical Editor for Media Service Description

Developers are able to create media service by following JSON format as shown in Fig. 5 without coding complicated logic. Furthermore, we implemented a web-based graphical flow editor for easily creating media service descriptions (Fig. 11-13). In the following, we describe how to use this client.

Figure 11 shows an example of the screen for creating and editing the media service, which is implemented using a SVG-based JavaScript library, JointJS[10]. When a user selects an item from the left-side list, a new node appears on the center area. The user can make a link from an input port of a node to an output port of another node by dragging and dropping. When a selected item requires some properties (e.g., image URL path for *FaceOverlayFilter*), input forms corresponding to each property appear on the right side of the screen. Event subscription can be defined in this area.

Figure 12 shows the screen for executing specified media service. The user can select which field sites to apply the specified media service to. Figure 13 shows a list of executed media services and the user can operate (i.e., start, stop, pause, release) each media service.

5 PROTOTYPE APPLICATIONS

In order to verify the effectiveness of our framework, we created three prototype IoT applications. In this section, we explain these IoT applications and discuss the features of each application.

[Prototype 1] Supporting Lectures in the Classroom

In the lectures at universities, teachers often use the projector to present their documents on the screen display. In such lectures, a teacher may use a stick or laser pointer to specify the focus area of the screen display. However, students may not clearly see the specified area in a large classroom. Therefore, we implemented an application that supports such lectures. We implemented it by connecting *TrapezoidCorrectorFilter* (which transforms a trapezoid-shaped area to square), *FingerDetectorFilter* (which detects the coordinates of fingertips and raises an event), and **ScalerFilter** (which expands the area around a specified point) as shown in Fig. 14.

There are three devices registered to the framework in the classroom: a camera (which captures video stream data including screen display area for detecting fingertips), a screen capturer (which captures video stream data from the teacher's PC screen), and a display screen (which displays the video stream process by *ScalerFilter*). Using this combination, the area of the screen the teacher points is scaled so that students can look at the focused area clearly. Since this media service is executed on the gateway-side, immediate response (i.e., followability of finger motion) can be expected compared with executing on the cloud-side.

[Prototype 2] Monitoring Suspicious Person

Suspicious person monitoring services using networked cameras are now widely used in various areas. However, when operating such monitoring services on the cloud, it consumes a large amount of network bandwidth and storage. Therefore, we created an IoT application that transfers a video stream data to the cloud while detecting a moving object and records it on the cloud.

We realized this by connecting *MotionDetectorFilter* (which detects moving objects using background subtraction) and *SwitchFilter* (which can be switched to drop or pass-through received buffer) as shown in Fig. 15. Since video stream data is transferred to the cloud only when moving objects are detected on the gateway side, network bandwidth and cloud storage can be saved.

[Prototype 3] Preventing Workers from Heatstroke

In summer, outdoor manual laborers are exposed to a risk of heatstroke due to both high temperatures and high-

[10] JointJs: http://www.jointjs.com/

Figure 11: Media service description screen

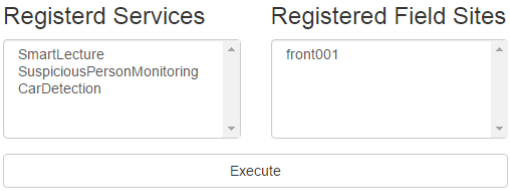


Figure 12: Media service execution screen

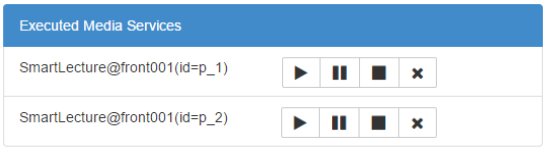


Figure 13: List of executed media services

humidity. To prevent this, there is a rule on restricting continuous work according to the heat index called WBGT[11]; however, it is difficult for the field overseer to know the WBGT of the corresponding field site and the health conditions of all workers at all time. Therefore, we implemented a monitoring application which records video stream data that captures a specified field site when WBGT is above a threshold and reports to the field overseer when a worker stops moving.

Here, we adopted an existing IoT platform that can detect events according to the changes in time series data. The WBGT value, calculated from temperature and humidity using sensors installed at the field, is continuously registered to the IoT platform. Whenever the WBGT value goes above a specified threshold, the IoT platform calls the web API defined by *InputHttpEndpoint*. This cooperation makes it possible to control media service (e.g., start recording video stream data, start detecting moving objects from video stream data) according to the changes in sensory data (e.g., WBGT). At the same time, our framework notifies the result of moving object detection to the IoT platform using *OutputHttpEndpoint*, and the IoT platform can send warnings to the overseer and workers according to the result. *InputHttpEndpoint* and *OutputHttpEndpoint* make it easy to create multimedia IoT applications that cooperate with existing IoT platforms.

[11] National Weather Service Weather Forecast Office: http://www.srh.noaa.gov/tsa/?n=wbgt

6 EVALUATION

We evaluated the effectiveness of our framework based on the above prototype applications from the perspectives as follows:

6.1 Complexity

To process single multimedia stream data cooperatively using multiple media servers, developers are required to establish a media session using RTP or WebRTC in addition to implementing originally required media processing. In addition, developers are required to take care of which endpoint to use and how to establish a media session, which varies with the device type installed at the field site. Our framework spares developers from such complexity, so IoT application developers can be dedicated to connecting devices at field sites with media processing modules.

6.2 Productivity

Table 3 shows the comparison of the number of program lines between cases using our framework and not. These numbers are counted without brackets. Although this comparison may not be fair since there is a difference between using or not using our framework (i.e., declarative description using JSON and procedural description using JavaScript), we confirmed that our framework works effectively since developers are not required to write logic to establish the media session both between the gateway and the cloud

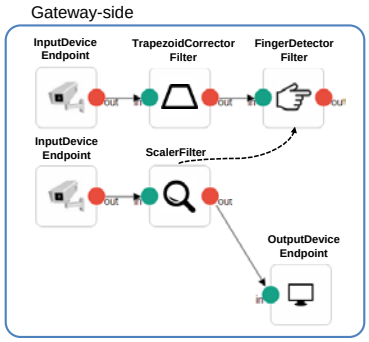


Figure 14: **[Prototype 1]** Media service description for lecture support

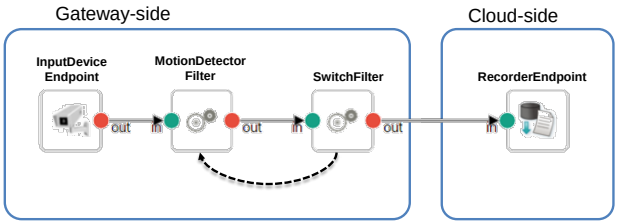


Figure 15: **[Prototype 2]** Media service description for suspicious person monitoring

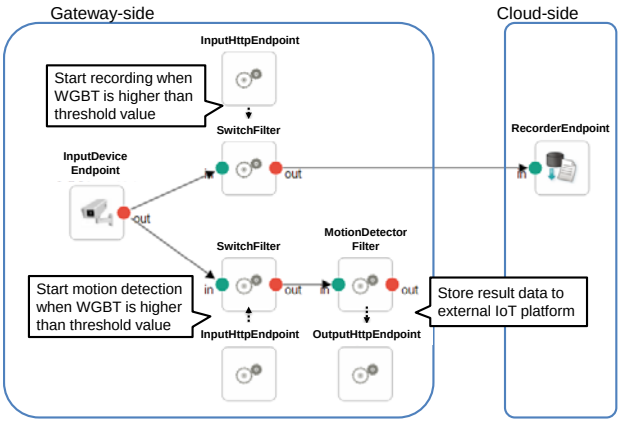


Figure 16: **[Prototype 3]** Media service description for heatstroke prevention

Table 3: Comparison of the number of program lines

	Used	Not used
Prototype 1	38	50
Prototype 2	26	42
Prototype 3	50	48

and between the media server and the device in the case of Prototype 1 and Prototype 2 (24% and 38% reduction, respectively).

On the other hand, considering the case of Prototype 3, i.e., the case where media processing pipeline topology changes according to an event, we defined a single static media service description that includes *SwitchFilter* in the case of using our framework, while we implemented this switching as IoT application-side logic in the case not using our framework. As a result, the number of lines not using our frameworks is fewer in spite of writing logic to establish a session with the gateway and the device. Therefore, the current media service description format is not suitable for describing a dynamically changing pipeline. We need to consider an effective way to describe event-driven media services as a future work.

6.3 Ease of Trial and Error

It is important to repeatedly use trial and error for creating IoT applications. However, in multimedia application, logic to handle multimedia input and output are tightly-coupled with multimedia processing logic itself. As a result, when application developers modify their application, they are required to rewrite programming logic itself. For example, in order to change the behavior of Prototype 1 to put a circle on the fingertips, developers are required only to replace ScalerFilter with another (i.e., a filter to put marker) and do not need to deploy the application again by using our framework.

6.4 Division of Labor

Considering the case of creating IoT applications such as **Prototype 1** using only Kurento Media Server, the application may not be adaptable to other classrooms, since the installed device type may differ from classroom to classroom. On the other hand, by using our framework, an IoT application once created can be adapted for any other classroom only if each device is registered in the same name owing to our framework's device virtualization mechanism. This makes it possible to create IoT applications independently from device installation.

In addition, not only cameras, microphones and screens, but also any software modules implemented to meet the specification of a virtualized device interface can be registered as a device. Developers can equally treat both cameras and screen capturer modules on our framework.

6.5 Cooperation with External Systems

Our framework can deal with not only the case where a media stream data captured from a device is processed, recorded and transferred to another device, but also the case where an external system affects multimedia stream data using *InputHttpEndpoint* and notifies the external system when an event occurs using *OutputHttpEndpoint*. In the case of Prototype 3, we adopted an existing IoT platform, but we are not limited to such IoT platforms. For example, cooperating with the existing complex event processing system enables more advanced event detection by using both sensory stream data stream and multimedia stream data.

In other words, our framework can be used as an extension to make it easy to deal with media stream data on an existing IoT platform rather than a substitute. Currently, our framework supports cooperation using only HTTP requests, but we plan to implement endpoint modules for protocols other than HTTP (e.g., MQTT, WebSocket) to make it easier to create IoT applications that deal with both multimedia and sensory data streams.

6.6 Performance

Our proposed framework can process single multimedia stream data by coordinating multiple multimedia servers running on gateways (installed at field sites) and the cloud, respectively. In typical IoT scenario, low-cost and not high-performance gateway devices are often used for each field site since it is necessary to distribute gateway devices to a large number of filed sites. In this section, we rather focus on gateway side where computational resources are restricted. We conducted a performance evaluation to confirm that our framework can work sufficiently when using a general gateway device used for IoT. Table 4 shows the specification of the gateway used for the following evaluation.

First, we discuss how many services our framework can handle when changing the quality of multimedia data and type of multimedia processing. Performance evaluations were conducted in the configuration shown in Fig. 17. A multimedia processing filter was deployed between WebRTC input and output endpoints on the gateway, while two Web browsers (Google Chrome) were used for capturing and showing video stream data using a USB camera on another machine. In this evaluation, two different quality of video streams were used; high quality (resolution: 640px x 480px, framerate: 30fps) and low quality (resolution: 320px x 240px, framerate: 10fps). Table 5 shows the result of CPU and memory usage rates when ScalerFilter, FingerDetectorFilter, and MotionDetectorFilter were used as a multimedia processing filter in the above configuration, respectively.

Compared with the case not using filters, CPU and memory usage rates increased when using filters, and these increased rates varied depending on the type of multimedia processing and video quality (11.9% - 84.30%). Our framework can handle 8-9 services simultaneously when using light-weight processing filters to low–quality video stream, while only one service when using heavy-weight processing filters to high-quality video stream.

Next, we discuss the overhead of our framework. As described in the section 4, our framework consists of two layers; multimedia processing layer and control layer (Fig. 18). Multimedia processing layer (written in C++) processes incoming multimedia stream data, while control layer (written in JavaScript (node.js)) receives requests from clients, manages registered devices, and coordinates multimedia servers and so on. We evaluated the overhead of control layer. The results were 0.2% of CPU usage rate and 4.6% of memory usage rate regardless of the type of multimedia processing filters and video quality. This result shows our extension have little effect on CPU usage at the time of execution.

Moreover, as shown in Fig. 18, our framework can handle event publishing and subscribing among multiple filters. In the example of Fig. 18, control layer subscribes an event of FingerDetectorFilter (a coordinate of a fingertip) and set the parameter (a coordinate to expand around that point) of ScalerFilter. Table 6 shows the result of CPU and memory usage rates of control layer using different framerate video streams (10fps and 30fps).

Table 4: Specification of the gateway

OS	Ubuntu 14.04 LTS 64 bit
CPU	Intel Atom E3826 @ 1.46 GHz x 2
Memory	1.8GB

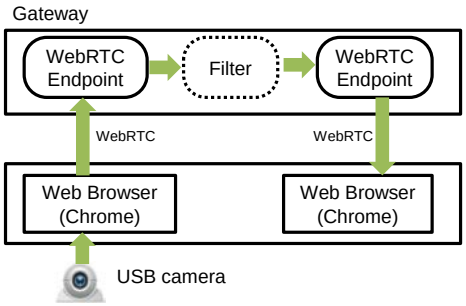


Figure 17: Configuration of evaluation environment

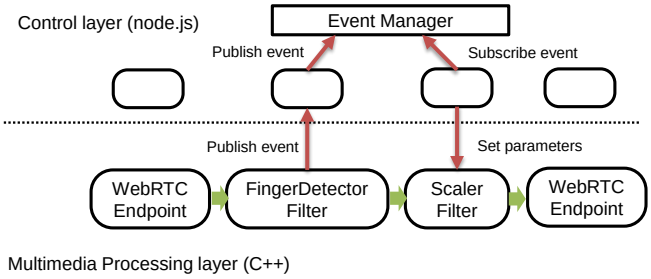


Figure 18: Proposed framework consists of two layers; multimedia processing (C++) and control (node.js) layer

The result shows that passing event data among filters had a little effect on CPU usage, and this rate was proportional to the framerate of the video stream, while memory usage rate was almost constant regardless of event subscription. From the result, we confirmed that the load of event publishing and subscribing was far smaller than that of multimedia processing.

From the above results, we confirmed that our framework worked sufficiently despite using a general gateway device for IoT when the quality of video stream was not so high or multimedia processing was not so heavy. Our framework can handle 8-9 services simultaneously when using light-weight processing filters to low–quality video stream. Besides, while our framework extends an existing multimedia server program to manage virtualized devices, to coordinates multiple multimedia servers and so on, processing overhead of this extension was sufficiently small.

On the other hand, however, our framework can handle at most one service simultaneously when using heavy-weight processing filters to high-quality video stream. If we need to use more high-quality video stream, this gateway device cannot process sufficiently. In order to provide a stable service, as a future work, we need to develop a mechanism to dynamically determine whether each multimedia processing filter should be run on the cloud or the gateway according to the performance of the gateway device and processing load of multimedia processing filters, and a mechanism to lively migrate a multimedia processing filter from the gateway to

Table 5: CPU and memory usage rates

	Low Quality (320 x 240, 10fps)		High Quality (640 x 480, 30fps)	
	CPU	Memory	CPU	Memory
None	7.4%	3.2 %	12.1%	3.3%
ScalerFilter	19.3%	5.9%	72.3%	7.1%
FingerDetector Filter	26.1%	6.4%	63.5%	7.5%
MotionDetector Filter	47.4%	6.8%	96.4%	9.0%

Table 6: CPU and memory usage rates of control layer

10fps		30fps	
CPU	Memory	CPU	Memory
1.7%	5.1%	3.0%	5.1%

the cloud if the processing load of the gateway becomes large.

7 SUMMARY

We designed and implemented a framework that makes it easy to deal with multimedia data such as audio and video generated from devices installed at field sites. The features of our framework are as follows:

- Virtualization of multimedia input/output devices
- Distributed execution of media service between gateways and a cloud
- Simple media service description using a graphical flow editor.

In this paper, we presented the three prototype applications we created and discussed the effectiveness of our framework from the perspective of complexity, productivity, and ease of trial and error.

As future work, we need to improve the media service description format and create endpoints other than HTTP. We plan to offer our framework to workshops and hackathons to verify the effectiveness of our framework from both qualitative and quantitative perspectives.

REFERENCES

[1] K. Akpinar, K. A. Hua, and K. Li., “ThingStore: A Platform of Internet-of-Things Application Development and Deployment,” ACM International Conference on Distributed Event-Based Systems (ACM DEBS 2015), pp.162-173, 2015.

[2] S. Alam , M. Chowdhury , and J. Noll., “SenaaS: An Event-Driven Sensor Virtualization Approach for Internet of Things Cloud,” IEEE International Conference on Networked Embedded Systems for Enterprise Applications (IEEE NESEA 2010), pp. 1-6, 2010.

[3] M. Blackstock, and R. Lea., “Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED),” International Workshop on Web of Things (WoT 2014), pp.34-39, 2014.

[4] L. Fernandez, M. P. Diaz, R. B. Mejias, and F. J. López, "Kurento: A Media Server Technology for Convergent WWW/Mobile Real-Time Multimedia Communications Supporting WebRTC," IEEE International Symposium and Workshops on World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2013), pp. 1-6, 2013.

[5] D. Munjin, and J. H. Morin, "Toward Internet of Things Application Markets," IEEE International Conference on Green Computing and Communication (IEEE GreenCom, 2012), pp. 156-162, 2012.

(Received October 8, 2016)
(Revised February 9 , 2017)

Daijiro Komaki received his B.E., M.E., and Ph.D. degrees from Osaka University, Japan in 2008, 2009, and 2012, respectively. He joined Fujitsu Laboratories Ltd. in 2012. His current research interests include multimedia processing framework and multimedia networking system.

Shunsuke Yamaguchi received his B.E. degree from The University of Electro-Communications, Japan in 2005, and Master of Information Science and Technology degree from The University of Tokyo, Japan in 2007. He joined Fujitsu Laboratories Ltd. in 2007. His current research interests include multimedia processing framework and multimedia networking system.

Masako Shinohara received the B.E., M.E., and Ph.D. degrees from Osaka University, Japan in 2004, 2006, and 2009. She joined Fujitsu Laboratories Ltd. in 2009. Her research interests include human-centric computing for smartly supporting user’s behavior, and multimedia networking system for real time communication.

Kenichi Horio received the B.E and M.E. degree from The University of Tokyo, Japan in 1999 and 2001, respectively. He currently works for Fujitsu Laboratories Ltd. His research interests include mobile communications and multimedia communications.

Masahiko Murakami received the B.E. and M.E. degrees in electrical engineering from Kyoto University, Japan, in 1990 and 1992. He joined Fujitsu Laboratories Ltd. in 1992. He is currently researching about human-centric connections for providing user-friendly timely services tailored for individuals, including multimedia networking system.

Kazuki Matsui received his B.E. and M.E. dgrees from Keio University, Japan in 1990, 1992, respectively. He joined Fujitsu Laboratories Ltd. in 1992. His current research interests include Virtual Reality Computing System.

Zero Interruption-Oriented Techniques for Mobile Video-on-Demand in Hybrid Broadcasting Environments

Tomoki Yoshihisa*, Tomoya Kawakami**, and Yusuke Gotoh***

* Cybermedia Center, Osaka University, Japan
** Graduate School of Information Science, Nara Institute of Science and Technology, Japan
*** Graduate School of Natural Science and Technology, Okayama University, Japan
yoshihisa@cmc.osaka-u.ac.jp, kawakami@is.naist.jp, gotoh@cs.okayama-u.ac.jp

Abstract - Due to the recent proliferation of mobile devices and video-on-demand delivery, mobile video-on-demand delivery, i.e., the users watch videos using mobile devices, gets great attention. In mobile video-on-demand delivery, the users often watch videos while moving outside such as riding cars or trains. Current mobile video-on-demand delivery faces the problem that interruptions of video playback occur in some situations. So, some interruption reduction techniques have been studied. However, these techniques are originally designed for non-mobile devices and it is difficult to solve the problem for mobile devices under various situations. Hence, in this paper, we propose some techniques aiming to zero interruption. To reduce the video interruptions effectively, our proposed techniques use hybrid broadcasting environments. We develop a mobile video-on-demand system with our proposed techniques and report our experiments of mobile video-on-demand delivery using the developed system[1].

Keywords: Internet Broadcasting, Continuous Media, Streaming Delivery, Mobile Devices

1 INTRODUCTION

Due to the recent development of Information and Communication Technologies (ICT), mobile devices such as smart phones or compact PCs become popular. Mobile devices are small, lightweight, and can connect to the Internet at most places. So, the users can get information from the Internet using mobile devices at most outside places even though they do not seat in front of non-mobile devices such as desktop PCs. Meanwhile, video-on-demand delivery such as Internet broadcasting services by YouTube or TV companies becomes popular due to the communication speed up of the Internet. In video-on-demand delivery, the users select their preferable videos from homepages and request the video deliveries to the delivery server. The delivery server sends the video data to the requested clients according to their requests. The users can watch the video from the playing position of the received data without waiting for the reception of all data by using streaming technique, in which the clients play the video data while receiving them. The proliferation of mobile devices and video-on-demand delivery leads mobile video-on-demand delivery, i.e., the users watch videos using mobile devices. In mobile video-on-demand delivery, the users often watch videos while moving outside such as riding cars or trains since they can select and watch videos at most places.

Current mobile video-on-demand delivery faces the following problems.

Problem 1: Interruptions occur when there are a large number of clients.

Interruptions of video playback occur when there are a large number of clients since the bandwidth between the server and each client decreases. For example, suppose the case when a user requests playing a popular video delivered in YouTube when he/she is stopping at a red traffic signal. In this case, the video playback sometimes does not start or interrupts even if it starts. This problem also occurs for non-mobile devices, but frequently occurs for mobile devices since there are a large number of mobile devices.

Problem 2: Interruptions occur when the condition of electric wave gets worse.

Mobile video-on-demand delivery is often used while moving and the condition of the electric wave for the Internet connection changes. A worse electric wave condition causes a lower bandwidth. So, the video playback interrupts when the bandwidth becomes lower than the bit rate. For example, a user requests playing a video while riding on a train at a station and starts watching it. When the train moves to a far distance from the station, the electric wave does not reach to his/her mobile device and the video playback interrupts.

Problem 3: Video playback stops when the remaining battery is low.

Mobile devices are often used outside and it is difficult to charge their batteries outside. So, the users may reduce the battery consumptions caused by video playbacks to lengthen the running time of the devices. The battery consumptions for playing high bit rate videos are high since the processing data size per time increases. Hence, in mobile video-on-demand delivery, some users prefer playing low bit rate videos to high bit rate videos so as to lengthen the time to play videos.

[1] This research was supported in part by Grant-in-Aids for Scientific Research (B) numbered 15H02702, and Grant-in-Aids for Challenging Exploratory Research numbered 26540045.

Some interruption reduction techniques have been studied. But, these techniques are originally designed for non-mobile devices and it is difficult to solve these problems. A simple approach to solve the problem 1 is exploiting streaming delivery techniques such as CDN (Contents Delivery Network) or P2P streaming. However, it is difficult to solve the problem by only these techniques when there are too many mobile devices, e.g., many mobile devices in Japan play the videos of disaster situations. Regarding the problem 2, mobile devices can continue to play videos even when they lack Internet connections by sufficiently buffering the video data. However, mobile devices cannot sometimes get enough time to buffer the data, e.g., the train moves fast or the power of the electric wave is weak. Regarding the problem 3, battery consumptions can be reduced by stopping other applications running on mobile devices. However, in this simple approach, the convenience of mobile devices degrades.

Hence, in this paper, we propose three techniques aiming to zero interruption. To reduce the interruptions of playing videos effectively, our proposed techniques use hybrid broadcasting environments. In hybrid broadcasting environments, the clients can receive data both from the broadcasting system and the communication system. The broadcasting system can deliver data to all clients concurrently. So, by broadcasting the data that are requested by many clients, the interruptions of playing videos are reduced effectively. Also, we develop a mobile video-on-demand system using our proposed techniques and report the experiments of mobile video-on-demand delivery using our developed system.

2 RELATED WORK

2.1 Types of Video-on-Demand Delivery

Video delivery through the Internet is called video-on-demand delivery. In video-on-demand delivery, the users select their preferable videos from the lists shown on homepages, etc. and request playing the video. The delivery server can deliver different videos to some clients (mobile devices). But, the interruptions of videos occur when there are a large number of the clients and the bandwidth between the delivery server and each client becomes smaller than the video bit rate. Hence, for communication systems such as the Internet, P2P streaming delivery techniques, in which clients also deliver videos to other clients, have been studied [1, 2].

Video delivery through broadcasting systems such as terrestrial broadcasting or one-segment broadcasting is called near video-on-demand delivery. Near video-on-demand delivery realizes short waiting time for starting playing videos by broadcasting data to all clients. In near video-on-demand delivery, some methods to reduce the interruption time by creating effective broadcast schedules of video data have been proposed [3, 4]. These methods divide the data into some segments and reduces the interruption time by broadcasting the segments repeatedly according to the created broadcast schedule. However, the interruption time of near video-on-demand delivery is longer than that of video-on-demand delivery when the number of the clients is small.

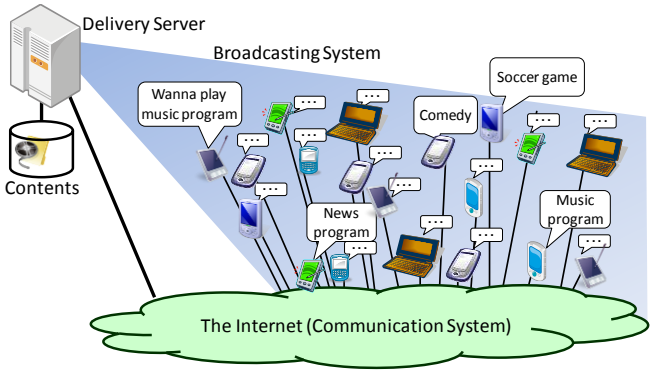


Figure 1: A hybrid broadcasting environment

Video delivery combining the video-on-demand and the near video-on-demand techniques is called unified video-on-demand delivery. Some methods for unified video-on-demand delivery have been proposed [5, 6]. In these methods, the delivery server fixes the broadcast schedule and sends the data, of that time to broadcast is long after, via the Internet. A method that generates the broadcast schedule dynamically is proposed in [7].

2.2 Video-on-Demand Services

Some video-on-demand systems have been developed for services.

One of the famous systems for delivering the videos that are submitted by the users is YouTube [8]. YouTube service started on 2004 at United States and widely used for delivering the users submitted videos. USTREAM also uses a video-on-demand system for delivering the users submitted videos [9]. USTREAM is often used for live broadcasting on the Internet. Niconico-douga is a Japanese video-on-demand service for users submitted videos [10].

Netflix provides a video-on-demand service for delivering the movies in Blu-rays or DVDs [11]. Netflix originally started a DVD rental service on 1997. Using the DVDs for the rental service, Netflix started the video-on-demand service. Hulu provides a similar video-on-demand service with charge [12]. There are many other video-on-demand services all over the world [13].

Techniques used in the above methods and services are originally designed for non-mobile devices and it is difficult to solve the problems for mobile devices under various situations.

2.3 Contribution of Our Work

Our proposed techniques for video delivery (the stream merge technique and the spare data delivery technique) are classified into unified video-on-demand delivery. Different from the delivery server of pure video-on-demand delivery or near video-on-demand delivery, the delivery server of unified video-on-demand delivery can deliver the video data both via the broadcasting system and the communication system. So, the video interruptions can be further reduced. Previously proposed methods for unified video-on-demand delivery focus on how to deliver the segmented video data to further reduce the interruptions. Our proposed techniques, moreover, focus on spare data and remaining batteries to reduce the

interruptions. These techniques to reduce the interruptions are differences between our research and [5]-[7]. Also, different from existing video-on-demand services, we adopt hybrid broadcasting environments in the paper.

The overhearing function in Wi-Fi communications can be regarded as a broadcasting system for hybrid broadcasting environments and our proposed stream merge technique can be applied for the overhearing function. However, different from the series of pseudo-broadcast methods using the function, our proposed techniques include the spare data delivery technique and the remaining battery based bit rate technique. We explain the detail of these techniques in the next section.

3 PROPOSED TECHNIQUES

In this section, we explain our proposed techniques. First, we explain our assumed hybrid broadcasting environments. After that, we explain our proposed techniques to solve the problems described in Section 1.

3.1 Hybrid Broadcasting Environments

Figure 1 shows our assumed hybrid broadcasting environment. The clients in the broadcasting area can receive data from the broadcasting system. Also, they can request their preferable data to the server and can receive them from the communication system. The broadcast station delivers data via some broadcast channels and is managed by the server. The server has streaming data and can broadcast the data to the clients using the broadcast station. Also, it can send the data to the clients using the communication system by unicasting.

3.2 Stream Merge

To solve the first problem, we propose a mobile video-on-demand system using stream merge technique. When there are many mobile devices that receive video data, the video-on-demand system can avoid decreasing the bandwidth between the server and the clients by sending the data to some clients concurrently and reducing the communication traffic. For example, suppose the case when Client 1 starts playing a news program in a train on 8:00 p.m. as shown in the left side of Fig. 2. The duration of the video is 5 minutes. One minutes after this, Client 2 starts playing the same video. In this case, the server delivers 2 video streams (a set of video data from the begging to the end) for 4 minutes from 8:01 to 8:05 in the simple conventional method. However, in our proposed technique, the server merges 2 streams from one minutes after the beginning to the end and delivers the merged stream to all clients as shown in the right side of Fig. 2. Client 2 can receive all data since it receives the data for one minutes from the beginning from the server directly. In this case, the server only delivers 2 streams for 1 minutes and can reduce the communication traffic. This is a simple example for the case of 2 clients. Actually, the server merges some streams for multiple clients. Broadcasting systems are suitable for delivering data to all clients. So, in our proposed technique, the server delivers the merged stream via the broadcasting

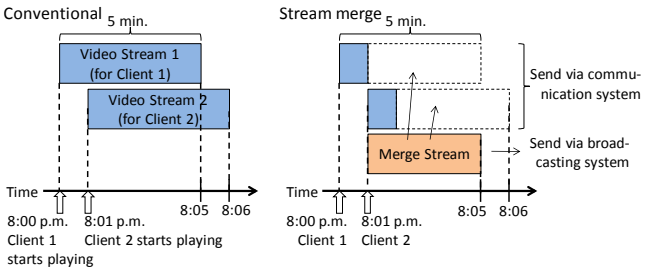


Figure 2: An example of stream merge

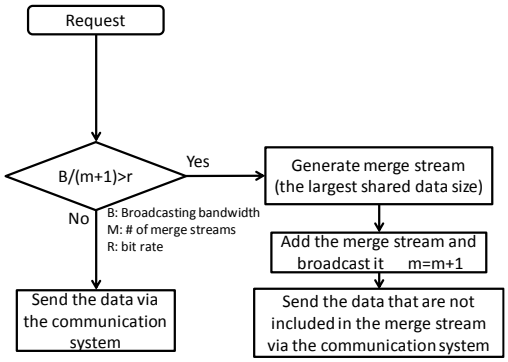


Figure 3: Flow chart for merging streams

system and other streams via the communication system. For this, we use hybrid broadcasting environments.

Figure 3 shows the flow chart for merging streams in our proposed technique. *B* denotes the broadcasting bandwidth and *m* denotes the number of the currently merged streams. The broadcasting bandwidth for each merge stream in case of adding another merge stream is $B/(m+1)$. When a client requests playing the data, the server compares this value and the bit rate *r*. If this is larger than *r*, the server generates a merge stream and sends it via the broadcasting system since interruptions do not occur by receiving the merge stream. Otherwise, the server sends the requested data to the client via the communication system since interruptions can occur by adding a merge stream.

3.3 Spare Data Delivery

To solve the second problem, we propose the spare data delivery technique. In the spare data delivery, for the case when the condition of the electric wave gets worse, the system delivers spare data beforehand. Different from the traditional technique that the clients buffer the requested video data, spare data are the data to keep the motivation for the users to watch the video. We use two types of spare data.

One is a data not related to the requested video such as commercial or weather forecasting. For example, the mobile device plays a commercial movie when the user riding train enters a tunnel and cannot catch the electric wave. The clients can download such spare data before requesting playing videos since the contents of the spare data is not related to video requests. The clients play spare data if the video playback interrupts.

The other is data related to the requested video but the data size is small compared with the video played when the clients have enough bandwidth, e.g., text or static image data. For example, the mobile device shows news by text when the user

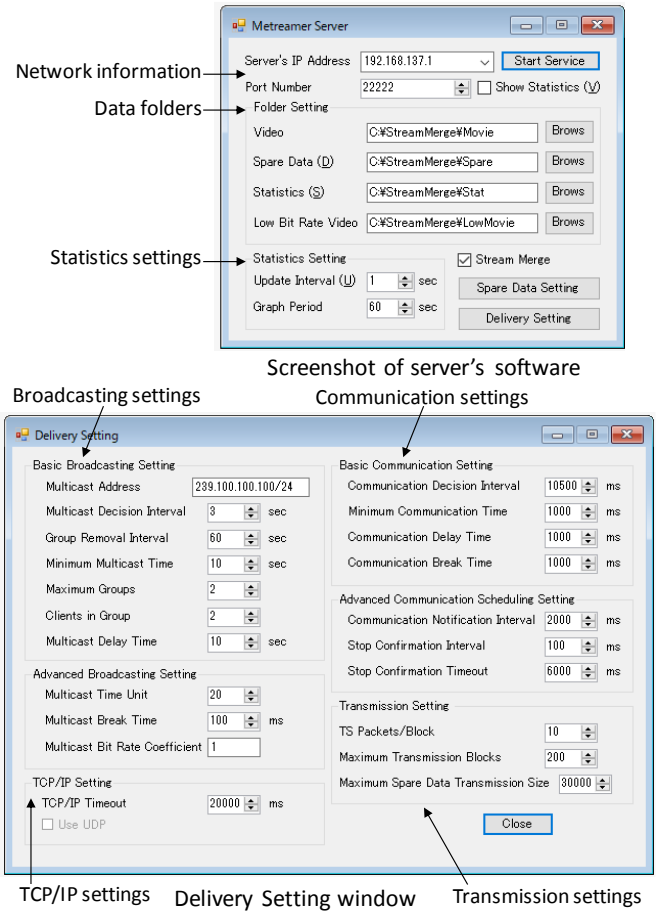


Figure 4: A screenshot of Metreamer server software

watches a news program and the video playback interrupts. The clients can download such spare data within a short time after requesting playing videos since the data size is very small.

When the users want to watch the whole requested video, the former type is suitable since the clients play the requested video although other videos can be played midstream. When the users want to grasp the content of the requested video, the latter type is suitable. In case of delivering the spare data not related to requests, broadcasting systems is suitable to deliver the data such as commercial or weather forecasting since these data are used as spare data for all clients. So, in our proposed technique, the server delivers such spare data via the broadcasting system on hybrid broadcasting environments. The detail algorithm such as how to get spare data and determine them depends on the implementation. We will explain the implementation for our developed system in the next section.

3.4 Remaining Battery based Bit Rate

To solve the last problem, we propose the remaining battery based bit rate. In the remaining battery based bit rate, the bit rate of the video is controlled on the client side. The clients play the video with normal bit rate when they have sufficient remaining battery to play the video. Otherwise, the clients reduce the battery consumption by decreasing the bit rate of the video. To change the bit rate on the client side, our

Figure 5: A mobile device running Metreamer

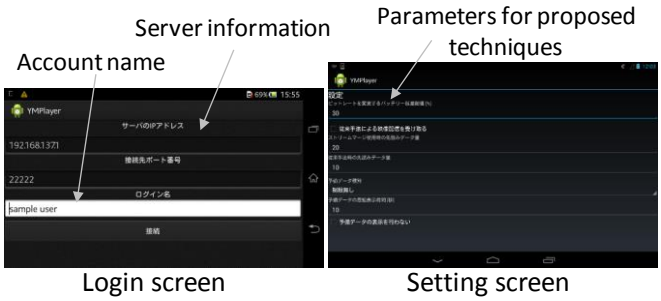


Figure 6: A screenshot of Metreamer client software

proposed technique uses multi-bit rate encoding. In multi-bit rate encoding, the clients can play the video with some bit rates determined beforehand. The resolution or the size decreases when playing the video at a lower bit rate. For example, by using multi-bit rate encoding, the clients can play the video at 128Kbps even when the server delivers the video at 1Mbps bit rate. In our proposed technique, the users set their preferable remaining battery and the corresponding bit rate. The server delivers the video at the highest bit rate for all the bit rate used by the clients so as to decrease the traffic.

4 DEVELOPED SYSTEM

We develop a mobile video-on-demand system using our proposed techniques. We call our developed system Metreamer, the abbreviation of mobile ever streamer.

4.1 Overview

Metreamer uses the broadcasting address of UDP protocol, that sends the same data to all the clients connected to the same network, as the broadcasting system and the unicasting of TCP protocol as the communication system. The delivery server has the video data and the spare data. If the server sends the spare data at the same time with the video data when the clients request playing the video, interruptions easily occur. So, in Metreamer, the server sends the spare data when there is a remaining bandwidth capacity such as after stopping the service or finishing all video data deliveries. Metreamer can use video, image, and text data as the spare data. The video encoding type is widely used MPEG2. Metreamer can

measure some statistic information such as the number of clients, and so on.

4.2 Software for Servers

Figure 4 shows a screenshot of the software for the servers of Metreamer. The software runs on Windows 7 and uppers. The server sometimes has some IP addresses and so the users can change the IP address on the software. The users can also change the directories for the video, spare, and statistic data. To show the statistic information, the users click the show button and set the interval to get the information. The users can set many other parameters for the video delivery as shown in the figure. By clicking the service start button, the software starts the delivery service and waits for the requests from the clients.

When the server software receives a request to play the video from a client, it start delivering the video data using our proposed stream merge technique explained in Subsection 3.2. Based on Fig. 3, the software sends the video data using the broadcasting address by UDP or the client's address by TCP. The software checks the connections from the clients cyclically and if a client disconnects for the reason of communication error or others, stops the video delivery. When the software finishes a video delivery to a client, it stores the statistic information. The software can show the information by graphs. When the software receives the request of receiving the spare data, it sends the data to the client.

4.3 Software for Clients

Figure 5 shows a mobile device playing a video using Metreamer. The mobile device is a special device for our p developed system. But the software for clients can run on other Android (4.1 or upper) mobile devices. The software for the clients of Metreamer first shows the login screen as shown in the left side of Fig. 6. By logging in, the client software can retrieve their settings. To enabling logging in, the users first register themselves to the server and after that they get IDs and passwords. The screen for setting some parameters is shown in the right side of Fig. 6. On this screen, the users can set the IP address of the server. In our developed system, the users directly input the server's IP address, but it can be set automatically by getting it from the server list wrote in some homepages in the Internet. On the setting screen, the users can set the data amount for cashing, the spare data types for the spare data delivery technique, the bit rates and the remaining battery to change the bit rate for remaining battery based bit rate.

After connecting to the server, the client software gets the video list and show the list. The users select their preferable video from the list and the software requests the video data by tapping the selected video. The software receives the data from the serer after the request, and the video playback starts when the software is ready for showing the video. If the software cannot receive the data for playing the subsequent video, it plays the spare data. In conventional systems, interruptions occur in such cases. The users can pause the video by tapping the pause button and stop the playback by tapping the stop button. When the users stop the playback or finish playing the video, the software again shows the video list.

4.4 Implementation of Proposed Techniques

4.4.1 Stream Merge Implementation

The server gets the broadcasting bandwidth B from its specification beforehand. If the actual broadcasting bandwidth largely differs from the specification, the server modifies the value. The server knows the number of merge stream N since it sends the merge streams. By checking the duration and the data size of each video, the server can get the bit rate r of the video data. Using these values, the server calculates $B/(m+1)$ when it receives the requests to play video from the clients. If the value is smaller than r, the server generates the video stream for the client and send it to the client. If the value is larger than r, the server generates a merge stream so that the duration of the merge stream becomes the longest. The merge streams are not merged to other merge stream again to make the merging algorithm simple in our developed system. Then, the server sends the merge stream to all clients via the broadcasting system and stops sending the streams that are merged to the merge stream. At the same time, the server sends the remaining data for the client, i.e., the data that the requested client cannot receive from the merge stream, to the client.

4.4.2 Spare Data Delivery Implementation

When a user runs the client software first, the client does not have spare data. So, the client requests the spare data to the server if it has no spare data. If the user changes the spare data type by setting screen, the client requests the spare data. The spare data are stored in the server and the clients receive some spare data from the server. Thus, the clients have some spare data before starting playing the video. If a client does not have the subsequent video and an interruption will occur, it selects a spare data randomly and shows it. If the number of the spare data is largely less than the number of the interruptions, the clients show the same spare data sometimes.

4.4.3 Remaining Battery based Bit Rate Implementation

The client software can get the remaining battery by asking it to the OS. When the remaining battery changes and the bit rate of the video changes according to the user's setting, the client changes the bit rate. If the client is playing a video, immediately changes the bit rate. Otherwise, the client uses the changed bit rate from the next video playback.

5 EXPERIMENTS

To investigate the effectiveness of our proposed techniques, we used our developed system in two practical situations since computer simulations do not reflect actual situations completely. We used our developed mobile video-on-demand system at Nakano central park and Okayama castle. Through these experiments, we got some questionnaire results. We report the experiments in this section.

Figure 7: Our experiment on Nakano central park

Figure 8: Our experiment on Okayama castle

5.1 Experiment at Nakano Central Park

With the cooperation of an industrial promotion organization in Nakano, we got a chance to provide a mobile video-on-demand service on practical field.

Before the full experiment, we did a preliminary experiment on July 2nd, 2014 at Nakano central park. In the preliminary experiment, we provided a mobile video-on-demand service using our developed Metreamer for 2 clients. We used the broadcasting equipment installed in the park. The broadcasting equipment uses wireless LAN. In the environment for the preliminary experiment, the client could receive the data both from the broadcasting system and the communication system when they were in the broadcasting area. In case where the clients were out of the broadcasting area, the clients could receive the data only from the communication system. We checked the number of interruptions using the measuring function of Metreamer and confirmed that Metreamer realized zero interruption in the case where the clients were in the broadcasting area. However, otherwise, interruptions frequently occurred. This was because many electric waves were emitted around the park and the communication bandwidth decreased down to the bit rate frequently. So, we decreased the bit rate of the video and did an experiment again on July 17th. In the experiment, Metreamer realized zero interruption even where the clients were out of the broadcasting area. To further investigate the performances of Metreamer, we did a full experiment on a large event.

Considering time and scale, the full experiment was done on Tohoku-Fukkou-Daisaiten-Nakano held on Oct. 25th and 26th, 2014 at the same place. The situations are shown in Fig. 7. For the experiment, we made the press release shown in the figure. To show the press release as it is, this is Japanese. The event is the largest one in Nakano area and the attendees are 170,000. Nebuta (large paper made statures) moves around the park during the event period. To deliver the video that is interesting for the attendees, we used the movie for moving Nebuta on the last year on the first day and that on the first day on the second day. The duration of the video was 2 minutes and 1 second, and the bit rate is 2Mbps. We set wireless LAN access point. The mobile devices of the attendees can receive the data both from the broadcasting system and the communication system by making them connect to the access point. We got booth, and there, the attendees watched the video using Metreamer. We asked the attendees answer the questionnaire after watching the video. We describe the detail of the questionnaire in Subsection 5.4. For the performance comparison, we also provide a video-on-demand service using conventional Windows Media System. In the environment for the full experiment, the video soon interrupted when the mobile devices started playing the video using the system. On the other hand, our developed Metreamer realized zero interruption even when about 10 clients connect to the system.

Table 1: System specification of Okayama experiment

Delivery Server

OS	Microsoft Windows 7 Professional SP1
CPU	Intel Core2 duo 2.5 GHz
Memory	4 GB
Ethernet	1000BASE-T

Wireless LAN

Access Point	Buffalo WAPM-AG300N
Wireless Standard	IEEE 802.11g
Additional Antenna	Buffalo WLE-HG-SEC

Clients for Demonstration

Google Nexsus 7 (2013, Android 4.3)
Sony Xperia C (C2305, Android 4.2.2)

Video Data

Coding	MPEG2-TS, H.264, AAC
Bit Rate	1Mbps
Frame Size	1280x720

5.2 Experiment at Okayama Castle

With the cooperation of Okayama city, we got a chance to provide a mobile video-on-demand service at Okayama castle. Considering time and scale, the experiment was done on Imagineering OKAYAMA ART PROJECT held on Nov. 22th, 2014 at Torishiro park in Okaya castle. This is the event to demonstrate arts made out and in Okayama. The situations are shown in Fig. 8. We set a wireless LAN access point and

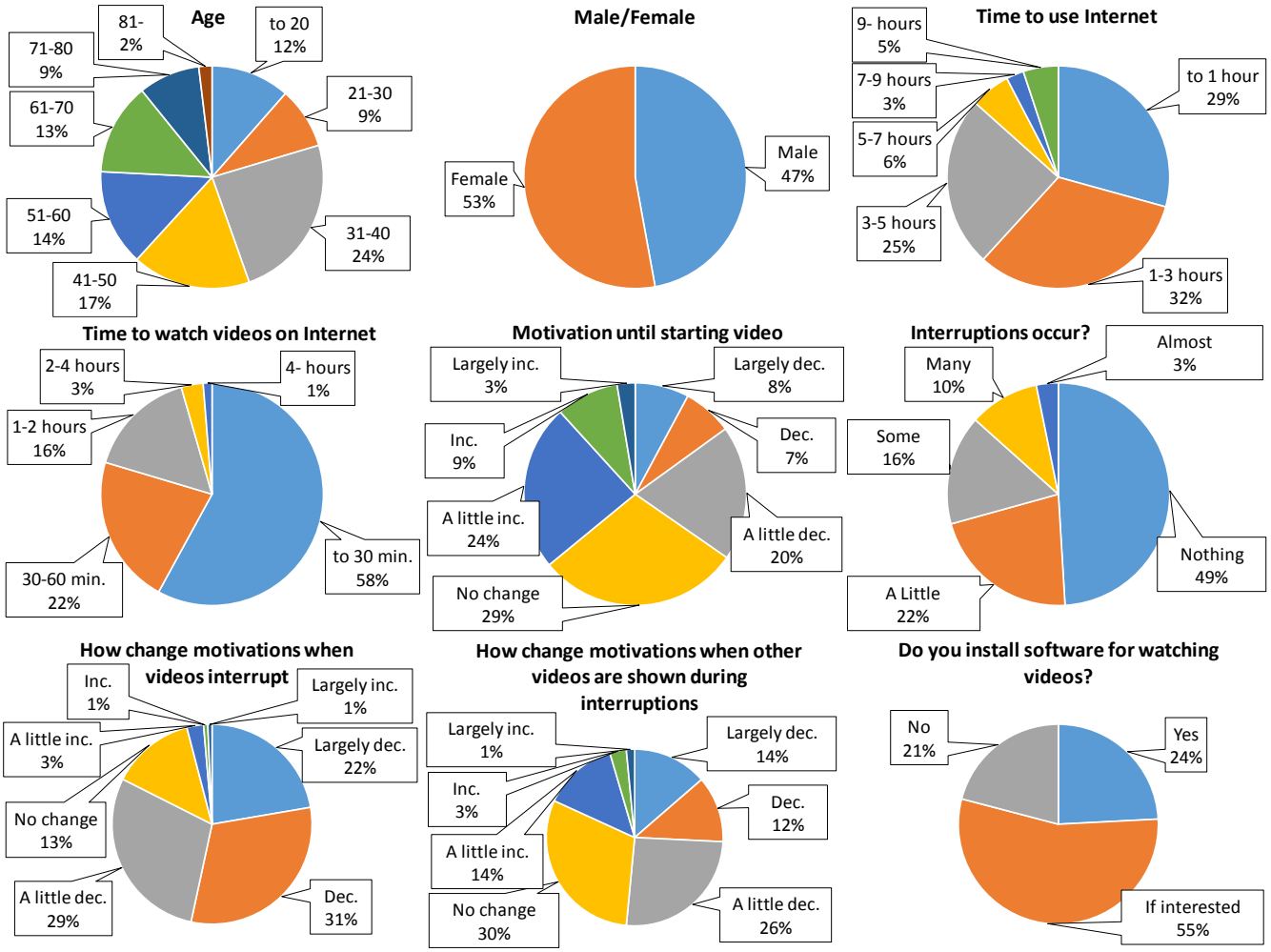


Figure 9: Questionnaire Results

booth in Okayama castle. Since Okayama castle does not have broadcasting equipment, we used the access point for the communication system and the broadcasting system. Table 1 shows the system specification for the experiment in Okayama. Okayama city has a PR of the city and uses it in some events. So, we delivered the video for the experiment. The duration was 3 minutes and 30 seconds and the bit rate is 1 Mbps. Same as the experiment at Nakano central park, the mobile devices of the attendees can receive the data both from the broadcasting system and the communication system by making them connect to the access point. We asked the attendees answer the questionnaire after watching the video at our booth. In this environment, the video soon interrupted when the mobile devices could not receive the data from the broadcasting system. On the other hand, our developed Metreamer realized zero interruption even when a few clients connect to the system.

5.3 Evaluation from Experiments

The experiments at Nakano central park was an experiments for a big event. Many people attended to the event during the period. Moreover, we delivered the video of moving Nebuta, which is the main event and many attendees were interested in. So, 16 clients connect to Metreamer at maximum. In the experiment, we made the situation that the clients could receive the data only from the communication system by moving them to the out of the broadcasting area. Even in this case, we confirmed that Metreamer realized zero interruption. Also we made the situation that the clients could receive the data only from the broadcasting system by disabling data transfer via the communication system. Even in this case, we confirmed that Metreamer realized zero interruption.

The experiments at Okayama castle was an experiments for a middle scale event. Only a few people were there sometimes. So, 7 clients connect to Metreamer at maximum. This is smaller than that of Nakano central park. Also in the experiment, we confirmed that Metreamer realized zero interruption even where the clients could receive the data only from the communication system or the broadcasting system.

5.4 Questionnaire Results

We got 275 questionnaire results from the experiments. The results are shown in Fig. 9.

First, regarding the statistics for the respondents, the most of the respondents' age is 31-40 years old and this is 24%. Female is more than male and is 53% of the respondents. The most of the time to use the Internet is 1-3 hours and is 32%.

In the respondents of using the Internet, 58% watch videos on the Internet up to 30 min and this is the most case.

To check the effectiveness of reducing the interruptions by the stream merge technology, we investigated the change of the motivation to watch videos. It takes a few seconds to start playing the video after the users tap the play button for the reason of starting the communication between the clients and the server. So, we asked the change of the motivation until staring playing the video. For the questionnaire, 29% of the respondents answered `no change'. We can say that the waiting time for starting playing the video after selecting the video does not influence the motivation largely. In the experiments, the video was not interrupted for 49% respondents. But, some respondents encountered the interruptions. In them, 82% of the respondents answered that their motivations decrease (largely, middle, a little) when the video interrupts. This is about 16 times more than that of increasing (total 5%). That is, our goal, reducing the interruptions, is significant to keep their motivations. One of the reasons why the motivation increases by the interruptions is that the users have an interest on the subsequent video contents. The motivations of 13% respondents did not change.

To check the effectiveness of the spare data delivery technique, we asked the change of the motivation when other videos are shown during interruptions. For the questionnaire, 52% of the respondents answered that their motivations decrease. The result is not a comparison with the video delivery without spare data delivery and the reason is simple. Their motivations decreased because of the video interrupts. If the system does not show the spare data during the interruptions, the ratio may increase. The motivations for 30% respondents did not change even when other videos are shown during interruptions. So, showing the spare data does not decrease the motivation largely.

It is difficult to check the effectiveness of the remaining battery based bit rate technique by questionnaires because the technique only changes the bit rate and the video quality decreases only thinking from the user side. However, we confirmed that the bit rate changes based on the remaining battery according to the user's setting and the technology worked on Metreamer.

To use Metreamer, the users have to install the software to receive the data from the server to their mobile devices. For the questionnaire, 55% of the respondents answered that they install the software if the system provides their interested videos. So, we can say that whether the users install a new software or not depends on the provided video contents.

6 CONCLUSION

Due to the recent proliferation of mobile devices and video-on-demand delivery, mobile video-on-demand delivery gets great attention. In this paper, aiming to zero interruption, we proposed 3 techniques for hybrid broadcasting environments. We developed a mobile video-on-demand system using our proposed techniques called Metreamer. In this paper, we reported the experiments of mobile video-on-demand delivery using our developed system.

In the future, we will again show the effectiveness of our proposed techniques by computer simulation. Also, we will develop the system for multiple streaming servers and live broadcasting.

REFERENCES

[1] V. Gopalakrishnan, B. Bhattacharjee, K. Ramakrishnan, R. Jana, and D. Srivastava, "CPM: Adaptive Video-on-Demand with Cooperative Peer Assists and Multicast," IEEE INFOCOM 2009, pp. 91-99 (2009).
[2] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-Driven Mesh-based Streaming," Proc. of IEEE INFOCOM 2007, pp. 1415-1423 (2007).
[3] H. Kim and H. Y. Yeom, "Dynamic Scheme Transition Adaptable to Variable Video Popularity in a Digital Broadcast Network," IEEE Trans. on Multimedia, Vol. 11, No. 3, pp. 486-493 (2009).
[4] S. Kulkarni, J.-F. Paris, and P. Shah, "A Stream Tapping Protocol Involving Clients in the Distribution of Videos on Demand," Springer Advances in Multimedia, Special Issue on Collaboration and Optimization for Multimedia Communications, Vol. 2008 (2008).
[5] T. Taleb, N. Kato, and Y. Nemoto, "Neighbors-Buffering-Based Video-on-Demand Architecture," Signal Processing: Image Communication, Vol. 18, Issue 7, pp. 515-526 (2003).
[6] J. B. Kwon, "Proxy-Assisted Scalable Periodic Broadcasting of Videos for Heterogeneous Clients," Multimedia Tools and Applications, Springer, Vol. 51, No. 3, pp. 1105-1125 (2011).
[7] T. Yoshihisa, "A Data Segments Scheduling Method for Streaming Delivery on Hybrid Broadcasting Environments," Proc. of International Workshop on Informatics (IWIN2015), pp. 3-8 (2015).
[8] YouTube, http://www.youtube.com/.
[9] Ustream.tv:You're On, http://www.ustream.tv/.
[10] niconico, http://www.nicovideo.jp/.
[11] Netflix, http://www.netflix.com/.
[12] Hulu - Watch TV, Original, and Hit Movies, http://www.hulu.com/.
[13] Actvila, http://actvila.jp/.

(Received October 10, 2016)
(Revised April 8, 2017)

Tomoki Yoshihisa received his Bachelor's, Master's, and Doctor's degrees from Osaka University, Osaka, Japan, in 2002, 2003, 2005, respectively. Since 2005 to 2007, he was an assistant professor at Kyoto University. In January 2008, he joined Cybermedia Center, Osaka University as a senior lecturer and in March 2009, he became an associate professor. From April 2008 to August 2008, he was a visiting researcher at University of California, Irvine. His research interests include video-on-demand, broadcasting systems, and webcasts.

Tomoya Kawakami received his B.E. degree from Kinki University in 2005 and his M.I. and Ph.D. degrees from Osaka University in 2007 and 2013, respectively. From 2007 to March 2013 and from July 2014 to March 2015, he was a specially appointed researcher at Osaka University. From April 2013 to June 2014, he was a Ph.D. researcher at Kobe University. Since April 2015, he has been an assistant professor at Nara Institute of Science and Technology. His research interests include distributed systems, rule-based systems, and stream data processing. He is a member of the IPSJ and IEEE.

Yusuke Gotoh received a Bachelor's degree from Okayama University, Okayama, Japan, in 2005 and Master's and Doctor's degrees from Kyoto University, Kyoto, Japan, in 2007 and 2009, respectively. Since April 2009, he has been an Assistant Professor at the Department of Information Technology, Faculty of Engineering, Okayama University, Okayama, Japan. From April 2011 to November 2012, he was a Visiting Researcher at La Trobe University, Melbourne, Australia, as a JSPS Postdoctoral Fellow for Research Abroad. His research interests include broadcast computing, scheduling, and spatial computing.

Submission Guidance

About IJIS

International Journal of Informatics Society (ISSN 1883-4566) is published in one volume of three issues a year. One should be a member of Informatics Society for the submission of the article at least. A submission article is reviewed at least two reviewer. The online version of the journal is available at the following site: http://www.infsoc.org.

Aims and Scope of Informatics Society

The evolution of informatics heralds a new information society. It provides more convenience to our life. Informatics and technologies have been integrated by various fields. For example, mathematics, linguistics, logics, engineering, and new fields will join it. Especially, we are continuing to maintain an awareness of informatics and communication convergence. Informatics Society is the organization that tries to develop informatics and technologies with this convergence. International Journal of Informatics Society (IJIS) is the journal of Informatics Society.

Areas of interest include, but are not limited to:

- Internet of things (IoT)
- Smart cities, communities, spaces
- Big data, artificial intelligence
- Data science
- Computer supported cooperative work and groupware
  Intelligent transport system
- Distributed computing
- Multi-media communication
- Information systems Mobile computing
- Ubiquitous computing

Instruction to Authors

For detailed instructions please refer to the Authors Corner on our Web site, http://www.infsoc.org/.

Submission of manuscripts: There is no limitation of page count as full papers, each of which will be subject to a full review process. An electronic, PDF-based submission of papers is mandatory. Download and use the LaTeX2e or Microsoft Word sample IJIS formats.

http://www.infsoc.org/IJIS-Format.pdf

LaTeX2e

LaTeX2e files (ZIP) http://www.infsoc.org/template_IJIS.zip

Microsoft Word™

Sample document http://www.infsoc.org/sample_IJIS.doc

Please send the PDF file of your paper to secretariat@infsoc.org with the following information:

Title, Author: Name (Affiliation), Name (Affiliation), Corresponding Author. Address, Tel, Fax, E-mail:

Copyright

Publisher

Address: Informatics Laboratory, 3-41 Tsujimachi, Kitaku, Nagoya 462-0032, Japan

E-mail: secretariat@infsoc.org

CONTENTS