**Journal of Information Processing Vol.27 701–710 (Nov. 2019)**

[DOI: 10.2197/ipsjjip.27.701]

**Recommended Paper**

# A Method to Reduce Transaction Time for Real-time IoT Applications

Chaxiong Yukonhiatou[1,a] Tomoki Yoshihisa[2,b] Tomoya Kawakami[3,c] Yuuichi Teranishi[4,2,d]
Shinji Shimojo[2,e]

Received: January 31, 2019, Accepted: August 9, 2019

***Abstract:*** Due to the recent prevalence of IoT (Internet of Things) devices, streaming data such as video data or sensor data are collected and analyzed for real-time applications. The transaction time for data collection and analysis is one of the main factors to improve performance of some real-time IoT applications. For instance, in real-time surveillance camera systems, a shorter transaction time further reduces the delay to find thieves recorded by the cameras. However, computational and communication capacities of processing computers give lower limits to transaction time. To break this limitation, we propose an efficient data collection method using a progressive quality improvement approach. In our proposed method, each data source produces some content data of those qualities are lower than the quality of the original content data such as low resolution image data. Only the cases where higher quality data are needed for analyses, the processing computer progressively collects them. Thus, by reducing the average data amount of collections and for analyses, our proposed method reduces the average transaction time. We measured the transaction time of our proposed method in our developed simulator and confirmed that our proposed method can reduce the transaction time.

***Keywords:*** stream processing, stream data, communication delay, progressive quality improvement

## 1. Introduction

Recently, various IoT devices such as cameras and weather sensors connected to the Internet attract great attention. These devices generate stream data such as video data or temperature data and act as stream data sources. In most IoT applications, processing computers collect stream data from these IoT devices continuously and analyze them in real-time. The transaction time for data collection and analysis (from the time to start transmitting a content data from a stream data source to the time to finish analyzing the content data) is one of the main factors to improve performance of some real-time IoT applications. For example, suppose a system in that a processing computer receives video data continuously from some surveillance cameras and analyzes image data of each video frame to find thieves. In this example, a shorter transaction (image data receptions and analyses) time enables the system to grasp their more recent positions.

Larger computational and communication capacities ofa processing computer further reduce the transaction time. However, these are actually limited and give lower limits to the transaction time. In stream data processing, a long transaction time has a possibility to increase the transaction time continuously since the transaction time increases if the time span of a transaction overlaps with that of the next transaction. A longer transaction time also requires a larger amount of data buffers since the collected data are stored to the buffer of the processing computer while it collects and analyzes them. Therefore, transaction time reduction is one of the main research topics for real-time IoT applications.

To reduce the transaction time, many methods have been proposed [1], [2], [3]. Most of these methods degrade the qualities of the content data to complete analyses, such as resolutions for image data to reduce the data amount to collect. Quality degradations result in performance degradations of IoT applications. Their performance can improve by reducing data to be collected even when the analysis frequency is high or there is a large number of stream data sources.

In this paper, we propose an efficient data collection method using a progressive quality improvement approach. In our proposed method, each stream data source produces some content data of those qualities are lower than the quality of the original content data such as low resolution image data. Only the cases where higher quality data are needed for analyses, processing computers progressively collect them (a detailed explanation is in Section 4). Thus, by reducing the average data amount of collections and for analyses, our proposed method reduces the average transaction time. Here, the transaction time is afactor that affects the network

1 Graduate School of Information Science and Technology, Osaka University, Ibaraki, Osaka 567–0047, Japan
2 Cybermedia Center, Osaka University, Ibaraki, Osaka 567–0047, Japan
3 Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
4 National Institute of Information and Communications Technology, Koganei, Tokyo 184–8795, Japan
a) chaxiong@ais.cmc.osaka-u.ac.jp
b) yoshihisa@cmc.osaka-u.ac.jp
c) kawakami@is.naist.jp
d) teranisi@cmc.osaka-u.ac.jp
e) shimojo@cmc.osaka-u.ac.jp

The preliminary version of this paper was published at Multimedia, Distributed, Cooperative, and Mobile Symposium (DICOMO 2018), July 2018. The paper was recommended to be submitted to Journal of Information Processing (JIP) by the chief examiner of SIGDPS.

bandwidth or delays. A larger data amount causes using a large network bandwidth, this causes a longer transaction time since the time starting the next transaction is longer. In other hand, if a smaller data amount causes using less network bandwidth, this causes a shorter transaction time. Different from a traditional approach in that a transaction sequentially proceeds with improving the quality of the content data, our progressive quality improvement approach targets streaming data and the transactions occur continuously. In stream data transactions, it is difficult to reduce the transaction time since they continue to increase when the time span for a transaction overlaps with that for the next transaction. Therefore, as we stated above, the transaction time reduction is an important performance for stream data transactions. In our proposed method, the processing computer sets a priority to reduce the transaction time (stops redundant processes, executes processes one by one) not to fail the stream data transactions. The contributions of the paper are: 1) the proposal of a progressive quality improvement (PQI) approach, 2) the proposal of a method using the PQI approach.

The rest of this paper is organized as follows. In Section 2, we introduce some works that are related to our proposed method. In Section 3, we explain our assumed system environments. Our proposed method is explained in Section 4, and evaluated in Section 5. Finally, we will conclude the paper in Section 6.

## 2. Related Works

To reduce communication traffic and improve stream data analysis for real-time, some methods have been proposed.

A two-layer system architecture for stream data analysis is proposed in Ref. [4]. In the first layer, the system executes pre-analyses to received data and determines whether to proceed to the main analysis of the second layer. The proposed system architecture can reduce processing loads since the system does not execute redundant main processes. Though the method divides processes, data are not divided into some parts as in our proposed method.

A two-level indexing structure for data collection is proposed in Ref. [5]. In this method, the data are first stored to the memory having tree structures in the first level and then each data segment passes to the second level with their reference key tree. The method can faster collect data due to each data segment being stored separately. Their method is different from ourmethod in the point that they reduce the loads of the processing computers but we reduce the transaction time.

In Ref. [6], queuing models for processing stream data are analyzed for improving the transaction time and a queuing method is proposed. In the method, the received stream data are stored to the buffer of processing computers. Processing computers use different buffers for each application. The method reduces the transaction time considering queuing situations of other buffers. Our proposed method is different from this in the point that we reduce the transaction time by managing how to process data.

In Ref. [7], a dynamic bitrate adaptationmethod is proposed. In this proposed method, theappropriate bitrate is selected so as not to exceed the communication buffer of the processing computer. In the processing computer, the data is divided into series of segments then sent to the requested users. However, the method does not aim to reduce the transaction time.

An efficient CPU resource allocationmethod for stream data analysis is proposed in Ref. [8]. By allocating CPU resources to each data stream and processing received stream data in a single process manner, the method enables faster stream data analysis. The approach of themethod is effective CPU resource allocations and is different from our approach. In our approach, the transaction time is reduced by improving the quality of contents data progressively.

A stream split processing model was proposed in Ref. [9]. This model allocates data stream into two types such as normal data and delay data. Each type of allocated data is executed separately. The normal data gets faster for execution since the waiting time for the next execution is shorter while the execution of the delay data is long. However, this method has to share the output to each other after complete execution of the normal data and delay data in order to get the final results. Therefore, the average transaction time to achieve the final results of this method cannot reduce. Our method reduces the average transaction time.

A method for optimizing data transaction and computation was proposed in Ref. [10]. In this method, the processing computer collects only the image that contains the changed contents of image data. By ignoring some duplicated contents of image data, the method can keep transaction rates. This method is similar to our method in the case that the processing computer stores only the different contents of image data from the previous image data. However, the quality of image in their method is not improved. In our proposed method, the quality of image is improved progressively.

The method proposed in Refs. [11] and [12] reduces the bandwidth consumption and the amount of transmitted data keeping the quality of stream data and communication delays by compressing stream data. One of the drawbacks of the method is that the data sources need to compress the data before their transmissions and this causes further transaction time.

In addition, some stream data analysis systems have been developed in Refs. [13], [14]. However, the quality of stream data is fixed under these methods. Our proposed approach in this paper improves the quality progressively.

## 3. Assumed System

In this section, we explain our assumed system.

### 3.1 System Architecture

**Figure 1** shows our assumed system architecture. An user designates processes for continuously generated data (stream data) to a processing computer. The processing computer executes designated processes at every data reception. Such a type of processes is called stream processing. The processing computer gathers necessary data for processes and executes processes continuously. The processing computer has a buffer for storing received data and executes processes for the data.

Some IoT devices such as surveillance cameras continuously get data about their observations such as video data and act as stream data sources. They and the processing computer connect

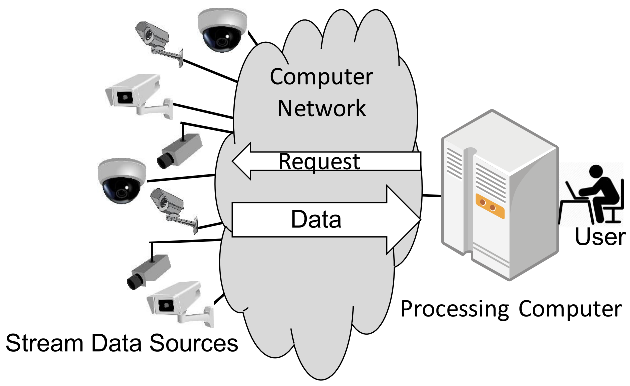

**Fig. 1** Our assumed system architecture.

to a computer network. In the cases that the network bandwidth is stable, the Internet can be assumed to be the computer network. The data sources and the processing computer can communicate with each other via the computer network. The data sources divide their generated stream data into some parts and store them to their buffer temporarily. When the processing computer requests data to stream data sources, the requested data sources return it to the processing computer. The processing computer receives the requested data in its own buffer and performs processing.

## 3.2 Application Scenario

In this subsection, we introduce an application scenario. Suppose an area where some surveillance cameras are deployed and a processing computer gathers their recorded video frame data. They connect to a designated computer network and communicate with each other similar to our assumed system architecture.

As an example application scenario, we assume a thieves finding system by a face recognition. For this, the application designates the process that notifies to the user when the processing computer detects the humans whose faces resemble to thieves' faces in the video data got from surveillance cameras. To detect faces, the user submits the face images of thieves to the processing computer beforehand. The processing computer continuously analyzes image data got from surveillance cameras and identifies faces in received image data. When the processing computer finds faces in an image data, it checks whether the found faces are those of thieves' faces. If the processing computer detects thieves' faces, it sends a notification to the user by e-mail or other messaging services.

The transaction time in this scenario is from the time to start transmitting an image data recorded by each camera to the time that the processing computer finishes checking thieves' faces. One of the simple approaches to reduce the transaction time is that the processing computer does not identify faces when the difference between an image and the previous video frame is excessively small since the image does not change largely. Our proposed method requires network connections among cameras and computers, but it does not use any information about those networks such as bandwidth, delay, and hop counts. Therefore, the application providers can deploy the system using our proposed method with the network connections. The reduction of transaction time has an advantage also in the application domain for the viewpoint of resources usage including energy consumption.

We assume that the network connections among cameras and computers, and those network performances are given such as bandwidth and delay. Our assumed problem is caused by these limited computational and communication capacities, and a longer communication time lengthens the transaction time. This is because the communication time strongly influences the transaction time in many cases compared to the time for local processes. A longer transaction time requires higher costs in both of computational and communication domains. Therefore, we propose a method to reduce the transferred data by collecting higher quality data only in the cases that they are requested. The reduction of the transferred data enables to shorten the communication time in the transaction time and save the network resources consumed. In addition, a shorter transaction time has an advantage also in the application domain for the limited computational resources including energy consumption.

## 3.3 Transaction Time Definition

In this section, we explain the transaction time. The transaction means the processes to complete sending the reserved data item in each sending sequent between data sources and the processing computer. We call the processes for receiving all divided data of a data item a "transaction". Each transaction takes some time to transmit the data item from the data sources to the processing computer. Therefore, the transaction time in our paper means the time to get the first level of a divided data item to the time to finish the processing of the last level of a divided data item.

## 3.4 Research Objective

In the scenario introduced in Section 3.2, the application performance is the probability to catch thieves. This can increase by a shorter transaction time since the system can grasp their more recent positions. Moreover, the probability to detect thieves increase by collecting video data from more surveillance cameras.

However, computational and communication capacities ofthe processing computer are limited. Therefore, a more frequent data analysis and also a larger number of data sources cause a longer transaction time. A longer communication time lengthens the transaction time from the time to start transmitting a content data at a stream data source to the time to finish analyzing them. Therefore, our research objective is reducing the transaction time keeping the application performance.

## 3.5 Mathematical Definition

In this subsection, we explain a mathematical model for our assumed system. Suppose that the system has $N$ stream data sources. These stream data sources cyclically send their observed data every $C_n$ ($n = 1, \cdots, N$) unit time. Let $D_{n,a}(t)$ denotes the whole stream data at $t$th cycle while $n$ denotes the number of streams. The $a$ represents "all" and does not have any specific value. Whole stream data mean original stream data without dividing them into some qualities. The system can divide $D_{n,a}(t)$ into $Q$ data $D_{n,q}(t)$ which is $q$ th quality data of $D_{n,a}(t)$. The data amount of $D_{n,q}(t)$ is denoted by $S_{n,q}(t)$. $GT_{n,q}(t)$ denotes the generation time of $D_{n,q}(t)$ and $P_{n,q}(t)$ denotes the time required to process it. $ST_{n,q}(t)$ is the time to start processing

$D_{n,q}(t)$ and $FT_{n,q}(t)$ is the time to finish processing it. Here, $FT_{n,q}(t) = ST_{n,q}(t) + P_{n,q}(t)$. The transaction time, $TT_n(t)$, is the time from data generation to end of processing the data. In the case that the processing computer processes the divided data sequentially from the first quality data $D_{n,1}(t)$ to the $e_n(t)th$ quality data $D_{n,e_n(t)}(t)$, $TT_n(t)$ is given by the following equation:

$$TT_n(t) = FT_{n,e_n(t)}(t) - GT_n(t) \tag{1}$$

The average transaction time for the data source $n$ is:

$$\frac{1}{T}\sum_{t=1}^{T} TT_n(t) \tag{2}$$

where $T$ is the final cycle. The objective is maximizing the number of data sources to process, which corresponds to minimizing Eq. (2).

We denote the probability for processes to proceed to the next process $PProb_{n,p}(t)$ $(p = 1, \cdots, Q-1)$. For example, the probability to request $D_{1,2}(1)$ when the processing computer finishes processing $D_{1,1}(1)$ is $PProb_{1,1}(1)$.

## 4. Proposed Method

In this section, we explain our proposed method.

### 4.1 Basic Idea

Generally, data have some qualities. Data analyses can be applied for each quality and data with the highest quality often give the best performance for analyses. For example, one of qualities for image data is resolution. Image data with $640 \times 480$ pixel size has a higher quality than image data with $320 \times 240$ pixel size. Image analyses to find faces can be applied for various pixel sizes while a higher resolution image data generally gives a higher accuracy. By finally analyzing data with the highest quality, applications can achieve the same performance. When processing computers analyze data sequentially in the order of quality from the lowest to the highest, they can stop data analyses when the subsequent analyses for higher quality data are meaningless. For example, same as the example in the introduction section, suppose the case that a processing computer analyzes image data of each video frame to detect faces. The processing computer first receives the lowest quality image data of a frame and analyzes the difference from the previous frame. In case that the difference is small, the processing computer skips the analysis of higher quality image data since new humans do not appear in those frames because of small differences. In such cases, the processing computer does not need to receive higher quality image data when subsequent analyses are meaningless. Therefore, by analyzing data in the order of data quality and stopping analyses when subsequent analyses are meaningless, the processing computer can skip the receptions of higher quality image data.

In cases that the probability to proceed to analyses of higher quality image data is small, the total amount of received data is reduced, compared with the case that all quality data are received. Generally, the amount of a lower quality data is smaller. Therefore, the data amount to be collected by the processing computer is further reduced than that under the conventional method when

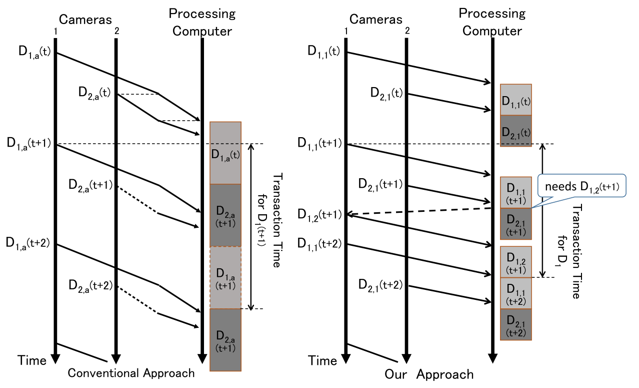

**Fig. 2** Stream data collection of an conventional approach and of our proposed approach.

the probability is small. Thus, the transaction time is reduced keeping the application performance. We call this approach *progressive quality improvement* approach.

### 4.2 Data Stream Processing

In this subsection, we explain the processes of data collection in the conventional approach and our proposed approach. In this example, the data sources are two cameras.

We first explain data streams transaction under the conventional approach. Camera 1 sends its recorded video data to the processing computer. It sends image data for each video frame each time it acquires it. In **Fig. 2**, the $t$ th frame data is shown by $D_{1,a}(t)$ $(t = 1, \cdots, T)$. For example, when the frame rate is 10 [Hz], Camera 1 sends image data every 100 [msec.]. Hence, $GT_{1,a}(t+1) = GT_{1,a}(t) + 0.1$. In addition, Camera 2 sends its recorded video data to the processing computer. In this example, the frame rate for Camera 2 is the same as that for Camera 1, but the time to start sending the video data differs. After Camera 1 sends $D_{1,a}(t)$, Camera 2 sends $D_{2,a}(t)$. While both Camera 1 and Camera 2 send data, the input communication bandwidth for the processing computer is equally divided between them. Therefore, the transaction speed of Camera 1 decreases as shown in the figure. After Camera 1 finishes sending $D_{1,a}(t)$, the input communication bandwidth is dedicated for the transaction with Camera 2 and the transaction speed of Camera 2 increases as shown in the figure. When the processing computer finishes receiving $D_{1,a}(t)$, it starts processing $D_{1,a}(t)$. While processing $D_{1,a}(t)$, the processing computer finishes receiving $D_{2,a}(t)$. Since the processing computer processes $D_{1,a}(t)$ at this time, it stores the received $D_{2,a}(t)$ in its transaction buffer and starts processing it after finishing processing $D_{1,a}(t)$. Similarly, while processing $D_{2,a}(t)$, the processing computer finishes receiving $D_{1,a}(t+1)$. The processing computer starts processing it after finishing processing $D_{2,a}(t)$. The transaction time for $D_{1,a}(t+1)$ in this case is shown in the figure. This is the time from the start of sending $D_{1,a}(t+1)$ to the end of processing $D_{1,a}(t+1)$.

Next, we explain data streams processing under our proposed approach. Similar to the example for the conventional method, Cameras 1 and 2 send their recorded image data to the processing computer cyclically. In contrast with the conventional method, the image data is divided into 2 levels. Each level has a different quality. For example, the data for the first level is the lowest

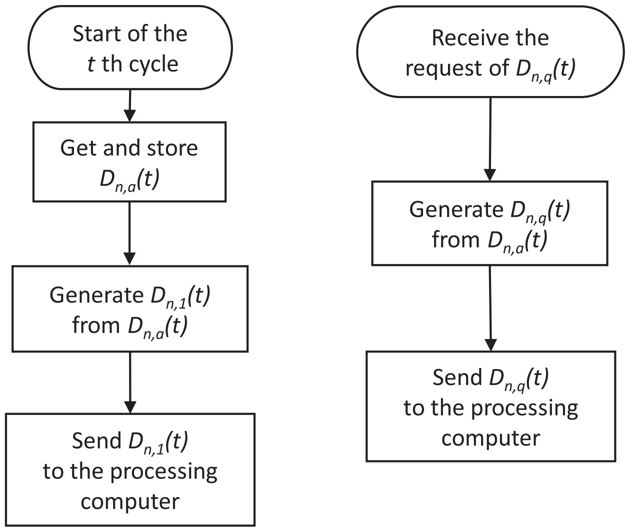
Start of the
t th cycle
Get and store $D_{n,a}(t)$
Generate $D_{n,1}(t)$ from $D_{n,a}(t)$
Send $D_{n,1}(t)$ to the processing computer
Receive the request of $D_{n,q}(t)$
Generate $D_{n,q}(t)$ from $D_{n,a}(t)$
Send $D_{n,q}(t)$ to the processing computer

**Fig. 3** Processes in data sources.

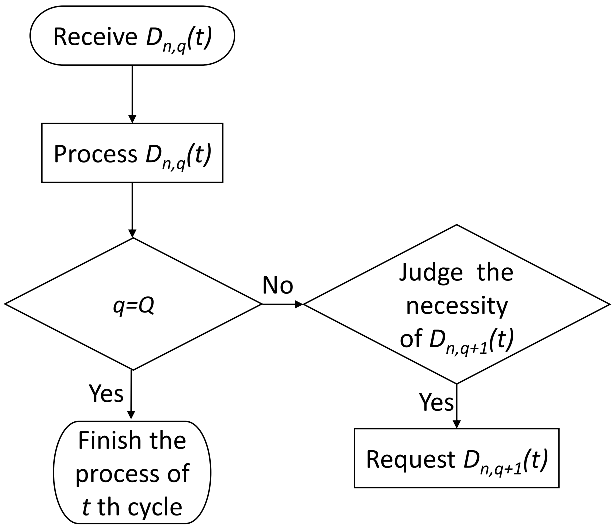

**Fig. 4** Dataprocessing in the processing computer.

quality image data and the data for the second level is an additional data to improve the quality of the data. Here, we assume that the data for the second level only includes the difference data from the first level and that the amount of the data for each level is the same. To make the example simple, we assume that the data amounts for each level are just half of the data amount of $D_{1,a}(t)$ $(t = 1, \cdots, T)$. The time needed to send $D_{n,q}(t)$ $(n = 1, 2, q = 1, 2)$ is the half of the time needed to send $D_{n,a}(t)$. Therefore, the transaction of $D_{1,1}(t)$ does not overlap that of $D_{2,1}(t)$ though the transaction of $D_{1,a}(t)$ overlaps that of $D_{2,a}(t)$. An example of the reason not to request the second level data is that the difference of the image data from the previous image data is not so large. The processing computer does not request the second level data in the first cycle. In the $t + 1$ th cycle, the processing computer starts processing $D_{1,1}(t+1)$ after finishing receiving it. After processing $D_{1,1}(t + 1)$, the processing computer requests the second level data. An example of the reason to request the second level data is that the difference of the image data from the previous image data is large. When Camera 1 receives the request for $D_{1,2}(t + 1)$, it starts sending $D_{1,2}(t + 1)$. The processing computer does not request the second level data for the Stream 2 in this case. After receiving $D_{1,2}(t + 1)$, the processing computer processes it and completes the data analysis of the $t + 1$ th cycle. $D_{1,2}(t + 1)$ includes only the data different from $D_{1,1}(t + 1)$. By combining $D_{1,1}(t + 1)$ and $D_{1,2}(t + 1)$, we can get a higher data quality. The transaction time for $D_1(t + 1)$ in this case is shown in the figure. This is the time from the start of sending $D_{1,1}(t + 1)$ to the end of processing $D_{1,2}(t + 1)$.

In this case, the transaction time under our approach is shorter than that under the conventional approach since the processing time for $D_2(t)$ is reduced.

## 4.3 Algorithms

In this subsection, we explain the algorithms for our proposed approach.

### 4.3.1 Algorithm for Data Sources

**Figure 3** shows the flow chart of data sources. When the $t$ th cycle starts, each data source $n$ gets $D_{n,a}(t)$ from their sensors and stores it to their storages temporarily. First, they generate $D_{n,1}(t)$ from $D_{n,a}(t)$. The data sources cyclically send $D_{n,1}(t)$ to the processing computer with the interval $I$. The data sources send higher quality data when they receive the requests except for the transmission of the first quality data.

When the data source $n$ receives the request of $D_{n,q}(t)$, it generates $D_{n,q}(t)$ from stored $D_{n,a}(t)$ and sends $D_{n,q}(t)$ to the processing computer.

### 4.3.2 Algorithm for Processing Computer

**Figure 4** shows the flow chart of the processing computer. When the processing computer receives $D_{n,q}(t)$, it processes $D_{n,q}(t)$. When $q = Q$ and $D_{n,q}(t)$ is the final quality data, the transaction of $t$ th cycle finishes. Otherwise, the processing computer judges the necessity of $D_{n,q+1}(t)$. Here, the time to judge the necessity of $D_{n,q+1}(t))$ is affected by many factors such as the processing computer and image data. In this paper, however, we assume that the processing computer has a specific performance and the time for local processes is not so long compared to the communication time. In case where $D_{n,q+1}(t)$ is needed for the process execution, the processing computer requests $D_{n,q}(t)$ to the data source $n$.

Note that the receptions of the first quality data are push-based receptions from the data sources. The receptions of higher quality data are pull-based receptions by the processing computer.

### 4.3.3 How to Divide Data

In our proposed method, each stream data is divided into some qualities. We suppose two approaches to divide data.

The first one is the case where a higher quality data can be constructed by combining some data. For example, image data with $640 \times 480$ pixel size can be constructed by 4 image data with $320 \times 240$ pixel size. In this case, the data amount of the $q$ th quality data of the stream data $n$ at the $t$ th cycle is given by:

$$\sum_{i=1}^{q} S_{n,i}(t) + \alpha_{n,i}(t) \tag{3}$$

Here, $\alpha_{n,i}(t)$ is an overhead caused by combining data.

The second one is the case where a higher quality data is constructed separately. For example, it is difficult to fully decode image data with $640 \times 480$ pixel size by combining compressed 4 image data with $320 \times 240$ pixel size. In this case, the data amount of the $q$ th quality data of the stream data $n$ at the $t$ th cycle is given by $S_{n,q}(t)$.

The application of the stream processing system selects an appropriate method to divide data.

#### 4.3.4 How to Process Data

In cases where the processing computer executes processes in parallel, the transaction time for each transaction lengthens because the computational resource is divided between them. Therefore, in our proposed method, the processing computer executes processes one by one. The processing computer earlier executes a process for a faster received content data (FIFO manner).

## 5. Evaluation

In this section, we show evaluation results of our proposed method by using our developed simulator.

### 5.1 Evaluation Setup

In this evaluation, we assume the application explained in Section 3.2. Under our proposed method, the processing computer calculates difference values between the current frame image data ($D_{n,q}(u)$, $n = 1, \cdots, N$, $q = 1, \cdots, Q$, $u = 2, \cdots$) and the previous one ($D_{n,q}(u-1)$) only when the difference value of the previous quality $q-1$ is large. Finally, the processing computer detects faces in $D_{n,Q}(u)$ in the cases that the difference values of all the lower qualities data are large. In the cases that a difference value is small, the processing computer stops the image analysis and waits for the next frame. In our assumption for real applications, the condition that the clients wait for the next frame is based on a designated (threshold) for its quality, and the condition is different by applications, e.g., whether the current frame has a enough quality to detect humans. Instead of the condition to wait for the next frame, this simulation tentatively has the additional parameter called “final probability” $FProb$ as the probability to proceed to the final level. **Table 1** shows other parameters and those values.

Input Bandwidth is the input communication bandwidth for the processing computer. When the processing computer communicates with some data sources, the input bandwidth is fairly shared among data sources. Output Bandwidth is the output communication bandwidth of each data source. Input Data is the data amount of $D_{n,a}(t)$ ($n = 1, \cdots, N$, $t = 1, \cdots, T$).

To make the evaluation results as realistic, we use an open image dataset in ‘changedetection.net’ [15] named ‘pedestrians’. The dataset includes 1,099 image data of standard JPG type with $360 \times 240$ resolution. These images are the frames of a video data got from a surveillance camera. To extract some data with different qualities from one original image data, we converted them into progressive-JPG type. A progressive-JPG image includes some images with different qualities (called ‘scans’ in progressive-JPG) and we regard each of them as $D_{n,q}(t)$. We cannot show the images in the data set for the reason of copyrights. Therefore, to show the qualities for each scan in images of progressive-JPG type, we show some images of each quality in **Fig. 5**, those are different from the images we used in the evaluation section. As shown in Fig. 5, the quality progressively improves. We calculate the difference values for each frame using the *norm* function in OpenCV [16]. The average processing time is 66.4 [usec.] We got the difference values of static images (no large change) and assume that the processing computer proceeds to the calculation of the next quality only when the values differ 5% from the values for static images. Also, we detected faces in the highest quality data. The average processing time is 389 [msec.]

We simulate the stream processing system for 300 seconds and get the transaction time.

**Table 1** Parameter values.

| | |
|---|---|
| Input Bandwidth | 10 [Mbps] |
| Output Bandwidth | 10 [Mbps] |
| 1,099 frames Data | changedetection.net |
| Image Size | $360 \times 240$ |

### 5.2 Transaction Time for Actual Data

We show the transaction time for each transaction. We explained in Section 5.1, the processing computer calculates the difference values between $D_{n,q}(u)$ and $D_{n,q}(u-1)$. Only when the difference values for all of lower quality data are large, the processing computer detects faces in $D_{n,Q}(u)$. We set $Q = 10$ since the default number of scans for images of progressive-JPG is 10. The average data amounts for each quality are shown in **Table 2**. We separated all image data fairly into $N$ groups to get multiple actual video streams. **Figure 6** shows the evaluation result. The horizontal axis is transaction IDs. A transaction ID is given to each transaction in that the processing computer collects a video frame data and analyzes it. Transaction IDs are given to each transaction sequentially. ‘No PQI’ means a conventional method in that the data sources do not divide the original data. ‘PQI’ is our proposed method.

Our proposed PQI method relatively gives shorter transaction time than those under the conventional method because the processing computer does not collect and analyze redundant data by dividing the original data into some qualities. In case that the

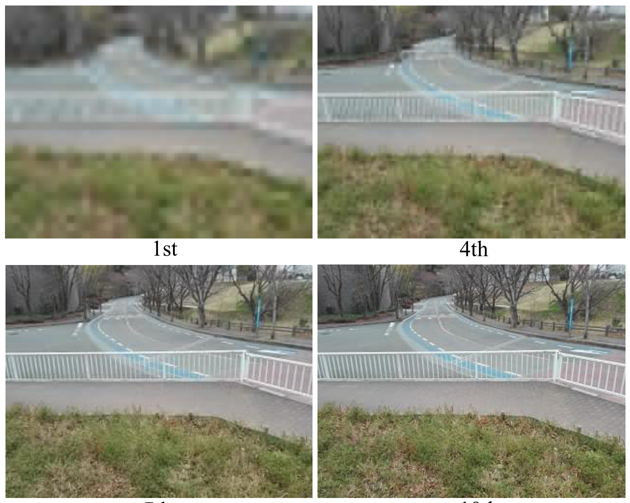

**Fig. 5** Resolution examples for some scans of progressive-JPG images.

**Table 2** Average data amounts for each scan [bytes].

| Scans | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Data amounts | 7204 | 1739 | 177 | 421 | 1630 | 3571 | 297 | 67 | 134 | 1421 |

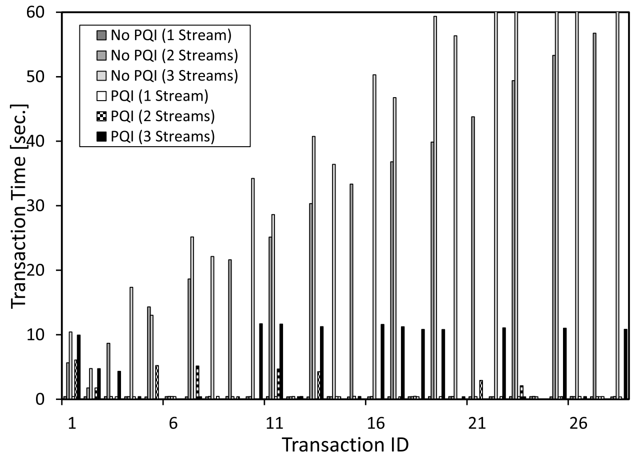

**Fig. 6** Transaction time of each transaction for an actual data.

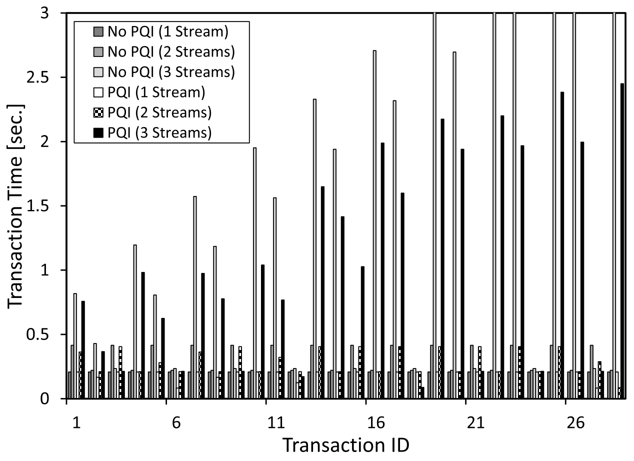

**Fig. 7** Transaction time of each transaction for a simulated data.

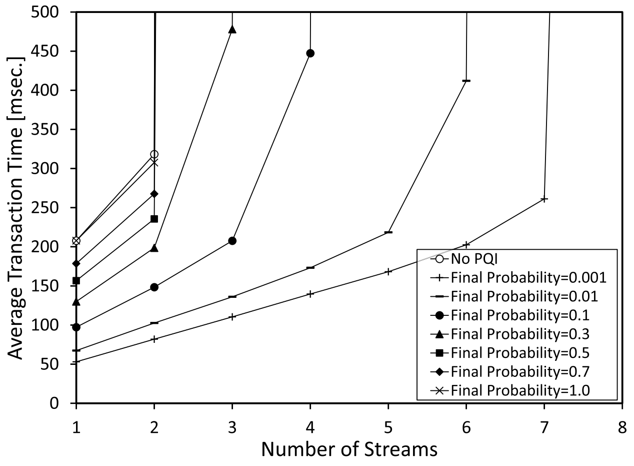

**Fig. 8** Average transaction time under different final probabilities.

difference value from the previous frame image is small under the PQI method, the process does not proceed to the final quality and the transaction time becomes short. Otherwise, the transaction time is long. A larger number of the streams gives a longer transaction time since the data amount that the processing computer collects and analyzes increases. Even when the number of the streams is 3, the transaction time does not diverge under the PQI method because some processes stop at lower qualities. In the conventional method, the transaction time diverges when the number of streams is 2 or 3.

## 5.3 Transaction Time for Simulated Data

We change some parameters to investigate the change of the transaction time under the conventional No PQI method and our proposed PQI method. For that, we simulate situations for stream processing based on the actual data used in the Section 5.2.

We use the same data amount and the same processing time with the actual data. We use the same values for $PProb_{n,p}(t)$ ($p = 1, \cdots, Q-1$). For this, we set the final probability $FProb$ for processes to proceed to the final level. $PProb_{n,p}(t) = FProb^{1/N}$. We set the value of $FProb$ by 0.788 and this is the same as the average probability to proceed to the next quality for the actual data used in the previous subsection. The interval is 0.4 and this is also the same as the interval for the actual data. The result is shown in **Fig. 7**. Similar to the actual data, the transaction times under the PQI method are shorter than those under the conventional method in many cases. In the cases that the number of the streams is less than 3 under both methods, the transaction time saturates. Therefore, in the remaining subsections, we use the average transaction time for the cases that the transaction time saturates as the index of the performance.

## 5.4 Influence of Number of Streams

The data amount that the processing computer receives increases as the number of streams increase. Thus, the average transaction time is influenced by the number of streams. We investigate the influence.

**Figure 8** shows the result of the average transaction time changing with the number of the streams. The horizontal axis is the number of the streams and the vertical axis is the average transaction time. We simulate the transaction time under different final probabilities. The final probabilities are 0.001, 0.01, 0.1, 0.3, 0.5, 0.7, and 1.0. The intervals for all data streams are the same and is 0.4 [sec.]. The number of qualities under our proposed method is 5.

A larger final probability causes a longer transaction time since the processing computer collects and analyzes a larger amount of data to proceed to a higher quality. The average transaction time under the No PQI method increases as the number of the streams increases for the cases where it is less than 3, because the data amount that the processing computer receives increases. For the cases where the number of the streams is larger than 2, the average transaction time increases sharply. This is because the transaction time increases as the time proceeds as shown in the case of 3 streams in Fig. 7 and the system fails to process stream data continuously. We can see similar phenomena for the cases of our proposed method. However, the maximum number of the streams that the system works is larger compared with the No PQI method. For example, in case where the final probability is 0.1, the average transaction time sharply increases when the number of the streams is 4. Therefore, the processing computer can collect data from more data sources by using our proposed PQI

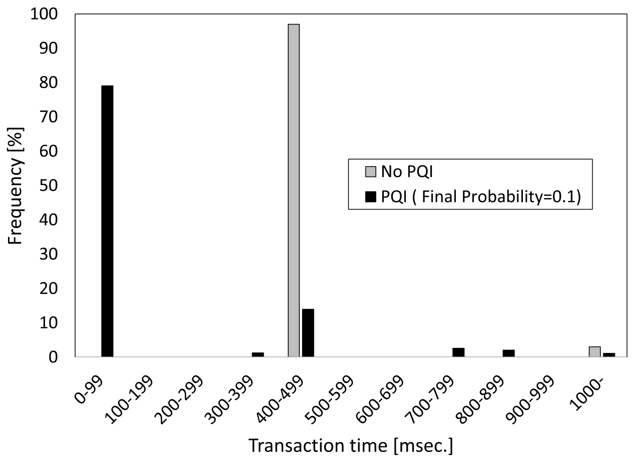

**Fig. 9** Distribution of transaction time when the number of streams is 3.

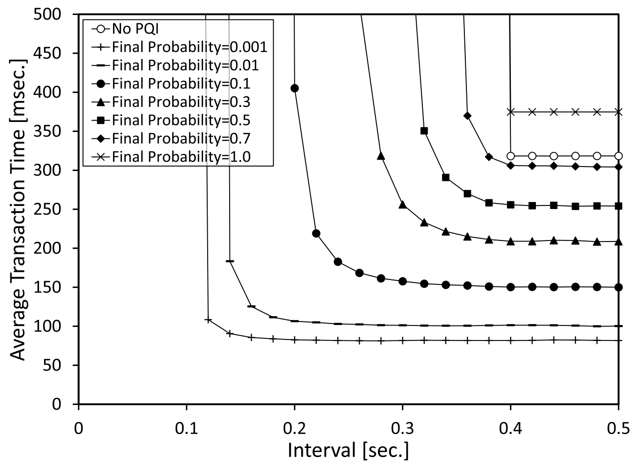

**Fig. 10** Average transaction time under different intervals.

method compared with the conventional No PQI method.

The average transaction time in the cases of the maximum number of the streams just before the system fails under the PQI method is longer than that under the No PQI method except for the case where the final probability is 0.001. This is because the processing computer receives a higher quality data after requesting it to data sources in the PQI method. For example, in case where the final probability is 0.1, the average transaction time is 447 [msec.] for 4 streams under the PQI method though it is 318 [msec.] for 2 streams under the No PQI method. This is a demerit of the PQI method.

Not only the average transaction time, we investigate the distribution of transaction time from over 700 simulations. **Figure 9** shows the distribution of transaction time in the cases of "No PQI" and "PQI (Final Probability=0.1)" when the number of streams is 3. We picked the result up as an example, and the result shows that our proposed method keeps the transaction time under 100 [msec.] for nearly 80% cases in this simulation environment. In addition, our implemented simulator has a small percentage result whose transaction time is over 1,000 [msec.].

## 5.5 Influence of Intervals

The processing computer frequently receives data as the interval shortens and the average processing time increases. Therefore, we investigate the influence of the intervals.

The result of the evaluation is shown in **Fig. 10**. The horizontal axis presents the interval values and the vertical axis presents the average of transaction time. In this evaluation, the number of streams is 2. For our proposed method, the number of the qualities is 5.

Similar to the previous evaluation result, the system fails when the interval is excessively short and the average transaction time sharply increases. In the No PQI method, the average transaction time is constant when the interval is larger than 0.4 [sec.]. This is because the transaction time sharply increases in the cases where the processing computer receives the next data during processing in the No PQI method. In the PQI method, the shortest interval that the system works is shorter compared with the No PQI method. For example, in the cases where the final probability is 0.1, the average transaction time sharply increases when the interval is 0.15. Therefore, the processing computer can collects data with a shorter interval by using our proposed PQI method compared with the conventional No PQI method.

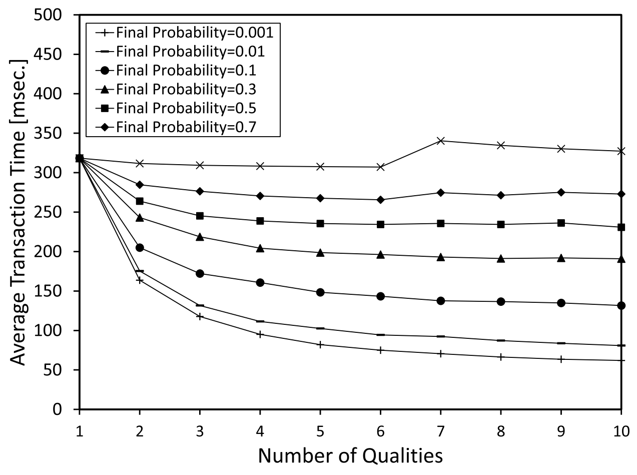

**Fig. 11** Average transaction time user different number of qualities.

## 5.6 Influence of Number of Qualities

The number of qualities influences the average transaction time. We investigate the influence.

**Figure 11** shows the average transaction time under different numbers of the qualities. The horizontal axis presents the numbers of the qualities and the vertical axis presents average transaction time. The intervals value is 0.4 [sec.] and the number of the streams is 2.

The average transaction time for the case where the number of the qualities is 1 represents the average transaction time under the No PQI method. The other qualities 2 to 10 are the average transaction times under the PQI method. The average transaction time decreases as the number of the qualities increases except for the case where the number of qualities is 7. When the number of levels is 7, the data reception overlaps with that of the next transaction and the computational resources of the processing computer intercept. Therefore, the average transaction time is long.

## 5.7 Influence of Number of Final Probabilities

As shown in the previous evaluation results, the average transaction time depends on the final probability. We show the trans-

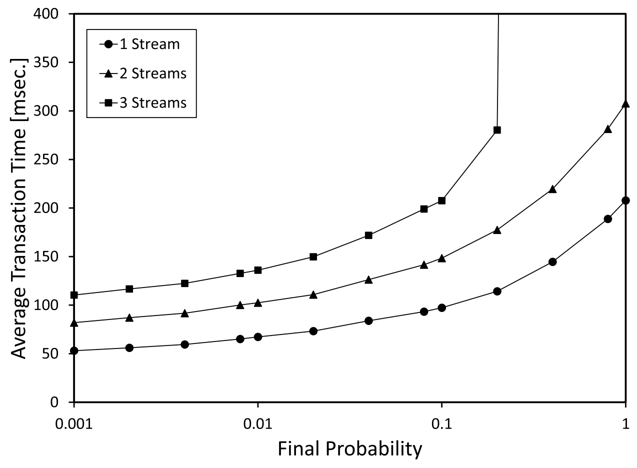

**Fig. 12** Average transaction time under different final probabilities.

action time changing the final probability.

**Figure 12** shows the result of average transaction time under different final probabilities. The horizontal axis presents the final probability with a logarithmic scale. The vertical axis presents the transaction time. To keep the consistency with the previous results, we set the number of qualities to 5 and the interval to 0.4 [sec.]

The transaction time increases as the final probability increases since more processes proceed to analyses of higher quality data and the transaction time increases. When the number of streams is 3, the transaction time sharply increases when the final probability is larger than 0.2. This is because, as explained in the previous subsections, the transaction time increases as the time proceeds and the system fails to process stream data continuously.

## 6. Conclusion

Transaction time is one of the main factors to improve performances of some IoT applications. To reduce the transaction time, we proposed an efficient data collection method using a progressive quality improvement approach. In our proposed method, only for cases where higher quality data are needed for analyses, processing computers progressively collect them by reducing the average data amount of collections and for analyses, our proposed method reduces the average transaction time. Our simulation evaluation revealed that our proposed method can reduce the transaction time while keeping the application performances compared with a conventional No PQI method.

In the future, we plan to propose a method for the situation where there are multiple processing computers. In addition, we will consider parallel processing of collected data. If smartphones with multi-core become popular in the future, it is possible to send streams using these devices. The extension of our method considering power consumption is interesting and a new challenge.

**Acknowledgments** This work was supported in part by JSPS KAKENHI Grant Numbers JP15H02702, JP17K00146, and JP18K11316. Also this work include the result of NICT Osaka University joint research "research and development of advanced network platform technology for large scale distributed computing", and of I-O DATA Foundation.

## References

[1] Xu, J., Andrepoulos, Y., Xiao, Y. and van der Schaar, M.: Nonstationary resource allocation policies for delay-constrained video streaming: Application to video over internet-of-things-enabled networks, *IEEE Journal on Selected Areas in Communications*, Vol.32, pp.782–794 (Apr. 2014).

[2] Molina-Giraldo, S., Insuasti-Ceballos, H.D., Arroyave, C.E., Montoya, J.F., Lopez-Villa, J.S., Alvarez-Meza, A. and Castellanos-Dominguez, G.: People detection in video streams using background subtraction and spatial-based scene modeling, *2015 20th Symposium on Signal Processing, Images and Computer Vision* (*STSIVA*), pp.1–6 (Sep. 2015).

[3] Rao, S. and Dakshayini, M.: Priority based optimal resource reservation mechanism in constrained networks for iot applications, *2016 International Conference on Wireless Communications, Signal Processing and Networking* (*WiSPNET*), pp.1228–1233 (Mar. 2016).

[4] Akbar, A., Kousiouris, G., Pervaiz, H., Sancho, J., Ta-Shma, P., Carrez, F. and Moessner, K.: Real-time probabilistic data fusion for large-scale iot applications, *IEEE Access*, Vol.6, pp.10015–10027 (Feb. 2018).

[5] Colmenares, J.A., Dorrigiv, R. and Waddington, D.G.: A single-node datastore for high-velocity multidimensional sensor data, *2017 IEEE International Conference on Big Data* (*Big Data*), pp.445–452 (Dec. 2017).

[6] Beard, J.C. and Chamberlain, R.D.: Analysis of a simple approach to modeling performance for streaming data applications, *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pp.345–349 (Aug. 2013).

[7] Zhao, P., Yu, W., Yang, X., Meng, D. and Wang, L.: Buffer data-driven adaptation of mobile video streaming over heterogeneous wireless networks, *IEEE Internet of Things Journal*, Vol.5, No.5, pp.3430–3441 (Oct. 2018).

[8] Buddhika, T. and Pallickara, S.: Neptune: Real time stream processing for internet of things and sensing environments, *2016 IEEE International Parallel and Distributed Processing Symposium* (*IPDPS*), pp.1143–1152 (May 2016).

[9] Sun, D. and Hwang, S.: DSSP: Stream Split Processing Model for High Correctness of Out-of-Order Data Processing, *2018 IEEE 1st International Conference on Artificial Intelligence and Knowledge Engineering* (*AIKE*), pp.193–197 (Nov. 2018).

[10] Ramachandra, M.: Optimization of the data transactions and computations in IoT sensors, *2016 International Conference on Internet of Things and Applications* (*IOTA*), pp.358–363 (Sep. 2016).

[11] Alieksieiev, V.: One Approach of Approximation for Incoming Data Stream in IoT Based Monitoring System, *2018 IEEE Second International Conference on Data Stream Mining & Processing* (*DSMP*), pp.94–97 (Oct. 2018).

[12] Agrawal, U.A. and Jani, P.V.: Performance analysis of real time object tracking system based on compressive sensing, *2017 4th International Conference on Signal Processing, Computing and Control* (*ISPCC*), pp.187–193 (Sep. 2017).

[13] Lu, C.H., Yu, C.H., Chen, B.H., Hwang, I.S. and Huang, S.S.: Semi-supervised data stream analytics with balanced recognition performance and processing speed, *2017 IEEE International Conference on Consumer Electronics - Taiwan* (*ICCE-TW*), pp.355–356 (June 2017).

[14] Ge, Y., Liang, X., Zhou, Y.C., Pan, Z., Zhao, G.T. and Zheng, Y.L.: Adaptive analytic service for real-time internet of things applications, *2016 IEEE International Conference on Web Services* (*ICWS*), pp.484–491 (June 2016).

[15] Image dataset for human detections "pedestrians" (2012), available from ⟨http://jacarini.dinf.usherbrooke.ca/static/dataset/baseline/pedestrians.zip⟩.

[16] Open CV library, available from ⟨https://opencv.org/⟩.

### Editor's Recommendation

This paper achieves to efficiently collect stream data such as video data or sensor data for real-time IoT applications. Since the line capacity limits the real-time performance and analysis frequency, the proposed method controls the data quality according to the application requirements. The paper gives insights to readers in this research field and thus is selected as a recommended paper.

(Chief examiner of SIGDPS Atsushi Tagami)

**Chaxiong Yukonhiatou** received his B.Eng., M.Eng. from National University of Laos (Laos), King Mongkut's Institute of Technology Ladkrabang (Thailand), in 2008 and 2014, respectively. Since 2009, he have been a lecturer for Department of Computer Engineering, Faculty of Engineering, National University of Laos (NUOL). He is currently pursuing his Ph.D. degree with the Graduate School of Information Science and Technology, Osaka University, Japan. His main research interests include stream data processing and IoT systems.

**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 a research associate at Kyoto University. In Jan. 2008, he joined the Cybermedia Center, Osaka University as an assistant professor and in Mar. 2009, he became an associate professor. From Apr. 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. He is a member of the IPSJ, IEICE, and IEEE.

**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 Mar. 2013 and from July 2014 to Mar. 2015, he was a specially appointed researcher at Osaka University. From Apr. 2013 to June 2014, he was a Ph.D. researcher at Kobe University. Since Apr. 2015, he has been a assistant professor at Nara Institute of Science and Technology. His research interests include distributed computing, rule-based systems, and stream data processing. He is a member of the IPSJ and IEEE.

**Yuuichi Teranishi** received his M.E. and Ph.D. degrees from Osaka University, Japan, in 1995 and 2004, respectively. From 1995 to 2004, he was engaged Nippon Telegraph and Telephone Corporation (NTT). From 2005 to 2007, he was a Lecturer of Cybermedia Center, Osaka University. From 2007 to 2011, he was an associate professor of Graduate School ofInformation Science and Technology, Osaka University. Since Aug. 2011, He has been a research manager and project manager of National Institute of Information and Communications Technology (NICT). He received IPSJ Best Paper Award in 2011. His research interests include technologies for distributed network systems and applications. He is a member of the IPSJ, IEICE, and IEEE.

**Shinji Shimojo** received his M.E. and Ph.D. degrees from Osaka University in 1983 and 1986, respectively. He was an Assistant Professor with the Department of Information and Computer Sciences, Faculty of Engineering Science at Osaka University from 1986, and an Associate Professor with Computation Center from 1991 to 1998. During this period, he also worked for a year as a Visiting Researcher at the University of California, Irvine. He has been a Professor with the Cybermedia Center (then the Computation Center) at Osaka University since 1998, and from 2005 to 2008 had been the director of the Center. He was an executive researcher and a director of Network Testbed Research and Development Promotion Center at National Institute of Information and Communications Technology from 2008 to 2011. He is currently a director of Cybermedia Center. His current research work is focusing on a wide variety of multimedia applications, peer-to-peer communication networks, ubiquitous network systems, and IoT systems. He is a founding member of PRAGMA and CENTRA. He was awarded the Osaka Science Prize in 2005. He was awarded by Minister of Internal affair on 2017. He is a member of IEEE, and IEICE and IPSJ fellow.