A fast stream transaction system for real-time IoT applications

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

A fast stream transaction system for real-time IoT applications

Chaxiong Yukonhiatou[a], Tomoki Yoshihisa[b,*], Tomoya Kawakami[c], Yuuichi Teranishi[b,d], Shinji Shimojo[b]

[a] *Graduate School of Information Science and Technology, Osaka University, Ibaraki, Osaka 5670047, Japan*
[b] *Cybermedia Center, Osaka University, Ibaraki, Osaka 5670047, Japan*
[c] *Nara Institute of Science and Technology, Ikoma, Nara 6300192, Japan*
[d] *National Institute of Information and Communications Technology, Koganei, Tokyo 1848795, Japan*

A R T I C L E   I N F O

Article history:
Received 15 January 2020
Revised 17 February 2020
Accepted 18 February 2020
Available online 26 February 2020

Keywords:
IoT
Stream data processing
Progressive quality improvement
Dynamic interval

A B S T R A C T

Due to the recent prevalence of IoT (Internet of Things) technologies, various IoT devices connect to the Internet and continuously send their generated data to remote processing computers such as video data or sensor data. The transaction rate is one of the main factors to improve the performance of some IoT applications. For instance, in surveillance systems, the probability to catch a thief increases as the processing computer analyzes the video with a higher transaction rate. To improve the transaction rate, some methods reduce the transaction time between a processing computer and stream data sources under a static transaction interval. However, the transaction rate can be further improved by changing the transaction interval dynamically depending on the transaction time. In this paper, we propose a method to improve the transaction rate by changing the transaction interval dynamically. In our proposed method, a processing computer sometimes changes the transaction interval to be the same length as the average transaction time. Moreover, our proposed method adopts a progressive quality improvement (PQI) approach to reduce the transaction time. We measured the transaction rate of our proposed method by both a simulator and an implemented system. We confirmed that our proposed method can improve the transaction rate by 4.4 times and the transaction time by 21% at least compared with the conventional method. Moreover, we confirmed that the average frame rate increases 22% compared with a simple method in a real situation.

1. Introduction

Nowadays, various IoT devices such as cameras or sensors connect to the Internet. They are generally small size and their processing powers are also low. Therefore, in most IoT applications, stream data generated by these devices are transmitted to remote processing computers. The processing computers process stream data continuously and get various useful results. For example, in surveillance systems, a processing computer receives video data continuously from surveillance cameras and analyzes image data of each video frame to identify recorded people.

In IoT applications, a higher stream transaction rate leads a more frequent data analyses and enables performance improvements. In the above example, the number of the people identified increases as the transaction rate increases since they

* Corresponding author.
E-mail addresses: chaxiong@fe-nuol.edu.la (C. Yukonhiatou), yoshihisa@cmc.osaka-u.ac.jp (T. Yoshihisa), kawakami@is.naist.jp (T. Kawakami), teranisi@cmc.osaka-u.ac.jp (Y. Teranishi), shimojo@cmc.osaka-u.ac.jp (S. Shimojo).

https://doi.org/10.1016/j.iot.2020.100182
2542-6605/© 2020 Elsevier B.V. All rights reserved.

are moving and the probabilities to record them in the video increase. Here, a transaction includes the data collection and the analysis. Generally, the transaction rate is determined so as not to overlap transactions because a transaction time (from the start of a data collection to the finish of the analysis) lengthens when the transaction overlaps with the previous or the next transaction. Therefore, to improve the transaction rate, some methods reduce the transaction time [1–4]. A shorter transaction time enables a shorter transaction interval of stream data and thus the stream transaction rate increases. They target periodic transactions and assume static transaction interval.

However, the transaction time dynamically changes depending on the communication time and the processing time of each transaction. Indeed, these times differ for each transaction in some existing methods [5–7]. Although a shorter transaction time enables a shorter transaction interval, the conventional methods assume static interval. Therefore, the transaction rate can be further improved by changing the transaction interval dynamically depending on these times. It is very difficult to determine the transaction interval so as to further improve the transaction rate because these times depend on the contents of the stream data. As explained above, the transaction rate decreases in cases that the transaction interval is too short and the transactions overlap. On the other hand, a longer transaction interval also decreases the transaction rate because the frequency that the data sources transmit the stream data decreases. It is required to determine the transaction interval to efficiently improve the transaction rate.

In this paper, we propose a method to improve the transaction rate by changing the transaction interval dynamically. In our proposed method, a processing computer sometimes changes the transaction interval to be the same length as the average transaction time. Frequent changes of transaction interval cause the inconsistency of the times to get data. To keep the consistency as possible, in our proposed method, the processing computer changes the interval every a fixed number of transactions. Moreover, our proposed method adopts a progressive quality improvement (PQI) approach to reduce the transaction time. In the approach, the data amount of transmitted data is reduced by transmitting the data that have necessary quality.

The main contributions of this paper are summarized as follows:

- A dynamic interval method to improve the transaction rate under the PQI approach.
- An adoption of the PQI approach.
- A performance evaluation of our proposed method by a simulator and a real situation.

Edge computing, Fog computing, and MEC Servers get attractions to realize the real time processing of IoT streams. Our proposed method in this paper achieves a faster stream transaction compared with a previously proposed methods and can contribute to realize low latency, fast round trip time systems under these paradigms.

The remainder of this paper is organized as follows. In Section 2, we introduce some related work. In Section 3, we will explain our assumed system environments. Our proposed method is explained in Section 4, Our implementation is explained in Section 5 and the performance evaluation results are shown in Section 6. Finally, we will conclude the paper in Section 7.

2. Related Work

Some methods to improve the transaction rate for IoT applications have been proposed.

To reduce the communication time, many schemes have been proposed ([8–11]). These schemes degrade qualities of data such as resolutions for image data, to reduce data to be collected and achieve a shorter transaction time. These methods result in performance degradations of IoT applications. In our proposed method, applications can improve their performances by changing the transaction interval dynamically.

A method to reduce communication time by controlling the number of the data packets for transactions was proposed in [12]. A method to control communication buffer to keep the transaction rate was proposed in [13]. These methods are similar with our proposed method that data is divided into some parts, but our method progressively collects the divided data considering their necessities.

A method to reduce communication traffic between a video stream data source and a processing computer was proposed in [14]. A method to reduce communication traffic by compressing data was proposed in [15]. Different from these methods, in our proposed method, the transaction rate is improved while the qualities of stream data progressively improve.

A model to reduce the delay for starting data processing were proposed in [16]. In the model, the processing computer prepares separated data queues for each process and selects the data to process so as to reduce the processing delays. In our proposed method, we can adopt this method in the processing computer. Our proposed method is different from this in the point that we improve the transaction rate by managing how to process data.

Some methods to control data generation timings to reduce communication traffic were proposed. The method proposed in [17] considers communication distance between the data sources and the processing computer. The method proposed in [18] considers the communication channels such as wireless or wired.

In [19], a method to improve the transaction rate has been proposed. In this method, the processing computer dynamically changes the transaction interval depending on the transaction time. The results showed that the transaction rate can improve under this method. However, the performance in practical situations is not investigated and the overheads caused by actual systems are unclear. In this paper, we further investigate the experimental results in a real situation.

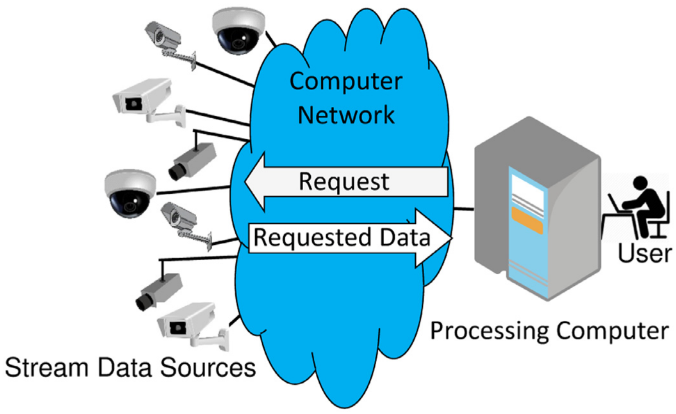

Fig. 1. Our assumed system architecture.

3. Assumed System

3.1. System Architecture

Fig. 1 shows our assumed system architecture. Some IoT devices such as surveillance cameras continuously get data about their observations such as video data and act as stream data sources. They can generate some data items which have qualities from their observed data. These IoT devices connect to a computer network such as the Internet and communicate with a processing computer. The processing computer executes designated processes every data reception from the stream data sources. Such a type of processes is called stream processing. The users designate the processes for the stream data to a processing computer. The processing computer has a buffer for storing received data and executes processes for the data.

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

Suppose an area in that some surveillance cameras are deployed and a processing computer gathers their recorded video 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 person identification system by a face recognition. For this, the application designates the process that notifies to the user when the processing computer identifies person face and determines whether the person is registered or not in the video data got from surveillance cameras. To detect faces, the user submits the face images of registered persons 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 registered persons or not. If the processing computer detects the faces that are not registered, it sends a notification to the user by e-mail or other messaging services.

3.3. Research Objective

In the scenario introduced in Section 3.2, a main application performance is the probability to catch thieves. This can increase by analyzing video data with a higher transaction rate. It is better to give a consistency to data transaction interval for transactions. For example, in the above scenario, a higher and consistent transaction rate, i.e., a more frequent and periodic data gathering from the cameras, increases the chance to identify people since they are moving and the probabilities to record them in the video increase.

Conventional methods improve the transaction rate by reducing communication time between a processing computer and stream data sources. They target periodic stream processing and assume static transaction interval. However, the transaction time dynamically change depending on the amount of transmitted data and the number of data sources. The transaction rate can be further improved by changing the transaction interval dynamically depending on transactions time. Therefore, our research objective is improving the transaction rate by changing the transaction interval dynamically and implement a real system using our proposed method.

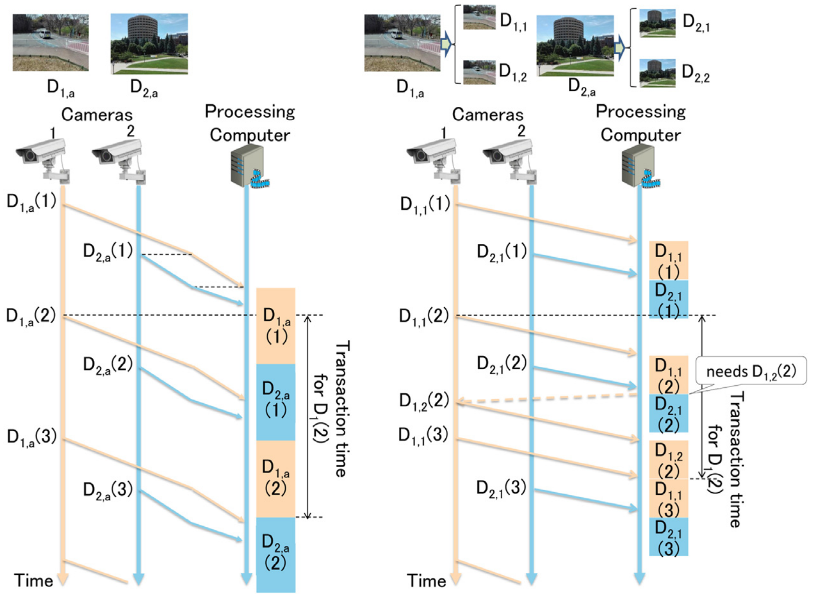

Fig. 2. Stream data collection of the conventional method and of the PQI-CDI method.

3.4. *Difficulty of the Problem*

A short transaction interval can give a higher transaction rate for real-time IoT applications. However, an excessively short transaction interval causes higher loads on both communication and processing since the processing computers receive the next transaction before finished processing the current transaction. Thus, the transaction rate decrease. In other hand, a transaction interval is longer than transaction time has a possibility to further improve the transaction rate. Therefore, in this paper, we proposed a method to change the transaction interval dynamically depending on the transaction time.

4. Proposed Method

In this section, we explain our proposed method. It called the PQI-CDI (Progressive Quality Improvement approach with Cycle-based Dynamic Interval) method. In this method, the system changes the transaction interval dynamically and adopts the PQI approach to reduce the transaction time.

4.1. *PQI approach*

Generally, data have some qualities, e.g., resolution of image data. Data analyses can be applied for each quality and data with the highest quality often gives the best performance for analyses. If 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, in the above scenario, 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 are small, the processing computer skips the analyses of higher quality image data since new humans do not appear in the frame because of small difference. In cases that the probability to proceed to higher quality data analyses is small, the total amount of received data is reduced, compared with the case that all quality data are received. Therefore, the data amount to be received is reduced when the probability is small compared with the processing computer receives the highest quality data without consideration of data qualities. Thus, the transaction time is reduced keeping the application performance. We call this approach *progressive quality improvement* approach.

Fig. 2 shows a timing chart for stream processing under the conventional method and the PQI-CDI method. In the PQI-CDI method, the data $D_{d,a}(t)$ $(d = 1, 2,\ t = 1, \cdots)$ is divided into some qualities. Each transaction includes some processes for each divided data. In the figure, the number of the qualities is 2 and the transaction consists of two processes for the divided data $D_{d,1}(t)$ and $D_{d,2(t)}$. d is the stream number, t is the cycles for data collections, and q is the quality. In the cycle 1, the transaction finishes at the first quality in both streams. In the cycle 2, the processing computer requires $D_{1,2}(2)$ when it finishes the process for $D_{1,1}(2)$. The camera 1 transmits the required $D_{1,2}(2)$ and the processing computer starts the process for $D_{1,2}(2)$. In this case, the transaction finishes when the processing computer finishes the process for $D_{1,2}(2)$ since the number of the qualities are 2. The transaction time in this case is reduced compared with that under the conventional approach as shown in the figure.

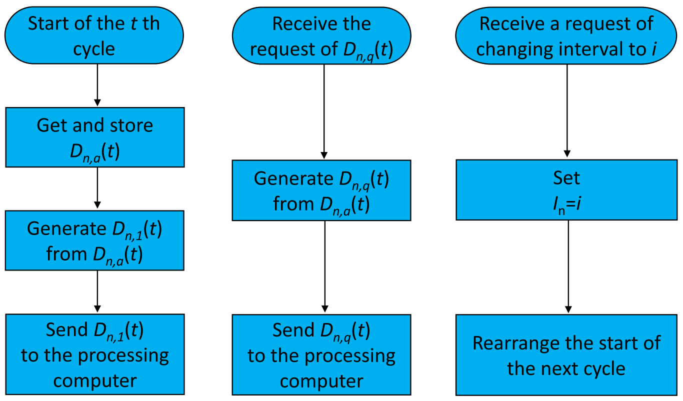
Start of the t th cycle
Get and store $D_{n,a}(t)$
Generate $D_{n,1}(t)$ from $D_{n,a}(t)$
Send $D_{n,1}(t)$ to the processing computer
Receive the request of $D_{n,q}(t)$
Generate $D_{n,q}(t)$ from $D_{n,a}(t)$
Send $D_{n,q}(t)$ to the processing computer
Receive a request of changing interval to i
Set I_n=i
Rearrange the start of the next cycle

Fig. 3. Flow chart of data sources.

4.2. Cycle-based Dynamic Interval

For changing the intervals, the PQI-CDI method determines when and how long the processing computer changes the transaction interval.

4.2.1. Timings to Change Intervals

A more frequent change of intervals causes a less consistency. On the other hand, a less change of intervals cannot improve the transaction rate further since it takes a longer time to change intervals based on the communication time and the processing time. However, it is difficult to find the appropriate timing to change intervals depending on the communication time and processing time since these times change dynamically.

One of the solution for such a dynamic situation is fixed period. Therefore, in the PQI-CDI method, the processing computer changes the intervals of each stream every finishing C_n transactions with the data source n.

4.2.2. How to Determine Intervals

A longer interval causes a less transaction rate. On the other hand, a shorter interval than a transaction time causes a longer transaction time because the processing computer receives the next data before it finishes the current process. Therefore, the interval that is the same length as the transaction time is the most appropriate value. However, it is difficult to grasp the transaction time since it depends on the communication time and the transaction time.

The PQI-CDI method has a cycle to change intervals C_n. Therefore, we adopt the average transaction time for the previous cycle as the new interval.

4.2.3. Algorithms

Fig. 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 temporary. Here, $D_{n,a}(t)$ is the observed original data of the data source n at the cycle t. $D_{n,q}(t)$ $(q = 1, \cdots, Q)$ is the generated data from $D_{n,a}(t)$ of that quality is q. First, they generate $D_{n,1}(t)$ from $D_{n,a}(t)$ and send $D_{n,1}(t)$ to the processing computer. 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. When the data source n receives the request of changing interval to i, it changes its interval to i and rearranges the start of the next cycle.

Fig. 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 process of t th cycle finishes. Otherwise, the processing computer judges the necessity of $D_{n,q+1}(t)$. In case that $D_{n,q+1}(t)$ is needed for the process execution, the processing computer requests $D_{n,q}(t)$ to the data source n, otherwise, the process finishes. When the process finishes, in the PQI-CDI method, the processing computer checks whether c_n reaches C_n or not. c_n is the variable to count the number of transactions for the data source n. Here, again, C_n is the interval of transactions to change the transaction interval of the data source n. In case that c_n reaches to C_n, the processing computer calculates the new interval using the average transaction time for the previous cycles. That is:

$$AveTT_n(t) = \sum_{\tau=t-C_n+1}^{C_n} TT_n(\tau) \tag{1}$$

Then it sends the request for changing the interval to the data source n. Then, initialize c_n. Here, $TT_n(t)$ is the transaction time of the t th cycle of the data source n, i.e., the time to get the original data at the data source n to the time to finish the process of the data at the processing computer.

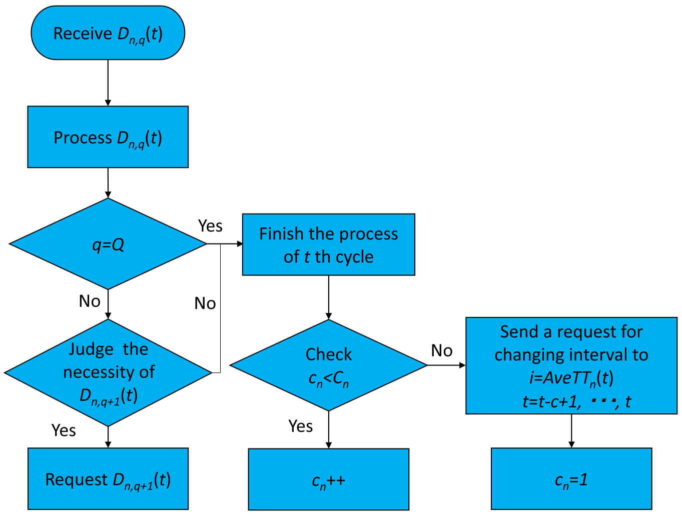
Receive $D_{n,q}(t)$
Process $D_{n,q}(t)$
$q=Q$
Yes
No
No
Finish the process of t th cycle
Judge the necessity of $D_{n,q+1}(t)$
Yes
Request $D_{n,q+1}(t)$
Check $c_n<C_n$
No
Yes
c_n++
Send a request for changing interval to $i=AveTT_n(t)$ $t=t-c+1, \cdots, t$
$c_n=1$

Fig. 4. Flow chart of processing computer.

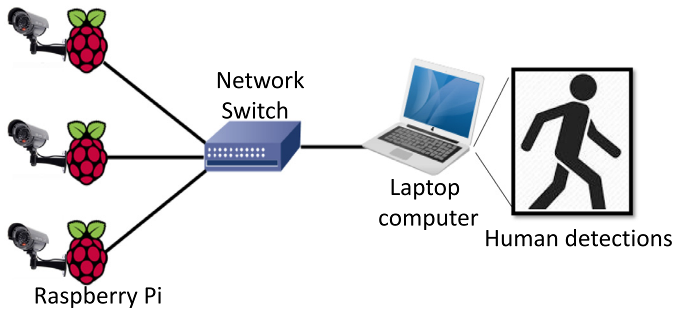

Fig. 5. System architecture of our implemented system.

The complexity (in terms of number of operations needed for convergence) of the proposed algorithm cannot be analyzed mathematically because the transaction time fluctuates according to the quality levels that the processing computer processes and it is not controllable by the system. Therefore, the transaction time does not converge and the transaction interval does not converge. Accordingly, our proposed method changes the transaction interval cyclically. In the case that the transaction time is happen to be constant, the transaction interval converges in only one cycle.

5. Implementation

In this section, we explain our implementation of the PQI-CDI method.

5.1. System Architecture

Fig. 5 shows the system architecture of our implemented system. The data sources are the cameras connected to the Raspberry Pi 3 a Laptop computer working as a processing computer via 100BASE-TX/1000BASE-T network (Allied Telesis CentreCOM GS908GT switch). We used the Python programming language to implement the human detection software. Each camera gets image data with 640 × 480 resolution and encodes into progressive JPG format, which contains 10 different qualities (called scans in progressive JPEG). Example images are show in the Fig. 6. These generated qualities are temporarily stored in the memory. At the time to start a transaction, each data source first sends the lowest quality data to the processing computer. When the processing computer receives the data, it tries to detect humans in the received image. If the processing computer detects human bodies in the firstly received scan, the processing computer requests to the cameras to get the remaining scans (the higher quality image data) and progressively collects them. Otherwise, the processing computer skips collecting the higher scans. Table 1 shows the specifications of our implemented system.

5.2. Communication Flow

Fig. 7 shows the communication flow of our implemented system. We use two channels for the communications. The communication channel is used for requesting higher quality data from the processing computer and for sending them to

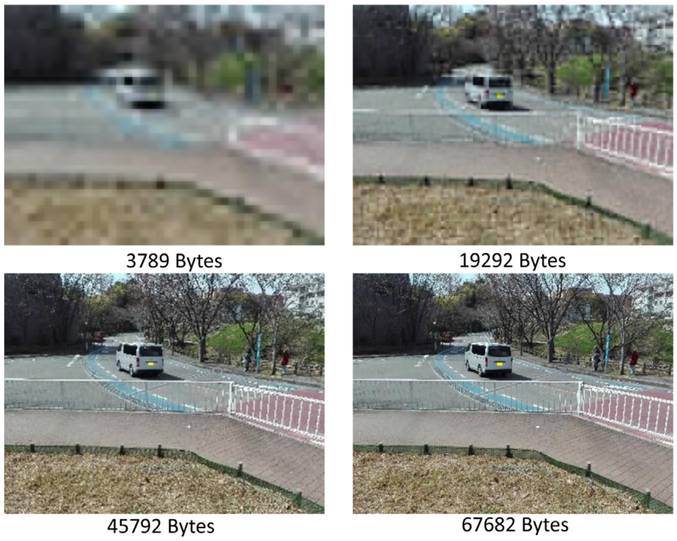

Fig. 6. Images with different qualities.

Table 1
Specifications of our implemented system.

Items	Details
Recording computer	Raspberry Pi 3 Model B (1.2GHz quad-core ARM Cortex-A53, 1GB memory, 100BASE-TX, Raspbian Ver 10. Sep. 2019)
Camera device	Raspberry Pi Camera Module V2
Processing computer (Windows 10 Pro 64bit)	Dell Latitude E7240 (2.10GHz dual-core Intel Core i7-4600U, 8GB memory, Intel HD Graphs 4600, 1000BASE-T
Network	Allied Telesis CentreCOM GS908GT (100BASE-TX/1000BASE-T), CAT5e cables
Cables	CAT5e, 1m length
Num. of recording computers	3
Num. of processing computers	1
Num. of qualities in prog. JPEG	10
Comparison methods	The PQI-CDI method and the PQI method
Evaluation items	Average Frame Rate [FPS]

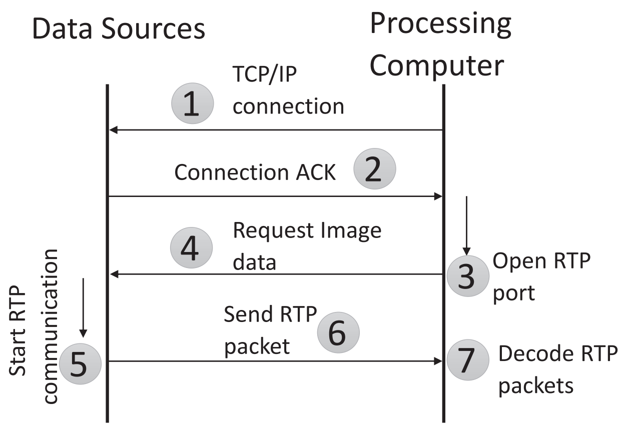

Fig. 7. The communication diagram for our implemented system.

it. For this, we use TCP/IP protocol. The data transfer channel is used for transferring the first quality data from the data sources to the processing computer. We use RTP for the data transfer channel. In our system, the processing computer first connects to the data sources via the communication channel. The data sources make connections to the processing computer in the second step. If the connection complete, the processing computer opens an RTP channel and waits for receiving RTP packets in the third step. If the connection confirmation is established, the processing computer requests the start of stream

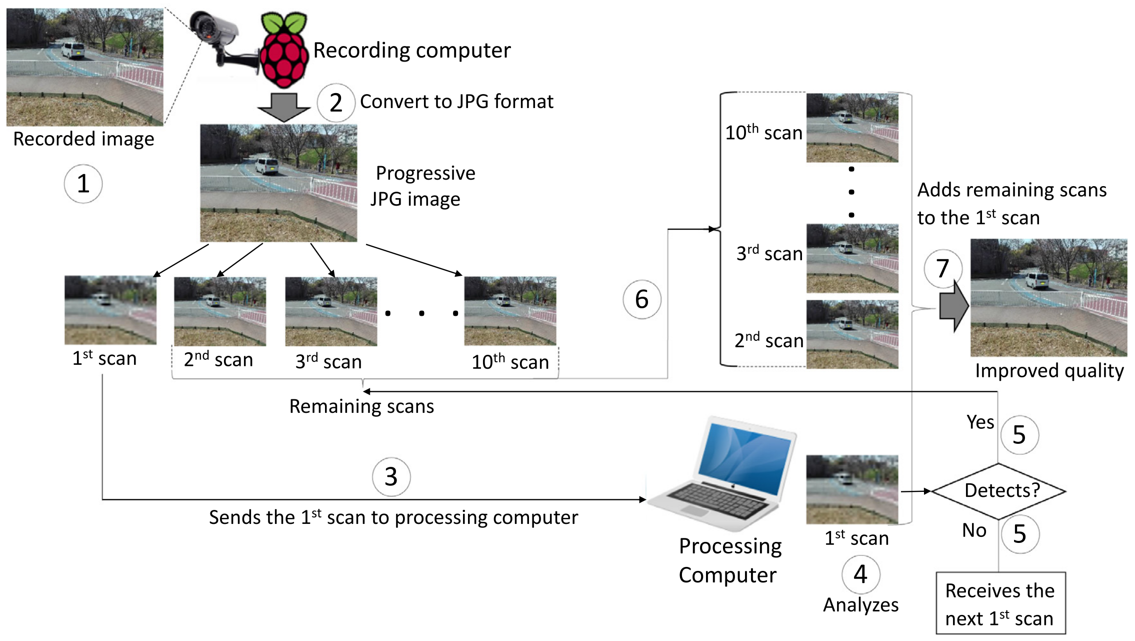

Fig. 8. The communication diagram how to generate and request higher quality data.

data transactions in the fourth step. When the data sources receive the requests for starting transactions, they get image frames from their camera's sensor and generates them into 10 qualities (10 scans) for frame. The data sources encode the lowest quality (first scan) data into RTP packets and send them to the processing computer via the data channel in the step sixth. Finally, the processing computer receives RTP packet and decodes the image data for analysis. In the case that a higher quality image data is needed, the processing computer sends the requests to the data sources via the communication channel.

5.3. How to Generate and Request Higher quality data

In this section, we explain how to generate different qualities data and requesting the higher quality data for each frame. Fig. 8 shows how to generate some qualities data for each frame and requesting them. First, the data source gets a raw image data from its camera's sensor. Second, the data source encodes the data into progressive JPG format with 10 qualities (called scans in progressive JPG) using OpenCV (a popular programming library for the computer vision field) and temporarily stores them in its buffer. Third, the data source sends the first scan (first quality) to the processing computer. Fourth, when the processing computer receives the first scan, it checks whether a human is detected or not in the image data. Fifth, in the case that a human is detected in the first scan, the processing computer requests to the data source in order to get the remaining qualities and progressively collect them in the step sixth and seventh, respectively. Finally, the processing computer combines the remaining scans with the first scan in order to get the improve quality image. In case there is not a human detected in the first scan in step fifth, the processing computer waits for receiving for the next frame (the first scan of the next image data).

6. Evaluation

In this section, we first show the simulation results to confirm the effectiveness of our proposed system. After that, we show the performance overheads in our developed real system. We show the simulation results in this subsection.

6.1. Results by Simulator

6.1.1. Evaluation Parameters

In this evaluation, we assume the application explained in Subsection 3.2 and use the parameters shown in Table 2.

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. Total data amount is the data amount of $D_{n,a}(t)$ $(n = 1, \cdots, N,$ $t = 1, \cdots, T)$. To make the evaluation results easily understandable, we set the same data amount for all data items. Processing Time Ratio is the value of the transaction time divided by the amount of the data item for the process. We set

Table 2
Simulation settings.

Input Bandwidth	10 [Mbps]
Output Bandwidth	10 [Mbps]
Total Data Amount	12.5 [Kbytes]
Processing Time Ratio	10^{-6}

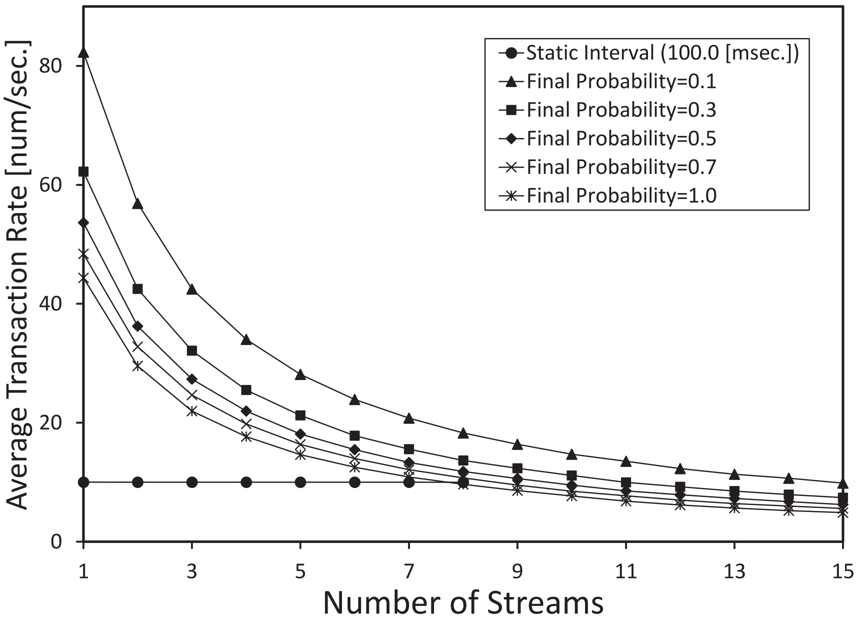

Fig. 9. Average transaction rates under different final probabilities.

these parameters considering practical situations. We use the same values for $PProb_{n,p}(t)$ $(p = 1, \cdots, Q-1)$. $PProb_{n,p}(t)$ is the probability to the next quality from the quality p. For this, we set the final probability $FProb$ for processes to proceed to the final quality. $PProb_{n,p}(t) = FProb^{1/N}$. We simulate the stream processing system for 60 seconds.

6.1.2. Evaluation Items

The main evaluation item is the transaction rate. The transaction rates are the number of the transactions that the processing computer finishes in a second. One of the other performance for stream processing is the transaction time. We calculate the average values since we confirmed that the transaction time converge. One of the demerits of dynamic interval is the inconsistency of the intervals. To investigate this, we calculate the fairnesses of the intervals. We adopt the Jain's coefficient for the fairness. A smaller fairness indicates a more inconsistent interval.

6.1.3. Influence of Number of Streams

We measure the performances changing the number of the streams. In this experiment, the number of the transactions that the processing computer changes the interval $C_n (n = 1, \cdots, N)$ is 2 since this value gives a higher transaction rate and a higher fairness as shown in the next subsection. N is the number of the streams. We set the number of the qualities to 5 as an example value. The initial interval is 100 [msec.].

Fig. 9 shows the average transaction rate. The horizontal axis is the number of the streams and the vertical axis is the average transaction rate. Our proposed PQI-CDI method gives a higher average transaction rate than that for the case of static interval (100 [msec.]) when the number of the streams is less than 9. This is because the network and the processing computer have an extra capacity to improve the transaction rate compared when the intervals are 100 [msec.] and the PQI-CDI method exploits this extra capacity by changing the interval dynamically. The average transaction rate increases as the final probability increases since the average amount of the data to be transmitted from the data sources decreases as the final probability decreases and the extra capacity increases. The line for the static interval stops at the point that the number of the streams is 8 since the transaction time diverge in the cases where the number of the streams is larger than 8. For example, when the number of streams is 1, the average transaction rate in the case that the final probability is 1.0 under our PQI-CDI method is 44 [num/sec.] although this under the static interval is 10 [num/sec.]. Therefore, our proposed method can improve the average transaction rate by 4.4 times even in the case that the final probability is 1.0. This is because the PQI-CDI method changes the transaction interval dynamically.

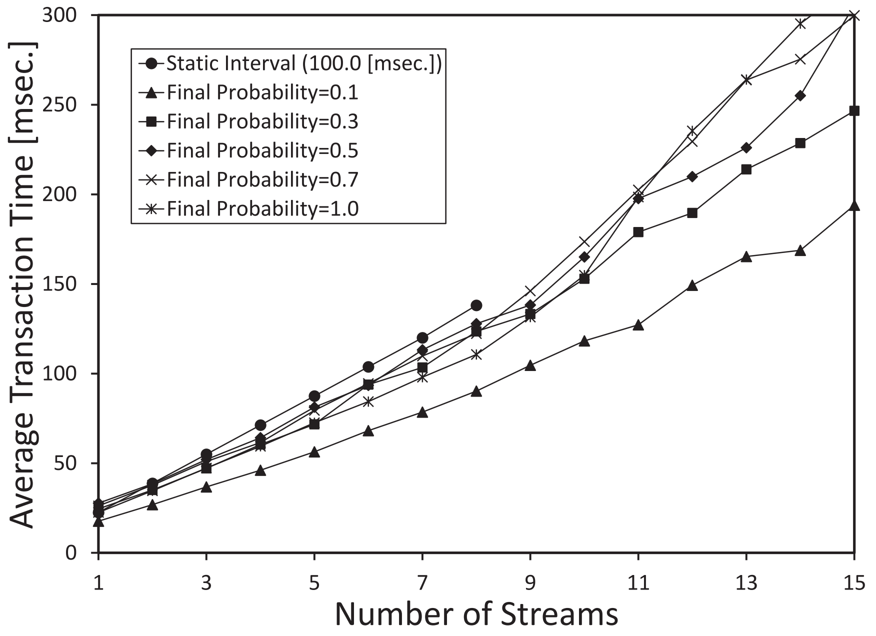

Fig. 10. Average transaction time under different final probabilities.

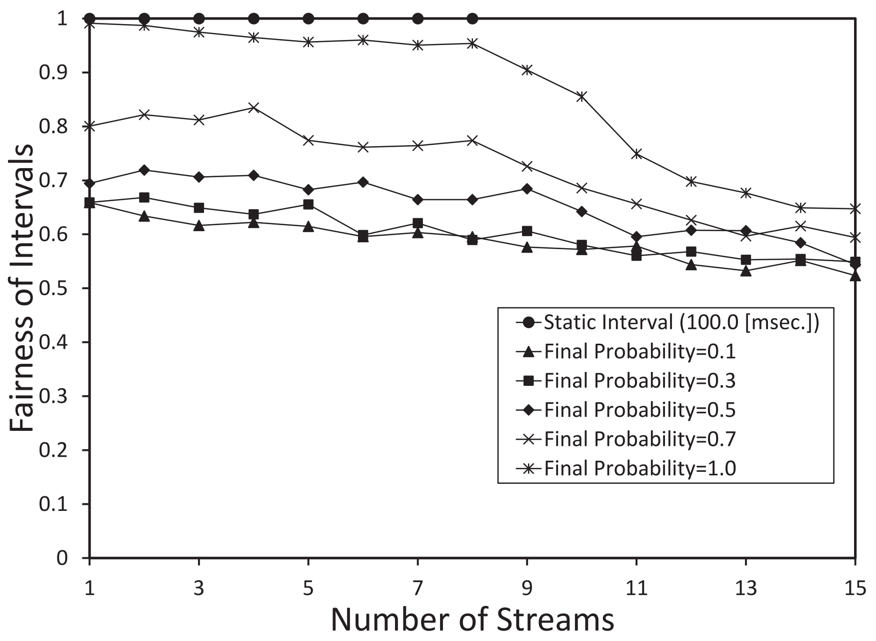

Fig. 11. Fairness of intervals under different final probabilities.

Fig. 10 shows the average transaction time. The horizontal axis is the number of the streams and the vertical axis is the average transaction time. Our proposed PQI-CDI method gives a shorter average transaction time than that for the case of static interval (100 [msec.]). This is because the network and the processing computer have an extra capacity to improve the transaction time compared when the intervals are 100 [msec.] as the same reason as the average transaction rate. For example, when the number of streams is 8, the average transaction time in the case that the final probability is 1.0 under our PQI-CDI method is 110 [msec.] although this under the static interval is 138 [msec.]. Therefore, our proposed method can improve the average transaction rate by 21% even in the case that the final probability is 1.0.

Fig. 11 shows the fairness of the intervals. Our proposed PQI-CDI method gives a lower fairness than that for the case of static interval (100 [msec.]) since the intervals dynamically change under the PQI-CDI method and the intervals have various

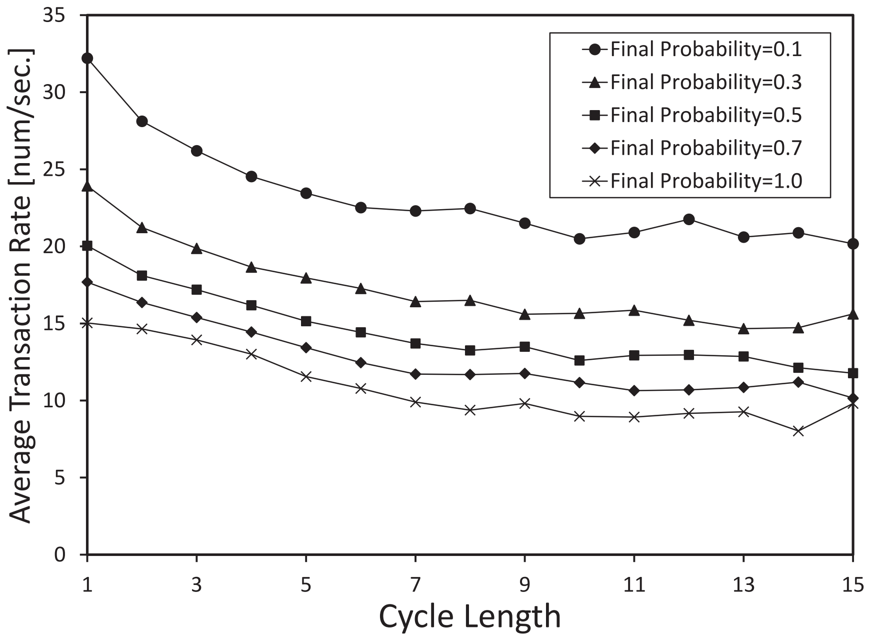

Fig. 12. Average transaction rates under different final probabilities changing cycle length.

values. The fairness decreases slightly as the number of the streams increases since the range of the intervals increases as the number of the streams increases.

6.1.4. *Influence of Cycles*

In our PQI-CDI method, the cycles for changing intervals influence the performances. We investigate the influence changing the cycle length $C_n(n = 1, \cdots, N)$ under different final probabilities. The cycle lengths for all streams are the same. The final probabilities are 0.1, 0.3, 0.5, 0.7, 1.0. The initial interval is 100 [msec.] and the number of the qualities is 5 as an example value.

Fig. 12 shows the average transaction rate. The horizontal axis is the number of the cycle length and the vertical axis is the average transaction rate. We can see that a shorter cycle length causes a higher average transaction rate since the interval is adjusted to the transaction time frequently by changing the interval. Therefore, the processing computer can collection more data by changing the transaction interval dynamically under the PQI-CDI method than that under the conventional method.

Fig. 13 shows the average transaction time. The horizontal axis is the number of the cycle length and the vertical axis is the average transaction time. In the PQI-CDI method, the average transaction time for a higher final probability gives a longer average transaction time since a higher final probability enables more transactions to proceed to the final quality. We can see some average transaction time with a lower final probability gives a shorter average transaction time.

Fig. 14 shows the fairness of the intervals. The horizontal axis is the number of cycle length and the vertical axis is the fairness of intervals values. The fairness decreases as the cycle length increases. This is because under our proposed method the intervals dynamically change.

6.2. *Results by Real System*

We show the results of our developed real system explained in Section 5 by using parameters shown in Table 3

6.2.1. *Frame Rate*

the transaction rate can be actually calculated by the frame rate and transaction time. influence the transaction rate. Therefore, we measure them using our developed system. In this section, we call the method of that transaction interval is static under the PQI approach, the PQI method.

Fig. 15 shows the average frame rate. The horizontal axis is the number of cameras with different methods (PQI and PQI-CDI) and the vertical axis is the average frame rate. We can see that the average frame rate under the PQI-CDI method gives a higher average frame rate than the PQI method since the PQI-CDI method can change the transaction interval to a shorter transaction interval than the static interval. In this experiment, the communication bandwidth and the processing power of the processing computer are sufficient and the average frame rate does not decrease even when the number of

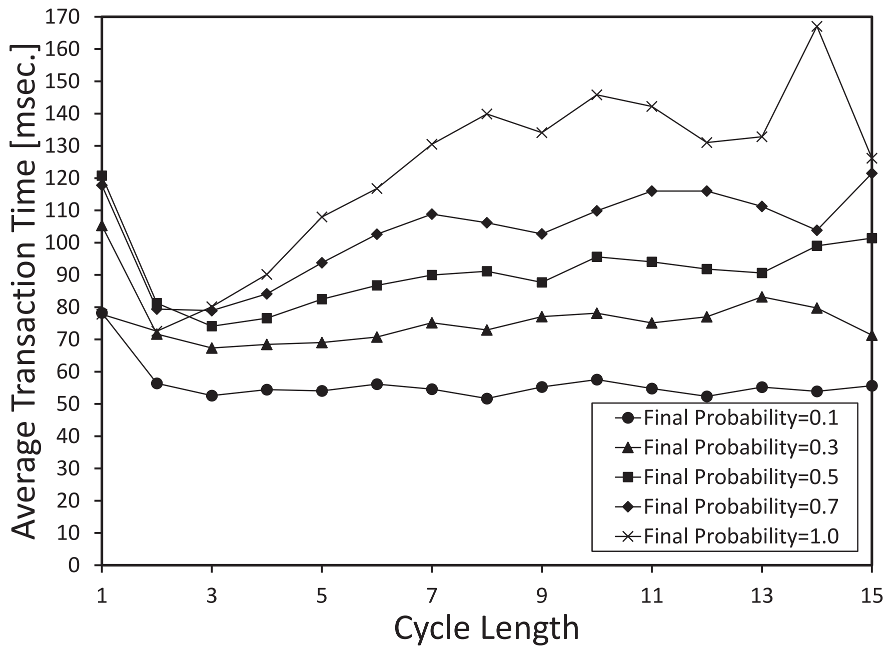

Fig. 13. Average transaction time under different final probabilities changing cycle length.

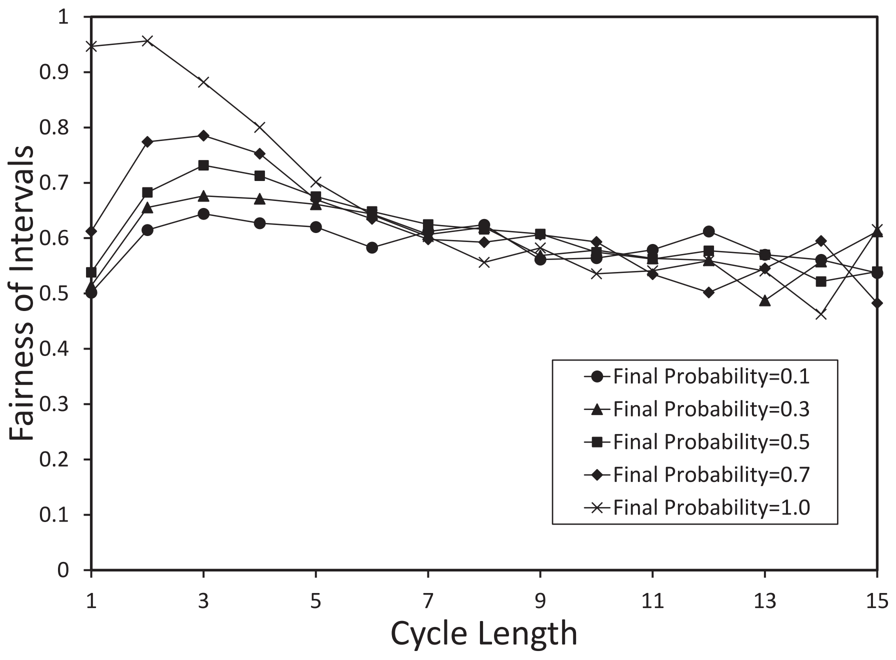

Fig. 14. Fairness of intervals under different final probabilities changing cycle length.

the cameras is 3. It is obvious that the average frame rate decreases when the number of the cameras further increases. For example, when the number of cameras is 3, the average frame rate under our PQI-CDI method is 10.8 [fps] although this under the original PQI method is 8.86 [fps]. Therefore, our proposed method can improve the frame rate by 22%.

6.2.2. Transaction Time

Fig. 16 shows the transaction time of the PQI and the PQI-CDI methods under 1 camera. The horizontal axis is the transaction IDs and the vertical axis is the transaction time. We can see that the transaction time are almost constant at

Table 3
Parameter values for the implemented system.

Communication Bandwidth	Approx.100 [Mbps]
Communication Protocol	RTP over UDP
3 Raspberry Pi devices	live camera
Laptop	1
Image Resolution	640 × 480
Progressive JPG	10 scans
Initial Interval	0.03
Cycle length	20
Image Analysis	upper human body detection (HAAR)

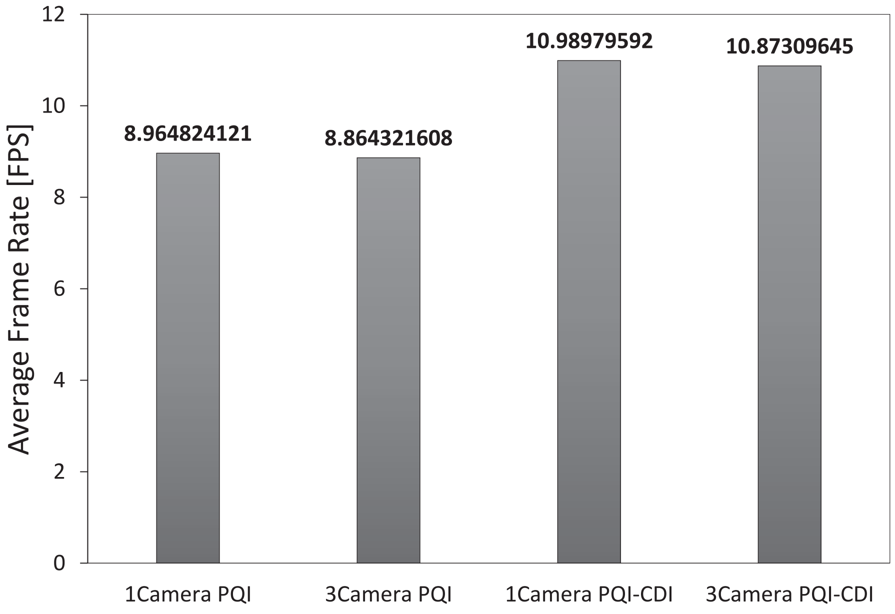

Fig. 15. Average frame rate.

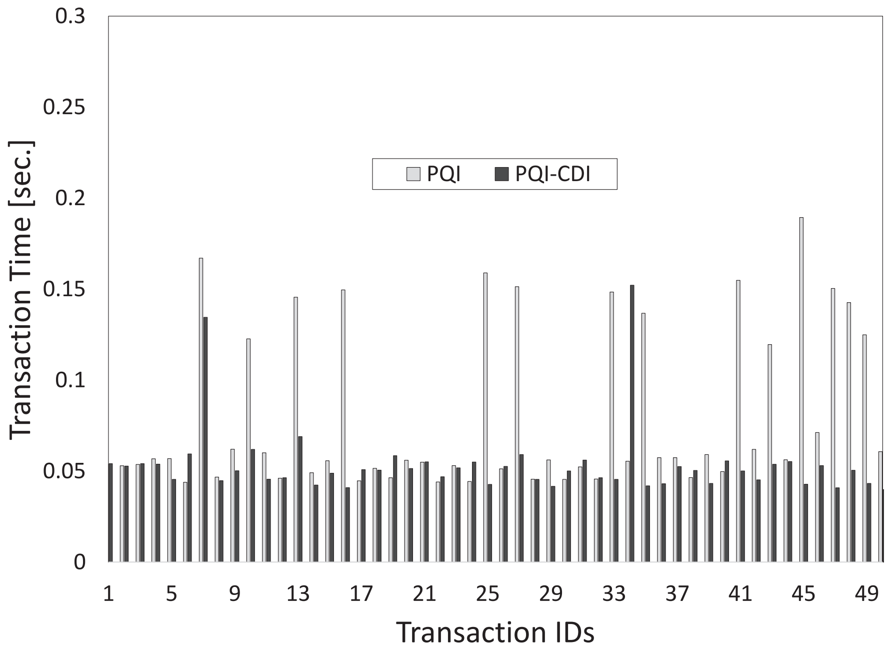

Fig. 16. Transaction time (1 camera).

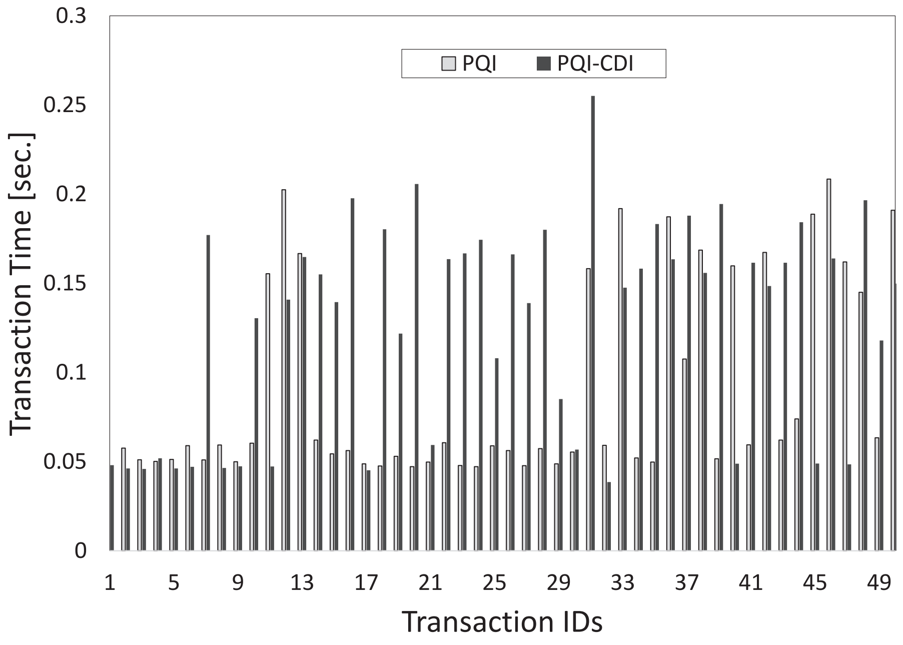

Fig. 17. Transaction time (3 cameras).

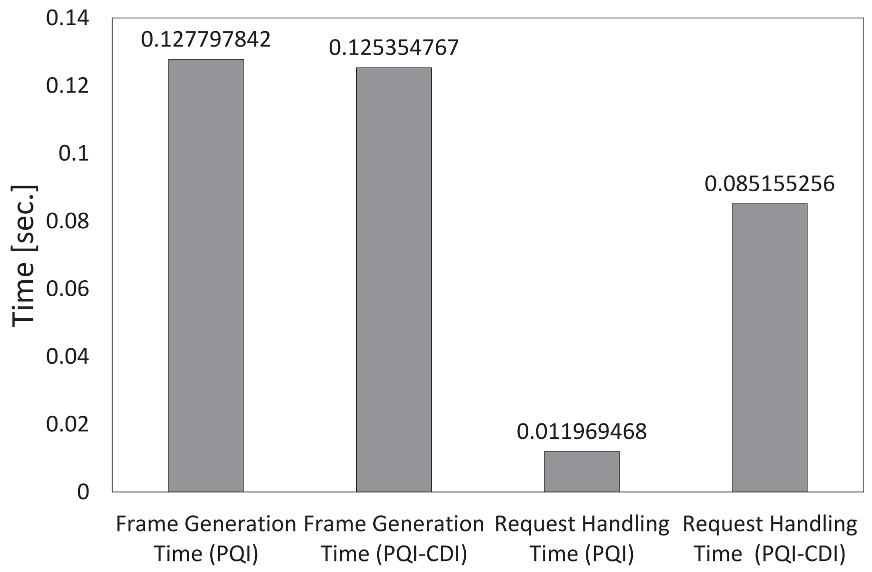

Fig. 18. Overheads.

0.05[sec.] for both the PQI and the PQI-CDI methods when the processing computer does not request the high quality data since the transmitted data amount is small. In the both methods, the transaction time increases in the case when the high quality data is needed. For example, the transaction time for the transaction ID 7 is longer than the transaction time for transaction ID 6. This is because it takes time to communicate the high quality data.

Fig. 17 shows the transaction time under 3 cameras compared with the case of 1 camera, the transaction time for getting high quality data is slightly longer. This is because the number of transmitted data amounts influence the communication bandwidth and processing computer's capacities. Hence, the transaction time gets longer.

6.2.3. Difference from Simulator

Our developed simulator does not consider the frame generation time and the request handling time. In the PQI method, each camera takes some times to generate the data with some different qualities and handle the requests as explained in Section 5.3. Hence, we check the overhead.

Fig. 18 shows the overhead in the cameras. The horizontal axis are the frame generation time on the camera side and the request handling time for the PQI and the PQI-CDI methods. The vertical axis is the average time for them per transaction. We can see that the frame generation time under the PQI-CDI and the PQI methods are almost the same. This is because the number qualities data generated are the same. The request handling time is the time to receive the requests of processing computer to the time starting frame generation. We can see that the request handling time under the PQI-CDI method is larger than the PQI method since the request handling time of the PQI-CDI method includes both the requests for the high quality data and also them for changing the intervals.

7. Conclusion

Stream transaction rate is one of the main factors to improve the performance of some IoT applications. In this paper, we proposed the PQI-CDI method in order to improve the transaction rate. In the PQI-CDI method, the processing computer changes the transaction interval to be the same length as the average transaction time every a fixed number of transactions. Moreover, the PQI-CDI method adopts the PQI approach to reduce the transaction time. Our evaluation results revealed that the PQI-CDI method can achieve a higher stream transaction rate than a conventional method with the PQI approach. Moreover, we compared the results by the simulation with that of our developed real system and found that the PQI-CDI method encounters some overheads for frame generation and request handling in each data source.

In the future, we plan to introduce machine learning technique to predict appropriate transaction interval and to investigate the effectiveness for real-time IoT applications.

Declaration of Competing Interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Acknowledgment

This work was supported in part by JSPS KAKENHI Grant Numbers 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".

References

[1] S. Gavrilov, G. Ivanova, D. Ryzhova, P. Volobuev, Multi-interval static timing analysis accounting logic compatibility, in: IEEE East-West Design & Test Symposium (EWDTS), Yerevan, 2016, pp. 1–4.
[2] M. Dyvak, A. Pucas, Identification of the static system interval models by application of optimal localization experiment, in: International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics, Slavske, Ukraine, 2003, pp. 180–183.
[3] M. Li, W. Tang, M. Yuan, Reliability-based topology optimization of interval parameters structures with dynamic response constraints, in: IEEE International Conference on Mechatronics and Automation, Tianjin, 2014, pp. 469–474.
[4] S. Siddiqui, A.A. Khan, S. Ghani, Investigating dynamic polling intervals for wireless sensor network applications with bursty traffic, in: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Daegu, 2017, pp. 448–451.
[5] T. Ito, H. Noguchi, Y. Yamato, T. Murase, Transaction offloading for access management to live data of iot in information-centric network, in: IEEE 7th Global Conference on Consumer Electronics (GCCE), Nara, 2018, pp. 287–288.
[6] M. Ramachandra, Optimization of the data transactions and computations in iot sensors, in: International Conference on Internet of Things and Applications (IOTA), Pune, 2016, pp. 358–363.
[7] M. Abe, G. Yamamoto, S. Furuichi, (WIP) iot context descriptor: Situation detection and action invocation model for real-time high-volume transactions, in: IEEE International Congress on Internet of Things (ICIOT), San Francisco, CA, 2018, pp. 130–133.
[8] A. Ortega, M. Khansari, Rate control for video coding over variable bit rate channels with applications to wireless transmission, in: International Conference on Image Processing, Washington, DC, USA, volume 3, 1995, pp. 388–391.
[9] O.D. Incel, B. Krishnamachari, Enhancing the data collection rate of tree-based aggregation in wireless sensor networks, in: Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, San Francisco, CA, 2008, pp. 569–577.
[10] F. Xhafa, V. Naranjo, S. CaballĘ, L. Barolli, A software chain approach to big data stream processing and analytics, in: International Conference on Complex, Intelligent, and Software Intensive Systems, Blumenau, 2015, pp. 179–186.
[11] G.Z. Papadopoulos, N. Pappas, A. Gallais, T. Noel, V. Angelakis, Distributed adaptive scheme for reliable data collection in fault tolerant WSNs, in: World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 116–121.
[12] S. Yu, T. Tian, J. Zhou, H. Guo, An adaptive packet transmission model for real-time embedded network streaming server, in: International Conference on Audio, Language and Image Processing, Shanghai, 2008, pp. 848–853.
[13] P. Zhao, W. Yu, X. Yang, D. Meng, L. Wang, Buffer data-driven adaptation of mobile video streaming over heterogeneous wireless networks, in IEEE Internet of Things Journal 5 (5) (2018) 3430–3441.
[14] H. Kanzaki, K. Schubert, N. Bambos, Video streaming schemes for industrial iot, in: International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, 2017, pp. 1–7.
[15] U.A. Agrawal, P.V. Jani, Performance analysis of real time object tracking system based on compressive sensing, in: International Conference on Signal Processing, Computing and Control (ISPCC), Solan, 2017, pp. 187–193.
[16] J.C. Beard, R.D. Chamberlain, Analysis of a simple approach to modeling performance for streaming data applications, in: International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, 2013, pp. 345–349.

[17] X. Zhu, P. Huang, S. Han, A.K. Mok, D. Chen, M. Nixon, Minmax: A sampling interval control algorithm for process control systems, in: IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Seoul, 2012, pp. 68–77.
[18] G.M. Dias, M. Nurchis, B. Bellalta, Adapting sampling interval of sensor networks using on-line reinforcement learning, in: IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, 2016, pp. 460–465.
[19] C. Yukonhiatou, T. Yoshihisa, T. Kawakami, Y. Teranishi, S. Shimojo, A scheme to improve stream transaction rate for real-time iot applications, in: International Conference on Advanced Information Networking and Applications (AINA, Matsue, Japan, March), 2019, pp. 787–798.