A Study on Stream Data Processing with Dynamic Quality Control

Submitted to

Graduate School of Information Science and Technology

Osaka University

2022/1/11

Chaxiong Yukonhiatou

List of Publications

1. Journal Paper

1. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, "A Method to Reduce Transaction Time for Real-time IoT Applications," in IPSJ Journal of Information Processing (JIP), pp.701-710, November 2019 [Recommended paper].

2. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, "A fast stream transaction system for real-time IoT applications," in Elsevier Internet of Things Journal, Volume 11, pp. 701–710, September 2020 [Recommended paper].

2. International Conference Paper (with review)

1. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, "A Scheme to Improve Stream Transaction Rates for Real-time IoT Applications," in Proceedings of the International Conference on Advanced Information Networking and Applications (AINA), pp 787-798, March 2019 (Best Paper Award) [acceptance rate: 25%].

2. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, "A Dynamic Intervals Determination Method Based on Transaction Rates for Real-Time IoT Applications," in Proceedings of the IEEE International Workshop on Architecture, Design, Deployment and Management of Networks and Applications (ADMNET) in Conjunction with the 43rd Annual International Computer, Software and Applications Conference (COMPSAC), pp. 7-12, July 2019.

3. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, “A Performance Evaluation of Object Detections by Progressive Quality Improvement Approach,” in Proceedings of the IEEE Global Conference on Consumer Electronics (GCCE), pp 321-325, October 2019.

3. Domestic Conference Paper

1. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, “A Scheme to Improve Stream Data Analysis Frequency for Real-time IoT Applications,” in Proceedings of the IPSJ Multimedia, Distributed, Cooperative and Mobile Symposium (DICOMO), pp. 1205-1211, July 2018 [excellent paper].

2. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, “An Implementation of Video Surveillance Systems with Progressive Quality Improvement Approach,” in Proceedings of the IPSJ Multimedia, Distributed, Cooperative and Mobile Symposium (DICOMO), pp. 979-984, July 2019.

3. Chaxiong Yukonhiatou, Tomoki Yoshihisa, Tomoya Kawakami, Yuuichi Teranishi and Shinji Shimojo, “An Implementation of Surveillance Systems with Dynamic Transaction Intervals under PQI Approach,” in Proceeding of the IPSJ Research Report on Multimedia Communication and Distributed Processing (2019-DPS-181), pp. 1-6, December 2019.

Summary

Recently, stream data processing systems such as Apache Flink and Spark Streaming have attracted considerable research attention. Stream data processing systems are widely used in various applications that require periodical and continuous data processing such as object detection and sensor data analysis. In the present stream data processing applications, sensor devices with network access capability are often applied as data sources. The computational power of such device is smaller than that of ordinary desktop computers. Therefore, to enable complex and heavy load stream data processing such as object detection or state analysis, sensor devices transmit stream data to remote processing computers with higher computation power to offload the data processes. The processing computer periodically processes each data item included in the stream data. For instance, several video surveillance systems execute stream data processes on a remote server to detect perpetrators in videos captured by surveillance cameras. Two key indexes can be considered to evaluate the performance of such stream data processing systems, namely, transaction time and transaction rate. The transaction time corresponds to the period between the instant at which a data item is produced on a stream data source to the instant at which the processing computer completes processing it. The transaction rate means the number of transactions completed per unit time (a second). A smaller transaction time and a higher transaction rate correspond to enhanced application performance. For instance, the probability of apprehending perpetrators increases when the processing computer analyzes the videos within a smaller transaction time and with a higher transaction rate. To enable performance enhancements, most methods reduce the communication traffic between the data sources and the processing computer.

Although stream data processes do not always require the original qualities of the data got from the sensor devices at the data sources, the processing computer in conventional systems receives original quality data. For example, in certain cases, the processing computer in face detection systems can recognize faces even when the image size is

smaller than the original size (the quality of data in this case pertains to the resolution). In certain human detection tasks, the processing computer can detect target humans in a part of the image area (the quality in this case pertains to the image region). In the SLAM (Simultaneous Localization and Mapping) systems for autonomous robots, original quality LiDAR (Light Detection And Ranging) data, including the detailed object shape, are not needed depending on the situation (e.g. detailed object shape is not needed to change the movement direction to avoid obstacles). By evaluating the necessity of original quality data, we can reduce the redundant communication traffic in certain situations, thereby enhancing the performance of stream data processing systems. Therefore, this study aims to establish a novel progressive quality improvement (PQI) approach to enhance the abovementioned performance indexes. In this approach, data with different qualities are generated from the original data. The data with the lowest quality are cyclically sent to the processing computer. Only in cases when data with higher qualities are needed, the processing computer progressively collects these data from the data sources. Moreover, we developed a method to improve the transaction rate by changing the transaction interval dynamically under the PQI approach. To investigate whether the PQI approach can enhance the performance in real situations, the PQI strategy was implemented in a video surveillance system to investigate the differences in the results obtained using the implemented system and developed simulator. This dissertation is divided into five chapters.

In Chapter 1, we provide a background overview, the objectives, related work, and the content of this research.

Chapter 2 describes the proposed PQI approach for stream data processing. The objective is to reduce the transaction time. Each stream data source generates certain data having different qualities from the original data. The data are transmitted only if requested by the processing computer. The quality of data for stream processing progressively improves. The PQI approach is evaluated through our developed simulator. The results show that the transaction time of the PQI approach is shorter than that of the conventional approach when the probability that the process proceeds to the original quality data is small.

Chapter 3 describes the method to improve the transaction rate under the PQI approach, termed the cycle-based dynamic interval (PQI-CDI) method. The transaction interval is dynamically changed based on the transaction time. The processing computer changes

the transaction intervals of each stream data every time when a predetermined number of transactions are finished. We evaluate the proposed method by using our developed simulator. The results show that the proposed method can enhance the transaction rate compared with that of the conventional static interval method.

Chapter 4 explains the implementation of a video surveillance system incorporated with the proposed PQI-CDI method. We used three camera devices and one processing computer for the implementation and developed two software for them. We investigate the differences between the performances obtained by our developed simulator and those by our implemented system. The experimental results show that the PQI-CDI method can improve the transaction time and the transaction rate in actual situations although the performances were not always similar to the simulation results.

Chapter 5 presents the concluding remarks and mentions several future research directions.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Recently, stream data processing systems such as Apache Flink[1], Spark Streaming[2], and Kafka Streams[3] have attracted considerable research attention. In this study, *stream data* is defined as a series of data items that is periodically and continuously generated from a variety of data sources such as camera devices and sensor devices. Representative examples of stream data include image data, sensor data, and social media feeds.

Stream data processing systems are widely used for various applications such as detecting objects in video data for real-time event detections (e.g., perpetrator detection [1–5], license plate detection [6–9], car detection [10–12], etc.), identifying behaviors of the users for health cares by the vibration sensor data analyses [13, 14], extracting disaster information from social media feeds [15, 16].

Considering the prevalence of object detection systems implementing stream data processing, object detection systems are regarded as a typical application of stream data processing in this study.

In the present stream data processing applications, sensor devices with network access capability are often applied as data sources. For example, compact computers like Raspberry Pi devices or other micro-computers are often equipped as stream data sources because their spatial sizes are small, and thus they are easily installable at various places. The computational power of such a device is smaller than that of ordinary desktop

[1] https://flink.apache.org/
[2] https://spark.apache.org/
[3] https://kafka.apache.org/

computers. Therefore, to enable complex and heavy-loaded stream data processing, sensor devices transmit stream data to remote processing computers with a higher computational power to offload the data processes. The processing computer periodically processes each data item included in the stream data. Such stream data processing systems in which a processing computer processes stream data transmitted by remote stream data sources are the target systems of this study. A set of processes (including communication processes) for each data item is termed as a *transaction*. A time span in that a transaction is running is termed as a *transaction time*. A transaction starts from the instant at which a data item is constructed on a stream data source to the instant at which the processes for the data item finish on the processing computer. A new transaction starts as soon as the *transaction interval* elapses from the instant at which a transaction starts. The transaction interval is a different value from the transaction time. If the transaction time is longer than the transaction interval, the transaction overlaps with the next one.

Figure 1.1 shows a representative our assumed system. In the figure, the data sources are camera devices deployed at an office. The example application is a video surveillance system. In this system, the image data captured by surveillance camera devices are transmitted to a processing computer in a security center [17–20] because the computational resources of the camera devices are often insufficient to accomplish object detection tasks. In this application, a transaction is a set of processes for each video frame data. The transaction interval is the time span between the transmissions of frame image data and the value equals to the inverse value of the frame rate on the camera devices (1/30 [s] for 30 [Hz] video).

Two key indexes can be considered to evaluate the performance of stream data processing systems.

(1) *Transaction time*

As we briefly explained in the above, a transaction time is the time required for processing a transaction and is the period between the instant at which a stream data source constructs a data item to the instant at which the processing computer completes the set of processes for it. The transaction time includes the communication time and the processing time for the transaction. A shorter transaction time enables stream data processing systems to get the processing results of a transaction earlier. In video surveillance systems, the transaction time corresponds to the delay from the time of a real incident to its detection time on the system. If the delay is long, countermeasures

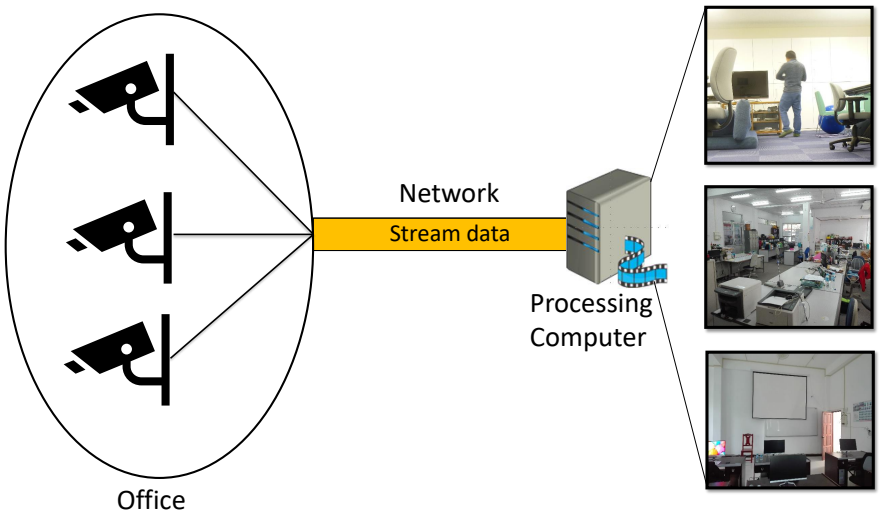


Figure 1.1: A representative our assumed system

against incidents may become invalid. For example, if the alarm is delayed due to the long transaction time, the perpetrator might be able to escape from the surveillance point. Therefore, the transaction time is a key factor to enhance the performance of stream data processing systems of which input data come from remote stream data sources.

(2) *Transaction rate*

The number of transactions completed per second is referred to as the *transaction rate*. The transaction rate is the inverse value of transaction interval if there is no resource competition in the communication network nor the processing computer. When the transaction is executed on competitive resources and the transaction does not finish within the transaction interval, the transaction rate becomes smaller than the inverse value of the transaction interval. In this case, the transaction time eventually gets longer and the transaction rate becomes smaller. Only in cases where the transaction time equals the transaction interval value, the transaction rate is the inverse value of the transaction time. A higher transaction rate enables stream data processing systems to get more frequent processing results of the transactions. In video surveillance systems, to keep the detection performance for moving objects, the transaction rate should be higher than a certain value to obtain enough number of detection process results in a unit time. Therefore, the transaction rate is a key factor to enhance the performance of stream data processing systems of which input data come from remote stream data sources.

We use video surveillance systems as typical examples of stream data processing

systems. However, our proposed approach does not depend on the contents of the data when the system architecture is the same as our target system mentioned in this section and the contents of the data have the quality attributes. For example, the contents of the data for our proposed approach can be temperature, SLAM (Simultaneous Localization and Mapping), or so on.

1.2 Research Objectives

This study aims to enhance the performances mentioned above. The research objectives of this study are described in the following.

- Reduction of transaction time
  As explained in the previous section, one of the performance indexes in our target stream data processing systems is transaction time. To reduce the transaction time, many methods have been proposed, as described in Section 2.2. Most of these methods degrade the data qualities such as the image data resolutions to reduce the communication traffic between the stream data sources and the processing computer. However, the application performance may deteriorate with the data quality degradation. For example, the detection accuracies for object detection applications decrease when the image data resolutions are excessively low. Hence, one of the objectives of this study is to reduce the transaction time while maintaining the application performance. The relevant details are presented in Chapter 2.

- Improvement of transaction rate
  The other performance index is transaction rate. To improve the transaction rates, several methods have been proposed, as described in Section 3.2. These methods target periodic stream transactions and assume static transaction intervals as the focus on video applications, in which the transaction intervals (the frame rate) are not changed frequently. If the frame rate changes frequently, the quality of the experience degrades as a trade-off for changing the transaction intervals. However, when the transaction interval is long, it results in a lower transaction rate. On the other hand, when the transaction interval is excessively short, the transactions overlap, which causes long transaction time and high computational load on the processing computers, and thus the transaction rate decreases. Hence, one of the

objectives of this study is to improve the transaction rate considering the dynamic adjustment of the transaction interval. The relevant details are presented in Chapter 3.

- Implementation of the proposed approach
  To investigate whether the proposed approaches for the transaction time reduction and transaction rate enhancement are effective in real situations, we implement a video surveillance system incorporated with the proposed approaches and investigate the differences in the results obtained from the implemented system and simulation results. Chapter 4 presents the relevant details.

1.3 Related Work

This section first provides an overview and introduction of the studies relating to stream data and video data processing systems. After that, we will introduce the studies relating to stream data sources. The studies relating to transaction time, transaction rate, and video surveillance systems are described in Subsections 2.2, 3.2, and 4.2, respectively.

1.3.1 Stream Data Processing Systems

Several researchers have proposed stream data processing systems for several decades. Recent researches focus on cloud computing environments, IoT devices, and so on. The stream data processing system proposed in [21] used a cloud service to analyze large amounts of heterogeneous and high-frequency sensor data. The stream data targeted in the research are generated by IoT (Internet of Things) devices connected to the Internet. The stream data processing system proposed in [22] also dealt with the stream data that came from remote stream data sources. The system adopted in-memory streaming and caching (temporally storing the received data to the memory) technologies for fast processing. The system extended Kafka streams. Similar to these studies, our target is the stream data processing systems of which input data come from remote stream data sources.

A two-layer system architecture for stream data processing was proposed in [23]. In the first layer, the system checks the received data and determines whether to proceed to the process in the second layer. The checking processes were designed for enabling a much lighter processing load than the processes in the second layer. Thus, this architecture

reduces the processing loads because the system does not execute unnecessary high load processes. This architecture is also similar to our approach in that unnecessary processes are skipped by checking their necessity. This method always processes all the sensor data in the first layer even if the size of the data is large. Therefore, the traffic in the communication network is not reduced that can cause a long transaction time.

A two-level indexing approach for data collection was proposed in [24]. In this approach, segmented data are first stored in the memory of the processing computer having a tree structure in the first level. Next, data segments are passed to the second level with its reference data to the tree. This approach enables rapid data acquisition because the processing computer can access each data segment rapidly by using its reference data. This approach can be applied together with our approach.

As explained in Section 1.1, object detection systems are one of the major systems using stream data processing techniques. There are many researches for object detection systems. To improve the system performances for embedded IoT devices, a parallel and light object detection framework was proposed in [25]. A noise reduction mechanism by background subtractions for object detection was proposed in [26]. To detect objects in a short time, a pipelining technique for general purpose graphics processing units (GPGPUs) was proposed in [27], a method using several GPUs in parallel was proposed in [28], and an occluded-region reduction method was proposed in [29]. However, these object detection systems did not consider the necessity of the original quality data, which our proposed approach considers to improve the performance indexes.

1.3.2 Video Data Processing

We introduce the studies relating to video data processing in this subsection because one of the attractive stream data is video stream data. For the cases that the computational resources of the camera devices are insufficient for video data processing, the camera devices of the proposed systems in [30–33] continuously transmit their captured image data to remote processing computers. Similar to these studies, the input data in our assumed system come from remote stream data sources. To reduce the processing time on the processing computer, several methods have been proposed.

A parallel processing technique for video data processing was proposed in [34]. The data are distributed to multiple slave processing computers by a primary processing

computer, and the results are collected by the primary processing computer. A parallel processing technique targeting connected vehicles as stream data sources was proposed in [35]. In the technique, the processes are modeled as a directed graph and are mapped to the logical network consisting of the connected vehicles. These methods reduced the transaction time by implementing parallel executions using several processing computers. These techniques require several processing computers. Stream processing systems targeted in this study consist of a single processing computer and stream data sources equipped with compact computers.

The video data processing system proposed in [36] uses the GRIB (Generally Regularly Distributed Information in Binary) technique to improve the speed of video data processes. Under the GRIB, the processing computer performs video encoding/decoding processes in parallel by packing the video data into the GRIB code in binary format. This technique can be applied to our proposed approach by using the GRIB code.

The video processing systems that realize real-time object tracking using GPGPUs were developed in [37, 38]. GPGPUs can often reduce the image processing time further compared with CPUs because their architectures are designed to enable faster image processing. If the processing computer has GPGPUs, we can adopt similar approach to these studies.

To reduce video data sizes, various systems use data compression techniques. The system proposed in [39] uses a data compression algorithm enabling to keep data quality. To prevent data quality degradation for wireless communications, several image data compression techniques were proposed in [40, 41]. Moreover, several video coding systems were developed to reduce video data sizes by compressing the data in [42–44]. Furthermore, several researchers proposed data compression techniques to reduce the data amount for video stream data transmissions [45–47]. A video surveillance system using a video encoding/decoding algorithm with data compression for fast data transmissions between the camera devices and the processing computer was proposed in [48]. The system architecture in the research is very similar to that in our study. These studies did not consider the necessity of the original quality data, and thus applied the data compression techniques according to the application requirements. One of the disadvantages of compressing video data is the time required for compressing the data at the data sources and at the processing computer. These data compression techniques can be applied for each data item in our proposed approach.

Several existing encoding techniques consider the data qualities, such as progressive JPEG (Joint Photographic Experts Group) [4]. In the progressive JPEG-based approach, the image data quality is gradually increased. At the beginning of the image data reception, a blurred image is shown, and the resolution later improves as the subsequent data arrive. Scalable video coding (SVC) on H.264 is a video compression algorithm that enables the performance enhancement for video transmission even under a low-bandwidth network[5]. As an extension of the H.264 codec standard, H.264 SVC allows multi-platform distribution and adaptive video bitrate from a single video file. These approaches can control the data quality according to the network congestion level. However, in these encoding techniques, the processing computer always finally receives the designated quality data. In our proposed approach, the stream data sources transmit higher quality data only when they are required by the processing computer (see Chapter 2 for details).

1.3.3 Stream Data Sources

Targeting stream data processing systems in which a processing computer processes stream data transmitted by remote stream data sources, many methods have been proposed to reduce the communication traffic ([49–52]). Some types of the sensor values such as temperature or humidity do not change sharply in a second due to temporal or spatial dependency. Therefore, a scheme for reducing the communication traffic from sensor devices by removing transmissions of similar data was proposed in [53]. The scheme also uses an in-memory computing technique to reduce the processing time. A method to reduce the number of the stream data sources using their physical positions was proposed in [54]. The method targets finding an extremely different value in stream data and removes the data sources whose stream data show correlations with others. However, these methods do not focus on data quality and do not consider the necessity of the original data quality. In cases where stream data sources run on batteries, power consumption reduction lengthens their lifetimes. The power consumptions are generally reduced by reducing the computational loads on the sensor devices. Therefore, the method proposed in [55] attempts to reduce the power consumption by omitting several data transmissions and processes. However, omitting them decreases the application performances such as

[4]https://cloudinary.com/blog/progressive_jpegs_and_green_martians

[5]https://www.mistralsolutions.com/articles/h-264-svc-update-h-264-video-compression-standard/

the number of the data items received.

A method to reduce communication traffic between stream data sources and processing computers by decreasing the video bitrates was proposed in [56]. The video data sources set a high bitrate only if objects are detected in the recorded image data. In [57], a dynamic bitrate adaptation scheme for mobile devices was proposed and evaluated in real situations. In the scheme, the system dynamically selects the appropriate bitrate for the mobile device based on the data amount buffered in the memory. In addition, a method to control the buffer for transactions to maintain the transaction rates on the stream data sources was proposed in [58]. A method to reduce the communication delays by controlling the number of data packets for transactions was proposed in [59]. A method to reduce the processing loads on camera devices was proposed in [60]. The method uses a light load CoAP (Constrained Application Protocol) and controlls the communication traffic not to make the camera devices overloaded. However, these researches did not focus on the progressive improvement of data quality, which is the key idea of our proposed approach in this study.

1.4 Organization of the Dissertation

This dissertation consists of five chapters, and the rest of this dissertation is organized as follows.

Chapter 2 describes our proposed progressive quality improvement (PQI) approach for our target systems. The objective is to reduce the transaction time. Each stream data source constructs several data items having different qualities from the original data, and the data items are transmitted only if they are requested by the processing computer. Accordingly, the quality of the data processed in the processing computer progressively improves. We evaluate the PQI approach using our developed simulator. The evaluation parameters are the transaction intervals, number of qualities, and final probability (we consider progressive coding for image data, and the final probability is the probability of proceeding to the original quality). The evaluation results show that under a small final probability, the transaction time of the PQI approach is smaller than that of the conventional approach.

Chapter 3 describes a method to increase the transaction rate under the PQI approach, termed the PQI-CDI (PQI with Cycle-based Dynamic Interval) method. The transaction

interval is dynamically changed based on the transaction time. The transaction intervals for each stream data are changed every time when a predetermined number of transactions are finished. Moreover, due to the adoption of the PQI approach, the communication time and the processing time are reduced. We evaluate our proposed method by using our developed simulator. The evaluation results show that the proposed method can enhance the transaction rate compared with that of the conventional method with static interval.

Chapter 4 explains the implementation of a video surveillance system incorporated with the proposed PQI-CDI method. In the implementation, three camera devices and one processing computer were connected via a local area network. We developed two softwares run on each of them. To investigate the differences between the results obtained by our developed simulator and those by our implemented system, we measure the transaction time, transaction rate, and the jitters of the transaction intervals under the implemented system. The experimental results show that the PQI-CDI method can improve the transaction time and the transaction rate in actual situations although the performances are not always similar to the simulation results.

Chapter 5 presents the concluding remarks and describe several future research directions.

Chapter 2

Transaction Time Reduction for Stream Data Processing

2.1 Introduction

In most applications for stream data processing systems, a shorter transaction time enables earlier process completion, and thus leads the improvement of application performance. In the example of perpetrators detection in Section 1.1, the delay in identifying the perpetrators recorded in the image data shortens as the transaction time is reduced. Therefore, transaction time is one of the main factors to improve the performance of our target stream data processing systems.

In our assumed system in this study, stream data sources transmit stream data to the processing computer. The reduction of the data amount that the data sources transmit leads to a reduction in the transaction time because the transaction time includes the communication time and the processing time. A larger data amount causes a longer communication time and a longer processing time. Therefore, most of the existing methods reduce the transaction time by reducing the data amount transmitted from the data sources [61–67]. One of the major approaches to reduce the data amount is the degradation of data quality, e.g., reducing image data resolutions. The data amount after quality degradation is smaller than that of the original data. Therefore, the transaction time is reduced. However, the quality degradation deteriorates the application performance such as the detection accuracy for object detection and the position estimation accuracy for SLAM (Simultaneous Localization and Mapping).

The key point in preventing deterioration of the application performance is that the

processing computer obtains original quality data as soon as it is needed to perform the transaction for the data. In some cases, the original quality data are not needed to establish the transaction. For example, in the case of perpetrator detection, if there are no faces (not only perpetrators) in the image data, the processing computer does not need the original quality data because perpetrators are not recorded in the image. The processing computer can check the necessity of the original quality data by using low quality data. Definitely, the system needs to control the quality not to deteriorate the application performance. The data amount of lower quality data is generally smaller than that of the original quality data. Therefore, by checking the necessity of the original quality data using lower quality data, it is possible for the processing computer to obtain the original quality data only when it is needed. The average data amount needed to establish the transaction is reduced, and thus the transaction time is reduced.

Hence, in this chapter, we propose a progressive quality improvement (PQI) approach for efficient stream data processing. The goal of this chapter is to reduce the transaction time without deteriorating the application performance. In our approach, each stream data source constructs data of that quality is lower than the original quality. The processing computer progressively collects data needed to obtain higher quality data only if they are needed. A detailed explanation of this technique is provided in Section 2.4. The drawback of collecting and processing data of several qualities is that the transaction time increases when the process proceeds finally to the original quality. However, if the probability that the process proceeds to the original quality is low, the approach can reduce the average data processing burden, thereby reducing the average transaction time. The contributions of this chapter are summarized as follows:

(1) An approach is proposed to reduce the transaction time by progressively collecting higher quality data.
(2) Evaluations of the proposed approach are performed by our developed simulator.

The rest of this chapter is organized as follows. In Section 2.2, some previous studies related to our proposed approach are presented. In Section 2.3, the assumed system environments are explained. Our proposed approach is explained in Section 2.4, is evaluated in Section 2.5, and is discussed in Section 2.6. Finally, the chapter is concluded in Section 2.7.

2.2 Related Work for Transaction Time Reduction

There are various studies related to the transaction time reduction for stream data processing systems of which input data come from remote stream data sources. We introduced some stream data processing systems in Subsection 1.3.1. Here, we introduce some related work especially from the view point of transaction time reduction.

An efficient computational resources (CPU cores, memories) allocation scheme for stream data processing was developed in [68]. In the scheme, processing computer allocates the dedicated computational resources for each data stream. The processing time for each stream data is reduced because the probability of occurring the computational overheads such as swapping, page faults, and thrashing is reduced. A method to reduce the delay for starting processing the data was proposed in [69]. This method employs a queueing model for buffering the data of each stream at the processing computer. The processing computer prepares separated data queues and selects the data to be processed in each queue to reduce the processing delay. For object detection systems, which is a typical application in this study, [70] developed a fast object detection framework. The programs and the communication parameters of the framework are optimized for the target systems and thus establishes faster object detection. However, these methods do not consider the necessity of the original data quality because they do not focus on data quality.

Some researches on wireless sensor network (WSN) systems have aimed to fast collection of sensor data which periodically transmitted from sensor devices and are related to transaction time reduction. A processing loads distribution method for WSN was proposed in [71]. The method adopts in-network processing, i.e., queries such as data selection or data aggregation are executed at each sensor device which is an intermediate node for the data collection network. In-network processing reduces the data amount to be communicated in the network resulting in the transaction time reduction. [72] and [73] proposed mechanisms to dynamically adjust the routes for sensor data collection in WSN. They are similar to the mechanisms with software defined networks and the routes are controlled by a management device so that the sink device can receive sensor data faster. However, these methods do not consider data quality and the sink device collects the sensor data with the original quality.

Some video codecs adopt multi-quality video, which include several video quality

data in a video data stream. Therefore, some methods to improve video data quality progressively were proposed. A quality selection considering the players' buffer capacities to reduce the number of video pauses was proposed in [74]. A faster motion vector calculation method was proposed in [75]. Motion vectors are the directions of moving objects and are used for video data compression. The method calculates the motion vector in a short time by using a low quality video data. Several video data compression techniques such as Wyner-Ziv coding could create several quality data as intermediate data. Methods to exploit them to establish fine and fast video data compression were proposed in [76–78]. However, these methods do not consider the necessity of the original quality data. Stream data processing systems sometimes do not need the original quality data for the processes as described in Section 1.3.

Some data compression techniques have been also proposed. We explained these techniques and their drawbacks in Subsection 1.3.2.

2.3 Assumed Environment

In this section, we explain our assumed environment for stream data processing system.

2.3.1 System Architecture

Figure 2.1 displays an image of our assumed environment. A typical application of stream data processing in this study is object detection systems. Therefore, we drew the figure assuming that the application is a video surveillance system. The users designate processes such as those for perpetrator detection to the processing computer. The processing computer receives necessary data for the processes and continuously executes the designated processes every time when transaction starts.

Various IoT (Internet of Things) devices, such as surveillance cameras, continuously get data about their observations such as image data and act as stream data sources. These data sources and the processing computer connect to a computer network such as a local area network or the Internet.

The stream data sources and the processing computer can continuously communicate with each other via the computer network. The stream data sources construct data items from their observed original data and transmit them to the processing computer when

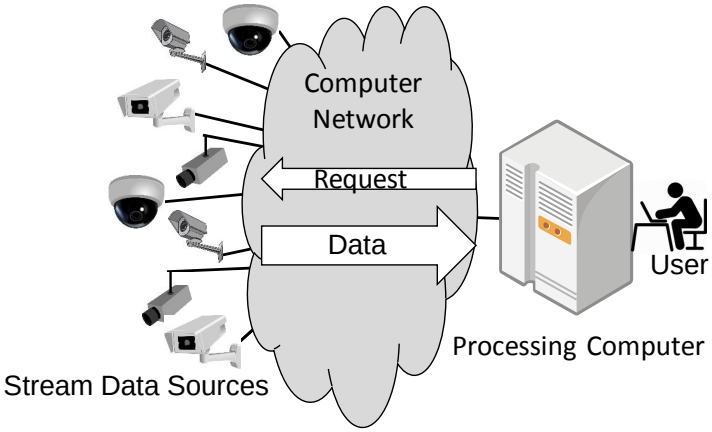


Figure 2.1: An image of our assumed environment

fixed interval elapses. For example, surveillance cameras transmit image data of a video frame every 100 [ms] when the frame rate is 10 [Hz]. The processing computer can request data to the stream data sources if it needs data that the stream data sources own. The stream data sources transmit the requested data as the responses for the requests.

2.3.2 Application Scenario

In this subsection, we introduce an application scenario. Suppose surveillance cameras are deployed in an area, and the cameras and a processing computer are connected to a local area network. The surveillance cameras, the processing computer, and their connected network use similar architecture as that explained in the previous subsection. We assume a perpetrator detection service. The service manager designates a process that notifies the user when designated perpetrators are detected in the image data captured by the surveillance cameras. For this, the user submits the face images of the perpetrators to the processing computer beforehand. The processing computer continuously executes processes to detect faces in the received image data. When the processing computer detects faces in image data, it checks whether the matched faces are those of perpetrators. If the processing computer detects perpetrators' faces, it sends a notification to the user by e-mail or other messaging services.

2.3.3 Symbol Definitions

In this subsection, we define the symbols for our assumed system. Given a system with N stream data sources that transmit image data every C_n ($n = 1, \cdots, N$) time unit, n is the number of stream data sources. Let $D_{n,a}(t)$ denote the original data of the stream data source n at the tth transaction. a represents 'all' and is not a variable. The stream data sources can construct Q data items, $D_{n,q}(t)$ ($q = 1, \cdots, Q$), which provide the data needed to get the qth quality data of $D_{n,a}(t)$. Subsection 2.4.2.3 describes how to construct them in our proposed approach. A data size of $D_{n,q}(t)$ is denoted by $S_{n,q}(t)$. $GT_{n,q}(t)$ denotes the generation time of $D_{n,q}(t)$ on the data source n. $ST_{n,q}(t)$ is the time to start processing $D_{n,q}(t)$, and $FT_{n,q}(t)$ is the time to finish processing it on the processing computer. Here, $FT_{n,q}(t) = ST_{n,q}(t) + P_{n,q}(t)$. $P_{n,q}(t)$ is the time needed to process $D_{n,q}(t)$. The transaction time, $TT_n(t)$, is the time consumed from data generation to transaction completion. In the case that the processing computer processes image data sequentially from the lowest quality, $D_{n,1}(t)$, to the $e_n(t)$th quality data, $D_{n,e_n(t)}(t)$, the transaction completion time is the time to finish processing $D_{n,e_n(t)}(t)$. The time to finish processing $D_{n,e_n(t)}(t)$ is denoted by $FT_{n,e_n(t)}(t)$ by the definition. The data generation time is the time to generate $D_{n,1}(t)$, the lowest quality data. The generation time of $D_{n,1}(t)$ is denoted by $GT_{n,1}(t)$. Therefore, $TT_n(t)$ is given by the following equation:

$$TT_n(t) = FT_{n,e_n(t)}(t) - GT_{n,1}(t). \tag{2.1}$$

The average transaction time for data-source n is

$$\frac{1}{T}\sum_{t=1}^{T} TT_n(t), \tag{2.2}$$

where T is the total number of transactions. The objective is to minimize (2.2).

We denote the probability that the processing computer executes the process for the $(p+1)$th ($p = 1, \cdots, Q-1$) quality data after finishing the process for the pth quality data at the tth transaction as $PProb_{n,p}(t)$. For example, the probability to request $D_{1,2}(1)$ when the processing computer finishes processing $D_{1,1}(1)$ is $PProb_{1,1}(1)$.

2.4 PQI Approach

In this section, we explain our proposed approach.

2.4.1 Basic Idea

Generally, data have certain qualities, and the original data gives the highest quality. Processes can be executed even if the data quality is lower than the original quality, and data with the highest original quality often give the best performance for applications. For example, one of the quality indexes of image data is resolution. Image data with 640 x 480 pixels have a higher quality than image data with 320 x 240 pixels. Image processing to detect faces can be executed for various pixel sizes; whereas higher resolution image data generally gives higher accuracy because they have more information. In the SLAM systems, the LiDAR (Light Detection And Ranging) data can be represented as an image with depth (distance) information. Autonomous robots can change their movement direction to avoid the obstacles using reduced resolution LiDAR data without detecting the detailed shape of the obstacles. However, a higher resolution LiDAR data is needed only if the robot needs to manipulate the obstacles to go through them.

When the processing computer processes data sequentially in the ascending order of quality in a transaction, they can skip to the next transaction when the subsequent processes for higher quality data are meaningless. Let us assume, similar to the example in the introduction section, that a processing computer executes the processes for detecting perpetrators in video data streams. The processing computer first receives the lowest quality image data of a frame and executes the processes to detect faces in the image. In cases where the faces are not detected, the processing computer skips the processing of higher quality image data because perpetrators will not appear in the frame. The data amount of lower quality data is smaller than that of the original quality data. Therefore, if the probability to proceed to the processes of higher quality data is small, the total data amount required for each transaction can be reduced compared with the cases in which the stream data sources always transmit the original quality data. Thus, the transaction times are reduced by maintaining the application performance. This approach is called *progressive quality improvement* approach in this study.

2.4.2 Process Flows

In this subsection, we explain the process flows for our proposed PQI approach.

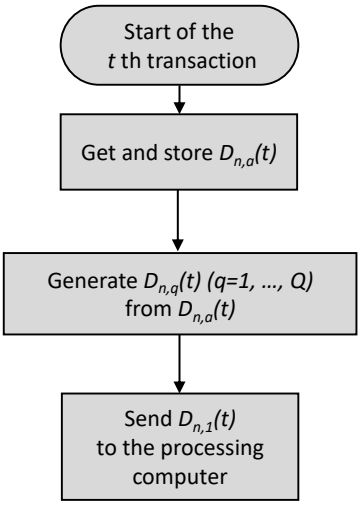


Figure 2.2: The flowchart for the stream data sources

2.4.2.1 Process Flow for Stream Data Sources

Figure 2.2 displays the flowchart for the stream data sources when the tth transaction starts. When the tth transaction begins, the data source n obtains $D_{n,a}(t)$ and stores the data in its storage temporarily. The data source then constructs $D_{n,q}(t)$ $(q = 1, \cdots, Q)$ from $D_{n,a}(t)$. First, the data source sends $D_{n,1}(t)$ to the processing computer. $D_{n,1}(t)$ is the lowest quality data. Thus, the stream data source n transmits the lowest quality data periodically every time when the transaction interval elapses. We regard object detection for image data as a typical application. A major format for the image data that have several qualities is the progressive-JPG. The image data encoded by the progressive-JPG simply concatenate all the differential data for each quality sequentially. Therefore, the camera devices (data sources) can get $D_{n,q}(t)$ easily by separating $D_{n,a}(t)$. Hence, we explained the case that the data sources construct all the data first. However, the PQI approach can be applied for the case that the data sources construct $D_{n,q}(t)$ upon its necessity. The construction time is short if the data sources can get $D_{n,q}(t)$ only by simple processes such as separating $D_{n,a}(t)$ like the progressive-JPG and is not a large problem.

Figure 2.3 shows the flowchart when higher quality data is requested. When the data source n receives a request for $D_{n,q}(t)$, it sends the data to the processing computer. We explain how to construct data for higher quality data in Clause 2.4.2.3. Apart from $D_{n,1}(t)$, higher quality data are transmitted upon request based on processing needs.

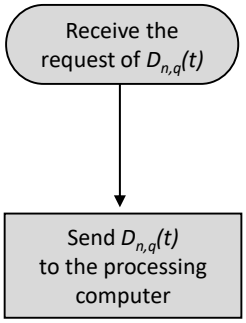


Figure 2.3: The flowchart when higher quality data is requested

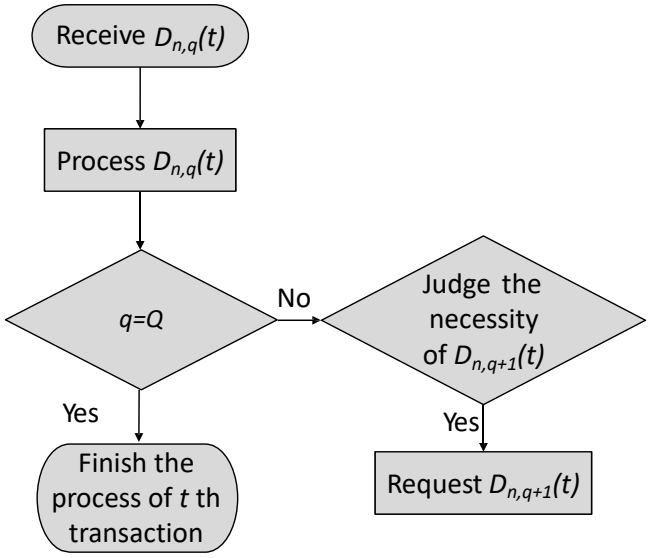


Figure 2.4: Process flow for the processing computer

2.4.2.2 Process Flow for Processing Computer

Figure 2.4 shows the flowchart of the processing computer. When it receives the data needed for getting the qth quality data of the data source n at the tth transaction, $D_{n,q}(t)$, it starts processing it. If the qth quality is the highest (original) quality Q, the tth transaction completes when processing $D_{n,q}(t)$ is finished. Otherwise, the processing computer judges the necessity of $D_{n,q+1}(t)$. If $D_{n,q+1}(t)$ is needed for proceeding the processes of the transaction, the processing computer requests it to the data source n. Note that the reception of the lowest quality data is push-based. Receptions of higher quality data are pull-based.

2.4.2.3 How to Use Data for Each Quality

There are two typical approaches to construct data for each quality.

In the first approach, the data sources transmit only differential data. In this case, the processing computer constructs the qth quality data item by combining the differential data and the $(q-1)$th quality data. For example, image data having 640x480 pixels resolution can be constructed using four images of 320x240 pixels resolution. Thus, the amount of the qth quality data item from the stream data source n at the tth transaction is given by

$$\sum_{i=1}^{q} S_{n,i}(t) + \alpha_{n,i}(t). \tag{2.3}$$

$S_{n,i}(t)$ is the data amount for the transmission and is the data amount needed to construct the ith quality data, not the data amount for the ith quality data. Here, $\alpha_{n,i}(t)$ is the overhead caused by combining the differential data with a lower quality data such as the data header or the data delimiter. Indeed, the image data encoded by the progressive-JPG concatenate several differential data sequentially putting the data delimiter in front of each differential data.

In the second approach, the data sources transmit the entire qth quality data item. When the data sources cannot pick up only differential data, e.g., the cases that the data format cannot separate each quality (BMP or RAW for image data), the system adopts this approach. In this case, the data amount for the transmission is the same as the data amount of the qth quality data. The overhead of this approach in the aspect of the increase of the data amount for the transmission is nothing because the data sources transmit the qth quality data itself.

The average transaction time decreases as the probability that the processes proceed to higher quality data decreases since more processes are skipped. Therefore, the approach that gives less probability to proceed to higher quality data should be selected in terms of the transaction time reduction.

2.4.3 Example Flow

This section demonstrates example flows under the conventional approach and the PQI approach.

Figure 2.5 shows the situation. In this example, the number of cameras is two.

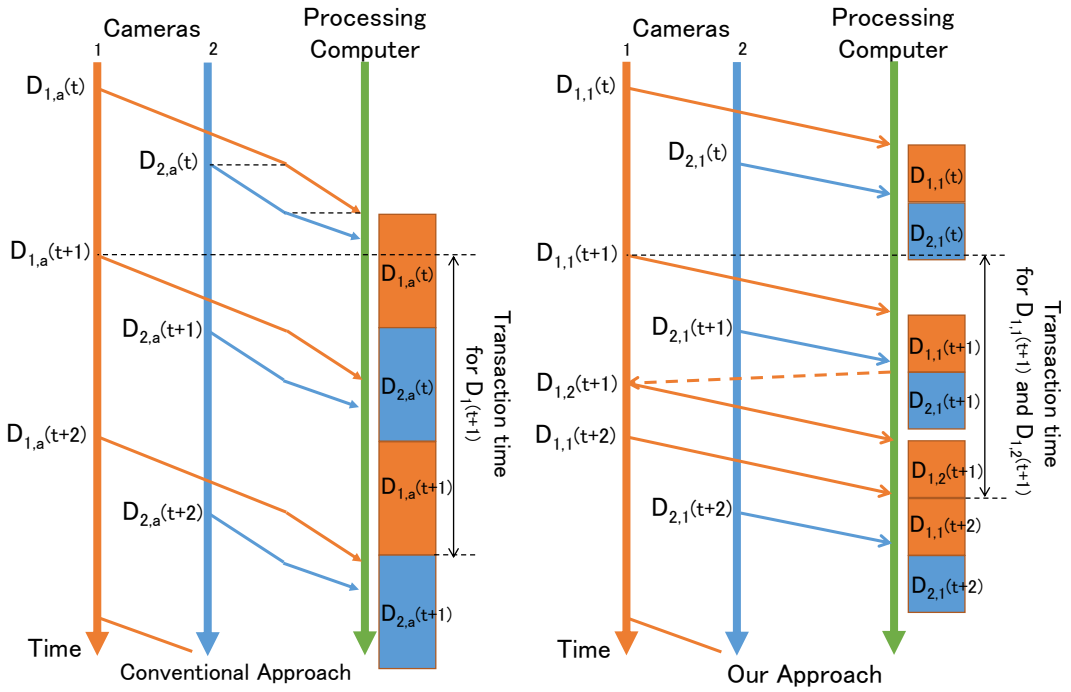


Figure 2.5: Example flow of a conventional approach vs. ours

First, an example transaction flow is explained under the conventional approach. Camera 1 sends its recorded image data frame-by-frame to the processing computer. In Figure 2.5, the tth frame is shown as $D_{1,a}(t)$ $(t = 1, \cdots, T)$. For example, when the frame rate is 10 [Hz], Camera 1 sends an image data every 0.1 [s]. Hence, $GT_{1,a}(t+1) = GT_{1,a}(t) + 0.1$. Additionally, Camera 2 sends its recorded image data to the processing computer. In this example, the frame rate for Camera 2 is the same as that of Camera 1, but the time to begin sending the data differs. After Camera 1 sends $D_{1,a}(t)$, Camera 2 sends $D_{2,a}(t)$. When the data transmissions from Camera 1 and Camera 2 overlap, the input communication bandwidth for the processing computer is equally divided. This is the reason why the communication speed of Camera 1 decreases while communicating with the processing computer, as shown in the figure. This is a natural phenomena if the bandwidth is shared with some entities and not a target problem. After Camera 1 finishes sending $D_{1,a}(t)$, the input communication bandwidth is dedicated to Camera 2 whose communication speed increases as shown in the figure. When the processing computer finishes receiving $D_{1,a}(t)$, it begins processing $D_{1,a}(t)$. While processing $D_{1,a}(t)$, the processing computer finishes receiving $D_{2,a}(t)$. As it processes $D_{1,a}(t)$ at this time, it stores the received

$D_{2,a}(t)$ in its buffer and begins processing it after finishing with $D_{1,a}(t)$. Similarly, while processing $D_{2,a}(t)$, the processing computer finishes receiving $D_{1,a}(t+1)$. It then begins processing after finishing with $D_{2,a}(t)$. The transaction time for $D_{1,a}(t+1)$ in this case is shown in the figure. This is the time consumed from the start of sending $D_{1,a}(t+1)$ to the end of $D_{1,a}(t+1)$ process.

Next, an example transaction flow is explained under our proposed PQI approach. Similar to the example for the conventional method, Cameras 1 and 2 send their recorded image data to the processing computer periodically. Unlike that in the conventional method, the image data are divided into two quality levels. The first is the lowest, and the second is the original quality. We assume that the data for the second quality only includes the difference in data from the first and the amount of the differential data for each quality data is the same. The data amount of $D_{n,a}(t)$ is given by the total of each data amounts of $D_{n,1}(t)$ and $D_{n,2}(t)$. Thereore, the time required to send $D_{n,q}(t)$ ($n = 1, 2$, $q = 1, 2$) becomes the half of that needed to send $D_{n,a}(t)$ over a fixed communication bandwidth. Therefore, the communication of $D_{1,1}(t)$ does not overlap $D_{2,1}(t)$, although the communication of $D_{1,a}(t)$ overlaps that of $D_{2,a}(t)$ under the conventional approach. The processing computer does not request the second quality data item in the first cycle because it cannot detect human faces in the image data. In the $(t+1)$th transaction, the processing computer begins processing $D_{1,1}(t+1)$ after receiving it. Then, the processing computer requests the second quality data item in the transaction because it detects human faces in the image data. When Camera 1 receives the request for $D_{1,2}(t+1)$, it begins transmission for the difference data between $D_{1,2}(t+1)$ and $D_{1,1}(t+1)$. In this example, the processing computer does not request the second quality data item of $D_{2,1}(t+1)$. After receiving $D_{1,2}(t+1)$, the processing computer executes the necessary processes and completes the transaction. The transaction time needed for the tth transaction and for Camera 1 is visualized in the figure as the time from the start of sending $D_{1,1}(t+1)$ to the completed processing of $D_{1,2}(t+1)$.

In summary, the transaction time of our approach is shorter than that of the conventional approach because several transmissions of higher quality data are skipped.

In the other approach, the data sources transmit the entire qth quality data in the transmissions for processing qth quality data. This approach is different from the approach in that the data sources transmit only differential data, the data amounts for transmissions from them tend to be larger. Thus, the communication time and the processing time

lengthen compared with those in the example flow. However, the transaction time can be reduced even in this approach if the transaction time is reduced by skipping several transmissions of higher quality data compared with the conventional approach.

2.5 Evaluation of the PQI Approach

The transaction time under the PQI approach depends on the number of streams, the transaction intervals, and the number of quality levels. We show the evaluation results changing these values using our developed simulator in this section.

2.5.1 Evaluation Setup

In this section, we use the simulator that we developed using the C++ programming language to get the evaluation results.

In this evaluation, we use the parameter values shown in Table 2.1. The 'Input Bandwidth' is the input communication bandwidth of the processing computer. When the processing computer communicates with multiple data sources simultaneously, the input bandwidth is fairly shared among them. The 'Output Bandwidth' is the output communication bandwidth of each data source. We set these parameters considering their reality. The 'Original Data Amount' is the amount of the original data that are processed in each transaction. We get this value by averaging the original data amount for the images in the open image dataset, 'pedestrians', from changedetection.net [1]. The dataset includes 1,099 standard JPEG images of 360x240 resolution. These images are frames obtained from a surveillance camera. The 'Processing Time Ratio' is the processing time divided by the data amount. We use the processing time ratio to get the processing time for each quality data including the time needed to judge the necessity of higher quality data. For example, when the time needed to process the image data of that amount is 42.0 [Kbytes] is 15.8 [ms], the processing time ratio=$0.0158/42000 = 3.8 \times 10^{-7}$. The processing time depends on various factors such as the processing power of the processing computer, the data amount of the image data, etc. However, since we cannot measure the performances without the processing time ratio, we got this value by measuring the average processing time needed for detecting faces and judging the necessity of higher quality data, using the

[1] http://jacarini.dinf.usherbrooke.ca/static/dataset/baseline/pedestrians.zip, 2012.

Table 2.1: Parameter values for simulation

Input Bandwidth	10 [Mbps]
Output Bandwidth	10 [Mbps]
Original Data Amount	42.0 [Kbytes]
Processing Time Ratio	3.8×10^{-7}

image data included in the pedestrian image dataset. The processing time for image data is given by its data amount times the processing ratio regardless of the data quality in the assumption for the evaluation. We use OpenCV [2] for detecting faces in the image data.

We assume that the data for each quality is used similar to the first approach explained in Clause 2.4.2.3.

Thus, the data sources transmit only differential data. $\alpha_{n,i}(t)$ in the formula (2.3) is 4 [bytes] based on the header and the footer size of progressive-JPG encoded image data. The progressive-JPG uses the frequency domain for encoding image data. The differential data for a higher quality data include the data for higher digits of DC (Direct Current) or AC (Alternating Current) coefficients. The number of the digits included in each data is generally fixed. The amount of the differential data for each quality becomes almost the same by encoding image data so that each differential data includes the same number of the digits. Therefore, we assume that the original data amount is fairly divided into each quality data. We simulate the stream processing system for 300 [s] to get the average transaction time. To illustrate image quality, we use substitute images in Figure 2.6 since the pedestrian images are copyrighted.

We compare our proposed PQI method (we term the method that uses the PQI approach by the PQI method here) with the No PQI method. The No PQI method corresponds to the conventional method where the stream data sources always transmit the original quality data.

2.5.2 Final Probability

For the evaluation, we use the final probability to proceed to the final quality data item, $FProb$. The details of the final probability are provided in this subsection. We set the same values for all $PProb_{n,p}(t)$ ($p = 1, \cdots, Q-1$), i.e., $PProb_{n,p}(t) = FProb^{1/Q-1}$.

[2] https://opencv.org/

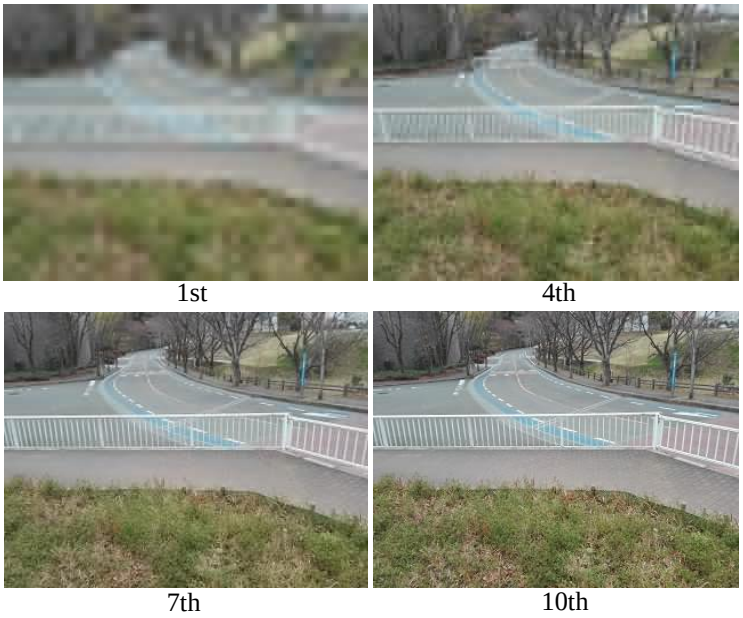


Figure 2.6: Example image qualities

We assume the application in that the values of $PProb$ are independent of qualities such as the case shown in Figure 2.15. Although the values of $PProb$ for each quality would be different even in this case, we used the same value for the evaluation as an example value.

We will explain how to get the final probabilities. Figure 2.7 is a processing image to explain the final probability. The figure shows two transactions, and the image data for each quality are constructed from the original quality data. The data size of a lower quality image is smaller. The upper part of the figure is the first transaction. Here, the process finishes at the lowest quality data because no cars are detected. Therefore, the process does not proceed to the final quality level. The lower part of the figure is the second transaction. Here, final quality data are obtained because a car is detected in the first quality image data. The probability of proceeding to the final quality data is $1/2 = 0.5$; therefore, the final probability is 0.5.

2.5.3 Transaction Time

To check the change of the transaction times along the time, we show them of each transaction in this subsection. The transaction times increase continuously if the

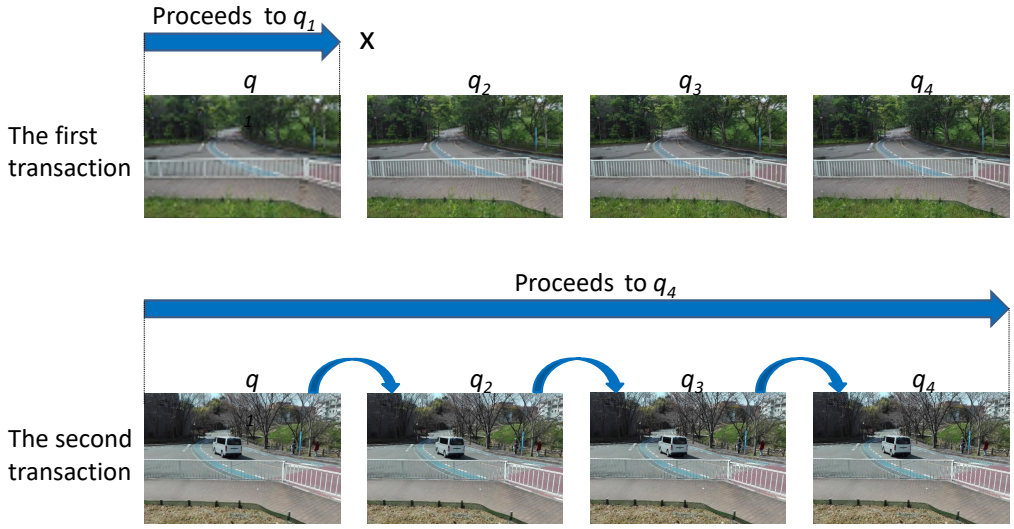


Figure 2.7: A processing image to explain the final probability

transactions keep to overlap with others. In this case, the transaction times can be unrealistically long. Otherwise, the transaction time will be in a certain time range.

We set the transaction interval to 400 [ms], the final probability to 0.7, and the number of quality levels to five as example values. The result is shown in Figure 2.8. To make the graph be easily seen, we show only the transaction times of the beginning 30 transactions. The horizontal axis is the transaction ID, and the vertical axis is the transaction time. We measured transaction times in the cases that the number of streams is five to seven.

We can see that the transaction time under the PQI method is shorter than the conventional method in many cases. In cases where the number of streams is less than seven, the transaction time does not increase so much during the entire simulation. On the other hand, in cases where the number of streams is seven, the transaction times tend to keep increasing with a certain cycle under the PQI method because the transactions keep overlapping with others. This is also similar to the No PQI method and the transaction times converge under the No PQI method only when the number of streams is five in this case. The transaction times when the number of streams is six surely tend to increase gradually as the transaction proceeds. Therefore, in the remaining subsections, we use the average transaction time for cases in which the transaction time converges, as the index of performance.

To further understand the distribution of the transaction times, we present the histograms

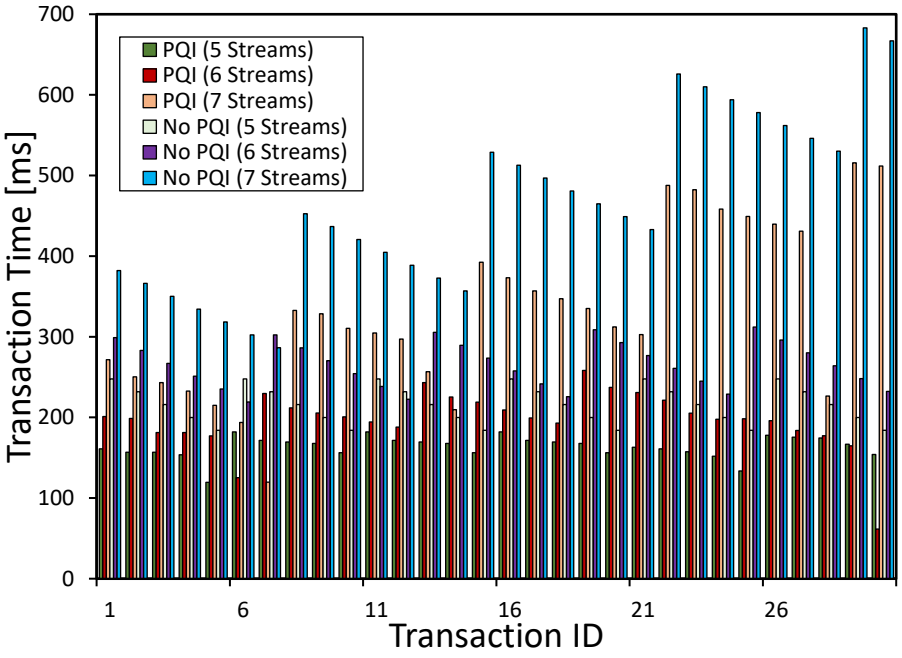


Figure 2.8: Transaction time of each transaction

of the transaction times. The histogram for the result when the number of streams is five is shown in Figure 2.9. This is the case that the transaction time converges under both methods. The horizontal axis is the range of transaction time and the vertical axis is the number of transactions of that transaction times fall in each range. The result shows that the PQI method keeps the transaction times under 200 [ms]. On the other hand, most of the transaction times under the No PQI method distribute at the transaction time close to 200 [ms]. Since the transaction time converges, the distribution of the transaction times has the upper limit, and it is 182 [ms] under the PQI method and is 248 [ms] under the No PQI method in the simulated situation.

Figure 2.10 shows the distribution of the transaction time when the number of streams is seven. This is the case that the transaction time keeps increasing under both methods. The horizontal axis is the range of transaction time and the vertical axis is the number of transactions of that transaction times fall in each range. In this case, the transaction times of most transactions are longer than 1.0 [s] because the transactions keep to overlap with others during the simulation.

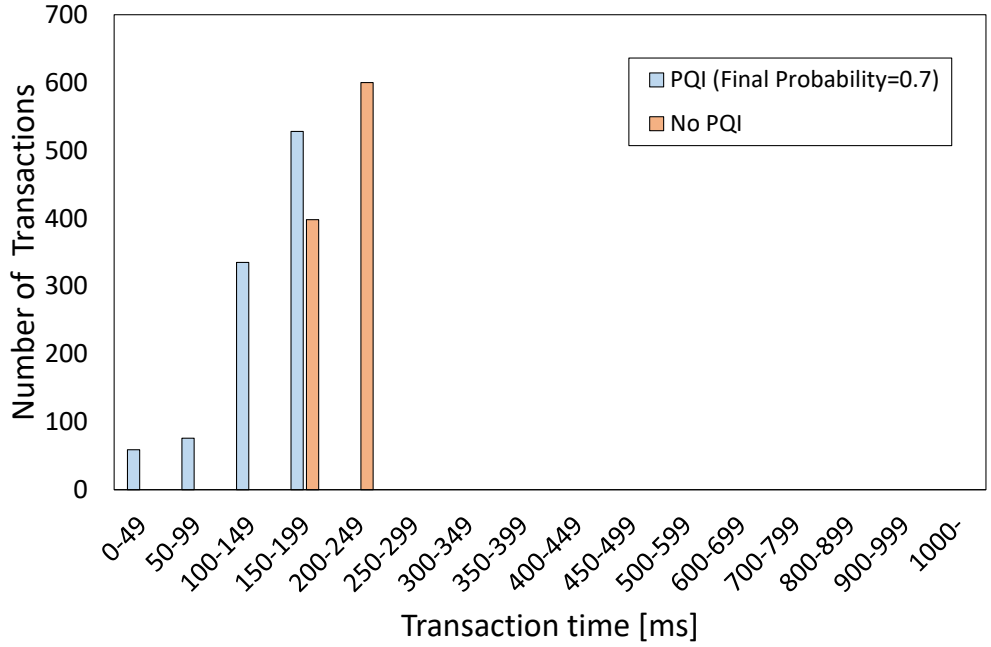


Figure 2.9: Distribution of transaction time when the number of streams is five

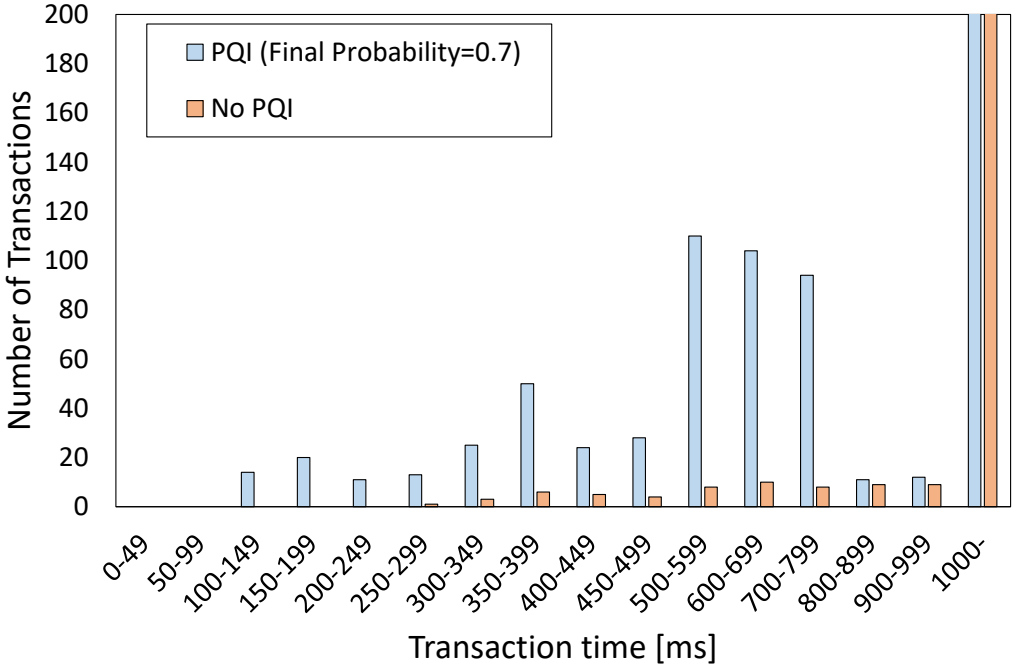


Figure 2.10: Distribution of transaction time when the number of streams is seven

2.5.4 Influence of Number of Streams

The data amount received by the processing computer increases as the number of streams increases. Therefore, a large number of streams has a possibility to keep increasing the transaction time. To investigate this phenomenon, we next investigate the influence of the number of streams.

We set the transaction interval to 400 [ms] and the number of quality levels to five as similar to the previous subsection. Figure 2.11 shows the result of the average transaction time changing the number of streams. We simulated the transaction time under different final probabilities (0.001, 0.01, 0.1, 0.3, 0.5, 0.7, and 1.0).

A larger final probability causes a longer transaction time because the processing computer collects and processes more data. The average transaction time under the No PQI method increases with the number of streams for cases where the number of streams was less than six because the data amount that the processing computer receives increases. For cases where the number of streams is greater than five, the average transaction time increases sharply because the transaction time increases over time as explained in Subsection 2.5.3. We observed similar phenomena with the PQI method. However, the maximum number of streams that the average transaction time does not increase sharply is larger compared with that of the No PQI method. For example, in the case where the final probability is 0.7, the average transaction time sharply increases when the number of streams is seven. Thus, the processing computer can collect data from more data sources using the PQI method.

In the case where the final probability is 1.0 under the PQI method, all the transactions proceed to the final quality. Thus, the amount of the data communicated between the processing computer and the stream data sources is almost the same as that for the No PQI method. Nevertheless, the average transaction time under the PQI method is shorter than that under the No PQI method. In the No PQI method, the processing computer finishes receiving the original data from all the stream data sources at the same time. After that, the processing computer processes them one by one in the FIFO manner. On the other hand, in the PQI method, the processing computer can receive data from different stream data sources while processing the data received from another stream data source because the transactions include several stages for processing the data for each quality. Since the processing computer can receive and process data simultaneously, the average transaction

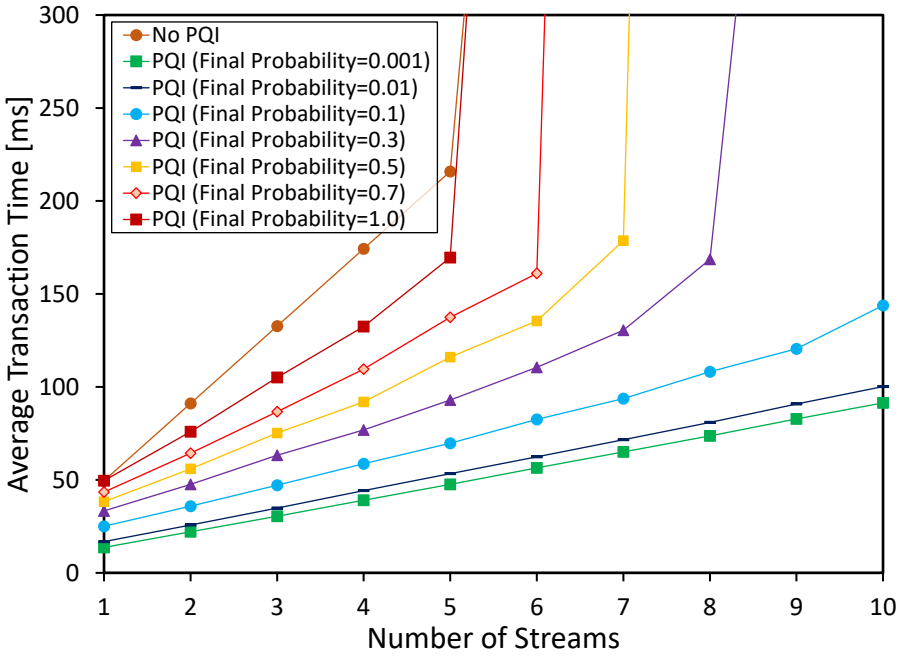


Figure 2.11: Average transaction time under different number of streams

time is shorter than the No PQI method even in the case where the final probability is 1.0.

2.5.5 Influence of Transaction Intervals

The probability that the transactions overlap with others increases as the transaction interval shortens. Thus, the transaction time keeps increasing if the transactions keep overlapping with others. To investigate this phenomenon, we next investigate the influence of the transaction interval.

We set the number of stream data sources to five, and the number of quality levels to five. The result of our evaluation is shown in Figure 2.12. The horizontal axis is the transaction interval and the vertical axis is the transaction time.

The transaction time keeps increasing when the transaction interval is excessively short as similar to the previous result, and the average transaction time sharply increases. In the No PQI method, the average transaction time is constant when the transaction interval is larger than 160 [ms] because the transactions do not overlap. In the PQI method, the shortest transaction interval that the average transaction time converges is shorter than that of the No PQI method. For example, in cases where the final probability is 0.7, the average transaction time converges in the PQI method even when the transaction

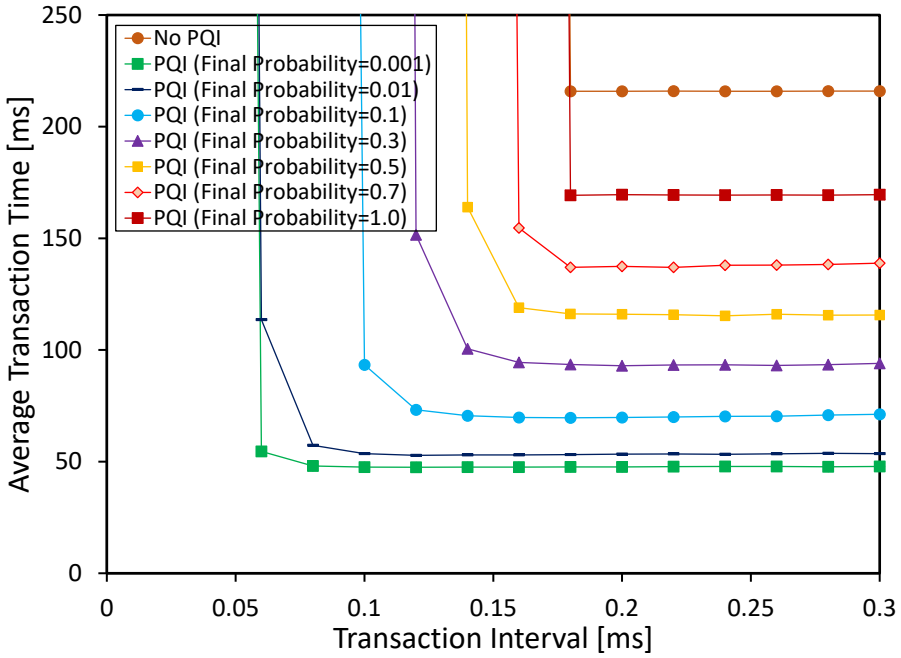


Figure 2.12: Average transaction time under different transaction intervals

interval is 160 [ms], whereas that of the No PQI sample sharply increases. Therefore, the processing computer can collect data with a shorter transaction interval by using the PQI method.

2.5.6 Influence of Number of Quality Levels

The number of quality levels influences the average processing time. A large number of quality levels is unrealistic because the number of quality levels has an upper limit in actual situations due to the encoding technique. For example, a general progressive-JPG encoded data has ten scans, and thus the maximum number of quality levels in this case, is ten. Therefore, the users of the proposed PQI approach such as system managers need to decide the number of quality levels considering the reduction of the transaction time and the reality. Hence, we measured the transaction time changing the number of quality levels.

We set the transaction interval to 400 [ms], and the number of streams to five. Figure 2.13 shows the average transaction time under the different number of quality levels. The horizontal axis is the number of quality levels, and the vertical axis is the average transaction time. The average transaction time for the case that the number of quality

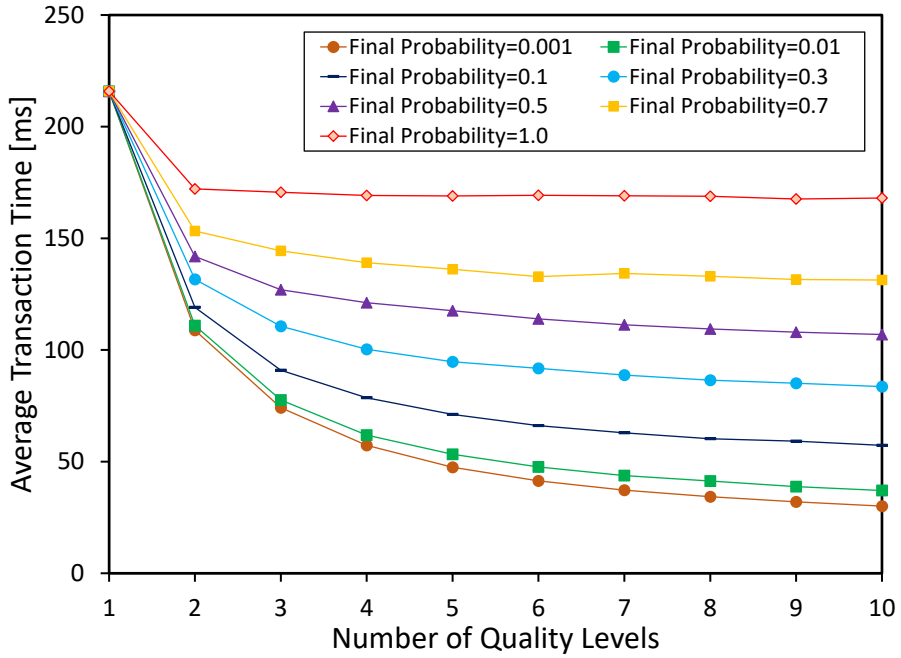


Figure 2.13: Average transaction time under different numbers of quality levels

levels is one represents that of the conventional No PQI method because the stream data sources always transmit the original quality data also in the PQI method.

The average transaction time decreases as the number of quality levels increases because the average data amount that the processing computer receives decreases as the original quality data is separated into more quality data. Thus, the processing computer can avoid redundant data receipts and processes, enabling the transaction time to decrease. The decreasing rate decreases as the number of quality levels increases because the amount of each quality data is in inverse proportional to the number of quality levels.

2.5.7 Influence of Final Probabilities

As shown in the previous evaluation results, the average transaction time depends on the final probability. The data amount received by the processing computer increases with the final probability increases. Therefore, a larger final probability has a larger possibility to keep increasing the transaction time. To investigate this, we next investigate the influence of the final probability.

To maintain the consistency with the previous results, we set the number of quality levels to five and the transaction interval to 400 [ms]. Figure 2.14 shows the result of

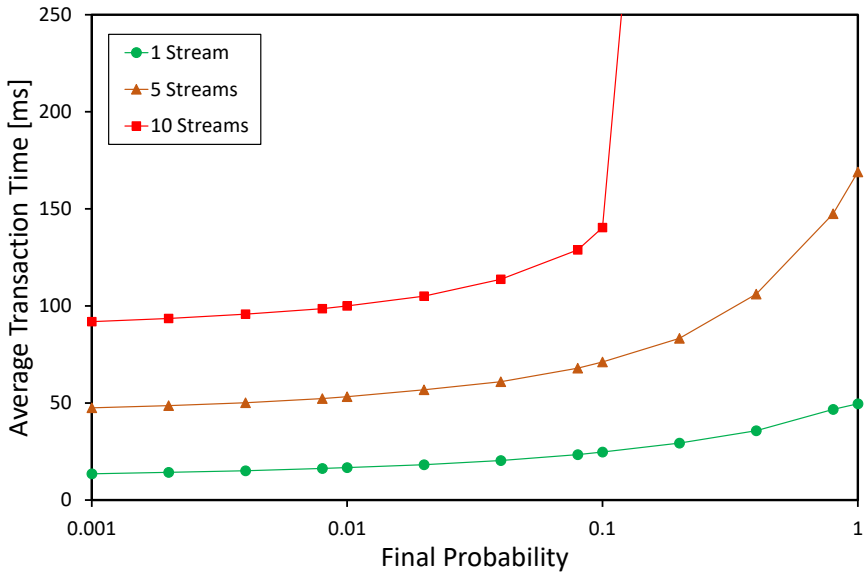


Figure 2.14: Average transaction time under different final probabilities

the average transaction time under different final probabilities. The horizontal axis is the final probability on a logarithmic scale. The vertical axis is the transaction time.

The transaction time increases with the final probability because the processing computer collects and processes more data. When the number of data streams is 10 and the final probability is larger than 0.1, the transaction time increases sharply because the transaction time is too long and the transactions overlap as similar to the sharp increases that appeared in other results.

2.6 Discussion

In this section, we discuss some points related to the performances of our proposed PQI approach.

2.6.1 Detection Performance

The value of the final probability for the preparator detection application depends on the performance of the face detection. Therefore, we evaluated the performance. For the evaluation, we used 250 images with 640 x 480 resolution in that there is one face and created two types of progressive-JPG images. The first 'Normal' type is created so that

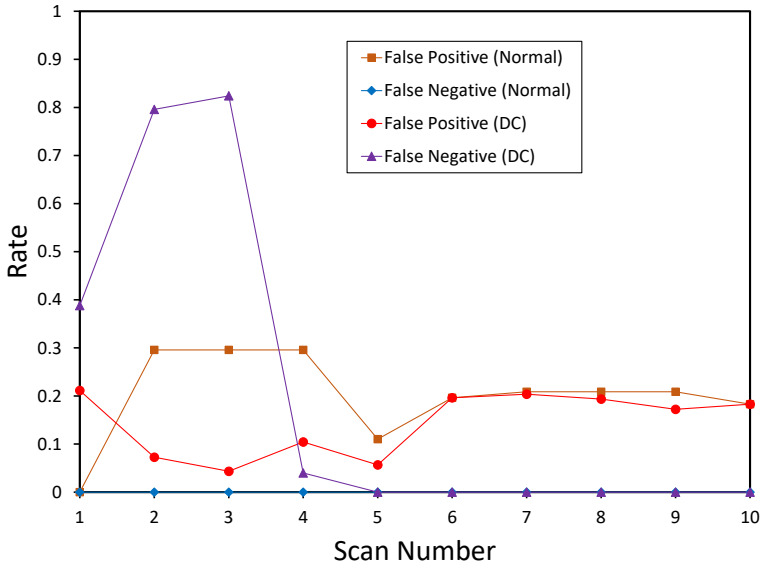


Figure 2.15: Detection Accuracies

the amount of the differential data for each scan becomes almost the same. The other 'DC' type is created so that the number of the significant digits for the DC coefficient increases as the scan number increases. The number of the scans is ten. We detected the faces in the images. Figure 2.15 shows the result. The horizontal axis is the scan number and the vertical axis is the rate value. The false positive rate means that the number of the incorrect face detections divided by the number of the detected faces. The false negative rate means that the number of the misdetected faces divided by the number of the faces in the images.

In the 'Normal' case, the false negative rate is zero under all the scan numbers. On the other hand, the false positive rate is approximately 0.2 except where the scan number is one. When the scan number is one, the image is unclear and the possibility of the incorrect detection becomes also low. In this case, the processing computer requests a higher quality data when faces are detected so as to reduce the false positive rate because the false negative rate is zero. In the 'DC' case, the false negative rate is zero when the quality level is larger than four. Therefore, if the processing computer requests a higher quality data when the faces are detected, the lowest quality should be larger than four. Otherwise, there is the possibility that the processing computer fails to get clear image even if faces appear in the image when the data quality is less than five.

2.6.2 Deciding Parameter Values

The users of the PQI approach such as system managers need to decide some parameter values. We discuss some of the main parameters for the PQI approach as follows. Although parameters such as the amount of each data item and the communication bandwidth influence the transaction time, they are not discussed here because the users cannot strictly control them generally.

- The number of streams: A larger number of data streams causes more communication traffic between the data sources and the processing computer. In cases where transactions continuously overlap, the transaction time increases toward infinity, as shown in Figure 2.11. Therefore, obviously, the number of streams should be small so that such situations do not occur.

- Transaction intervals: As shown in Figure 2.13, an excessively short transaction interval increases the transaction time. On the other hand, a longer transaction interval decreases the transaction rate. Therefore, the transaction interval should be decided considering the transaction time and the transaction rate. We propose a method to improve the transaction rate under the PQI approach in the next chapter. In the cases that the system manager adopts the proposed method in the next chapter, he/she does not need to decide the transaction interval because the value of the transaction interval is automatically calculated.

- The number of quality levels: As shown in Figure 2.13, a larger number of quality levels further reduces the average transaction time when the final probability is less than 1.0 because the average data amount transmitted from the data sources decreases. However, an excessively large number of quality levels is unrealistic because the number of quality levels has an upper limit in actual situations due to the encoding technique. A larger number of quality levels also requires more generations of data items causing a longer generation time. Moreover, a larger number of quality levels requires more processes on the stream data sources as discussed in the next item. Therefore, the number of quality levels should be decided considering the transaction time and the reality.

2.6.3 Processing Power

In the PQI approach, the computational load for generating the data items having several different quality levels occurs on the stream data sources. However, this is realistic because some camera devices implement the codec that enables several quality video data such as H.264/SVC. SVC is the technique to change the video quality based on the network congestion level. Since the process for generating the data having several different quality levels is similar to the process for generating several quality video data, the processing load caused by the data generation in the PQI is not a large problem.

Moreover, the processing load on the processing computer increases compared with the No PQI method when the final probability is 1.0 because it needs to judge the necessity of higher quality data in addition to the processing for the original (final) quality data. However, the load is reduced further under a smaller final probability. Therefore, in the situation that the final probability is small, there is a possibility that the average processing load on the processing computer is reduced.

2.7 Conclusion

Transaction time is one of the main factors that need to be considered to improve the performance of stream data processing applications. Transaction time can be further reduced by accounting for data quality. To reduce the transaction time, we proposed an efficient stream data processing method using a PQI approach. In our proposed method, the processing computers progressively collect higher quality data only in cases where they are needed for processing. By reducing the average data amount for data collection and processing, our proposed method reduces average transaction time. We used our developed simulation to evaluate the PQI approach and confirmed that the approach can reduce the transaction time further than the conventional No PQI method when the final probability is small. Moreover, we confirmed that the transaction time increases sharply when the number of streams or the final probability is excessively large, or the transaction interval is excessively short because the transactions continue to overlap with others.

Since a typical application of stream data processing is object detection system as described in Subsection 2.3.1, we set the parameter values for the evaluation assuming that the data type is image data. Meanwhile, our proposed PQI approach does not depend on

the contents of the data as far as they have the quality attributes. We found the following points: 1) Our proposed PQI approach can reduce the transaction time even when the final probability is 1.0 by using several quality data because the processing computer can perform the processes and the communications simultaneously, 2) A larger number of quality levels enables further transaction time reduction. However, an excessively large number of quality levels is unrealistic because the number of quality levels has an upper limit in actual situations due to the encoding technology.

One of the remaining issues is the use of multiple processing computers. The transaction time can be further reduced by distributing the processing loads to several processing computers. For this, an efficient algorithm for the data sources to select the processing computer to transmit data is required. Another issue is how to construct the data for each quality. A lower final probability gives a further transaction time reduction. Therefore, the system can reduce the transaction time further by constructing the data for each quality so as to make the average final probability lower. Moreover, we will investigate the performances where our proposed approach is applied for other contents data.

Chapter 3

A Method to Improve Transaction Rates under the PQI Approach

3.1 Introduction

In Chapter 2, we introduced the PQI approach to reduce the transaction time for stream data processing. In this chapter, we propose another method that improves the transaction rate under the PQI approach.

Most of the stream data processing applications such as video surveillance systems can adjust transaction intervals. For example, consumer video cameras generally have configuration settings to adjust the interval to capture the video frames. The transaction rate for an application varies depending on its transaction interval and transaction time. In such stream data processing applications, a higher transaction rate leads to more frequent transactions and enables the improvement of application performance. In the example of perpetrator detection, the number of faces detected in a time period increases as the transaction rate increases because the number of images that the processing computer receives increases as the transaction rate increases. If a perpetrator passes by quickly across the field of view of a camera, the surveillance system may miss the face of the perpetrator when the transaction rate is low. Therefore, the transaction rate is one of the main factors to improve the performance of stream data processing systems. In case where the transaction rate is excessively high and the processing computer does not detect new faces in different images, the number of the detected faces does not increase. In such cases, by giving an upper limit to the transaction interval, the system can avoid the unnecessary increase of the frame rate. In this chapter, we distinguish between

transactions and processes as similar to other chapters. A transaction consists of some processes (see Subsection 3.3).

To improve transaction rates, methods of reducing communication time for transferring data from the stream data sources to the processing computer have been published [79–81]. These studies aimed to increase the transaction rate by reducing the communication time. They assumed periodic stream data transactions and static transaction intervals because their general focus was on video applications that prefer static transaction interval. If the frame rate of the video changes frequently, the quality of the user's experiences decreases.

However, communication time and processing time change dynamically depending on the amounts of transmitted data in each transaction. For example, consider that the cameras transmit image data at 10 [fps] to the processing computer, i.e., the image data are sent every 100 [ms]. This time span corresponds to the transaction interval, and a shorter interval enables higher rates. Therefore, when the transaction interval is long, it results in a lower transaction rate. On the other hand, when the transaction interval is excessively short, the processing computer receives overlapping stream data, which causes a long transaction time and a high computational load on the processing computers. This results in decreasing the transaction rate. Moreover, we need to assume that the transaction time varies according to the necessity of the quality level in the stream data processing application when we apply the PQI approach.

Hence, this chapter aims to improve the transaction rate by changing the transaction intervals dynamically under the PQI approach. In our proposed method, the transaction interval is changed according to the requests from the processing computer. The transaction rate is further improved by changing transaction interval depending on the transaction time. A straightforward approach is to change the transaction interval for every transaction based on the previous transaction time. However, frequent changes in intervals are not preferable for visual applications such as video data processing in which a frequent interval change decreases the quality of experience. Therefore, we introduce a mechanism to avoid unmoderated changes of interval. The contributions of this chapter are summarized as follows:

(1) An algorithm is proposed to keep the transaction rate as high as possible by adjusting the transaction intervals.

(2) A method is proposed to avoid frequent transaction interval changes according to the application demand.

The remainder of this chapter is organized as follows. In Section 3.2, we introduce related works. Our proposed method is explained in Section 3.3, and the performance evaluation results are described in Section 3.4. Finally, we conclude the chapter in Section 3.5.

3.2 Related Work for Transaction Rate Improvement

To improve transaction rates, several methods have been proposed in the literature. A method to improve the transaction rate for multidimensional stream data was proposed in [82]. This method aims to increase the transaction rate by reducing the processing loads per transaction. In their method, the streaming data sources sends only the necessary dimensional axes of data for processing. As the processing computer does not receive unnecessary data, there is no need for data refinement activities to reduce processing loads. However, this method assumed a static transaction interval, thus the applications need to set an appropriate transaction interval for the multidimensional stream data.

The methods proposed in [83] and [84] use object detection frameworks to detect objects as fast as possible. They proposed fast object detection algorithms to improve the transaction rates. However, their frameworks does not care the necessary quality of image data which our PQI approach can adjust.

To improve transaction rates for object detection, various approaches have been proposed, such as searching-area reduction [85], detection accuracy improvement [86, 87], and dynamic background updating [88–91] focused on hardware (e.g., field-programmable gate arrays) for improvements. Their methods improve transaction rates by reducing the processing time. However, they also assume a static transaction interval. The performance and quality of object detection applications are not improved even there is an available capacity in the processing computer to process to achieve higher transaction rates.

GPGPUs, digital signal processors, and field-programmable gate arrays designated for stream data processing are recently developed to improve the transaction rate. In [92], the authors proposed a simple data processing model for these processing units to enable low complexity performance prediction for stream data processing. Their model is expressed by straight directed graphs with queues at each vertex. Vertices represent processing kernels. They compared the amount of the stream data (called flow) that the kernels receive per second in their proposed model with that in real situations. Their results

shows that the modeled flow matched with the actual flow when the queuing scheme is simple such as batching, i.e., picking up data from the queue when the data amount stored in the queue exceeds the threshold. However, the proposed model does not consider the data quality.

3.3 Proposed Method

We propose a method called the PQI-CDI (Progressive Quality Improvement approach with Cycle-based Dynamic Interval) method. In this section, we first discuss about the adjustment of the transaction interval to improve the transaction rate. After that, we explain the design and the algorithms of the PQI-CDI method. Further, we show an example how to improve the transaction rate under the PQI-CDI method.

3.3.1 Adjustment of Transaction Interval

In this subsection, we explain how transaction rate changes depending on transaction time and transaction interval.

Figure 3.1 shows the three cases which covers the different combinations of transaction time and transaction interval. We assume that there is one processing computer. For simplicity, we consider a case where the transaction time equals to processing time on the processing computer in this figure. In the figure, the horizontal axis shows the elapsed time. Each gray box represents the transaction time. The transaction time without overlapping is 50 [ms]. We assume that the computational resources of the processing computer are divided equally among each data processing in this example.

The line (1) indicates the case where the transaction interval is 70 [ms]. As the transaction interval is greater than the transaction time, the transactions do not overlap. However, the processing computer needs to wait for 20 [ms] for the next transaction after the previous transaction finishes. This wastage of time results in a smaller number of transaction processes. In this case, the number of transactions that finishes between 0 [ms] and 10 [ms] is one.

The line (2) indicates the case where the transaction interval is 30 [ms]. As the transaction interval is less than the transaction time, the transactions overlap. Because one processor on the processing computer processes all transactions, overlapping transactions

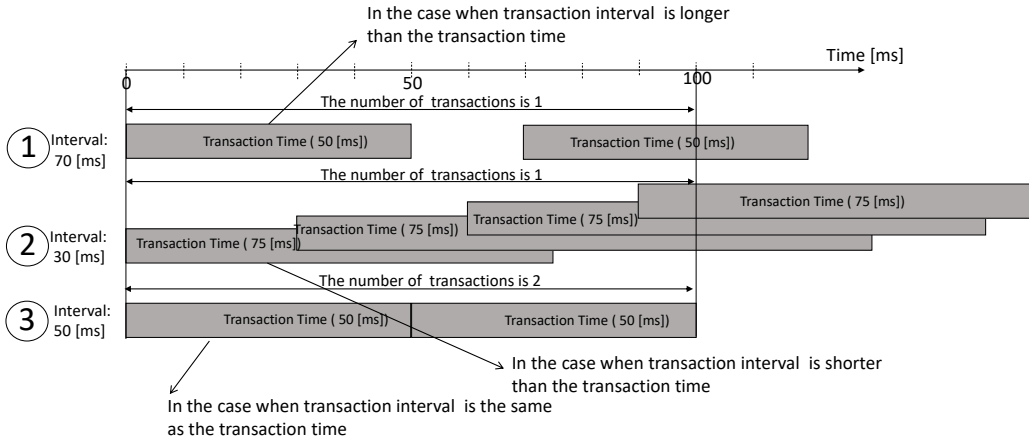


Figure 3.1: Example of transaction rates under different transaction intervals

cause resource competitions. The transaction time increases because the processing computer needs to execute processes to multiple transactions simultaneously. In this case, the transaction time for the first transaction is 630 [ms]. This is because the processing computer processes one data until 30 [ms] and retains the data that require $500-300 = 200$ [ms] for the process at this time. Between 300 [ms] and 600 [ms], the processing computer processes two data items and retains the data that require $200 - (600 - 300)/2 = 50$ [ms] for the process at this time. Between 60 [ms] and 630 [ms], the processing computer processes three data items and finishes the transaction at 630 [ms]. As in this case, overlapping transactions increase the transaction time and result in a lower transaction rate. In this case, the number of transactions that finishes between 0 [ms] and 100 [ms] is one.

The line (3) indicates the case where the transaction interval is 50 [ms]. As the transaction interval is the same as the transaction time, the transactions do not overlap and there is no wasted time in these transactions. In this case, the number of transactions that finish between 0 [ms] and 100 [ms] is two.

As shown in the above example, the highest transaction rate is obtained when the transaction interval is the same as the transaction time. If the transaction interval is smaller than the transaction time, the transaction time eventually gets longer and the transaction rate becomes smaller. In this case, the transaction rate converges near zero if the observation period has enough length. The transaction time fluctuates depending on the available processing power of the processing computer and the communication

bandwidth. Therefore, the transaction interval should be adjusted to dynamically changing transaction time to improve the transaction rate.

3.3.2 Design of Proposed Method

In the PQI-CDI method, the system changes transaction intervals dynamically and adopts the PQI approach to reduce transaction time. In our proposal, we assume that there is one processing computer to process transactions.

The details of the PQI approach and its evaluation results are described in Chapter 2, where we demonstrated that the PQI approach reduces transaction time. However, the transaction interval under the original PQI approach is static and cannot improve the transaction rate. Therefore, the PQI-CDI method dynamically changes the transaction intervals. For this, the PQI-CDI method determines the timings to change the intervals and new transaction intervals.

These are explained below.

3.3.2.1 Timings to Change Intervals

A frequent adjustment of intervals increases the transaction rate. Less frequent changes decrease the transaction rate because it also takes time to adjust the transaction intervals. The transaction interval is not changed until the next adjustment time even when the transactions are overlapped. The appropriate timing for interval changes depends on the communication time and the processing time as these times change dynamically. Therefore, we determine the period for changing transaction intervals by introducing a notion which we call *cycles*.

3.3.2.2 Determining New Intervals

In the PQI-CDI method, we define a parameter C_n which specifies a cycle length to change the intervals. When the number of the complete transactions reaches C_n, the processing computer changes the transaction intervals of the data source n. We assume the data sources (e.g., camera devices) can adjust the transaction interval arbitrarily. For example, on camera devices, we can implement the transaction interval adjustment function by specifying the timing of the camera image data capturing process. Once the transaction

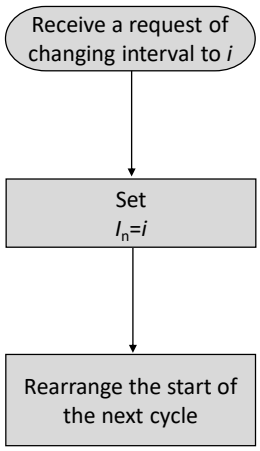


Figure 3.2: The flowchart for the data sources under the PQI-CDI method

interval changes, the processing computer starts counting the number of transactions. We adopt the average transaction time from the previous cycle as the new interval.

The average transaction time $AveTT_n(t)$ is given by the following equation:

$$AveTT_n(t) = \frac{\sum_{\tau=t-C_n+1}^{C_n} TT_n(\tau)}{C_n}. \tag{3.1}$$

$TT_n(\tau)$ is the transaction time of the data source n at the τth transaction.

3.3.3 Algorithms

In this subsection, we explain the algorithms for the data sources and the processing computer respectively.

Figure 3.2 shows the flowchart of data sources in the case where the processing computer requests the change of the transaction intervals. When the data source n receives the request to change the transaction interval to i, the data source changes the transaction interval and resets the counting value of the number of transactions for the next cycle.

Figure 3.3 shows the flowchart of the processing computer. When it receives $D_{n,q}(t)$, it processes the data. When $q = Q$, the tth transaction 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 process execution, the processing computer requests $D_{n,q}(t)$ to the stream data source n. Otherwise, the transaction finishes. We represent the counter which increments after the transaction from n finishes as c_n. In the PQI-CDI method, the processing computer

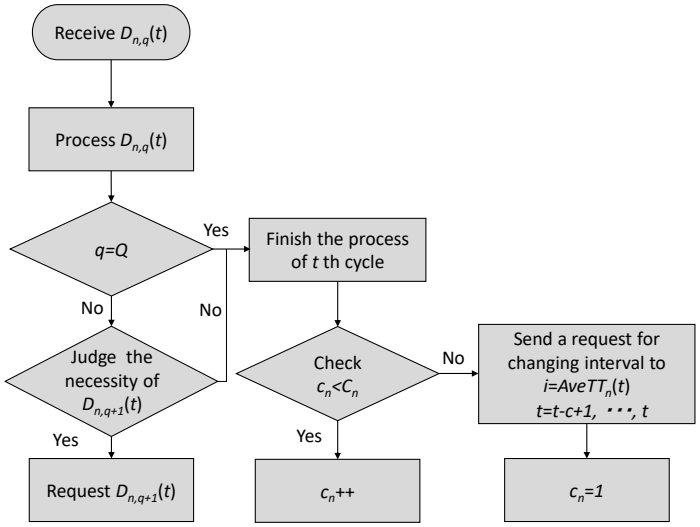


Figure 3.3: The flowchart for the processing computer under the PQI-CDI method

then checks whether c_n reaches C_n when a transaction finishes. Here, C_n is the interval needed to change the transaction interval of the data-source n. If c_n reaches C_n, the processing computer calculates $i = AveTT_n(t)$ and sends a request to n in order to change the transaction interval to i. Then, c_n is initialized as zero and the new cycle starts.

3.3.4 Example

Figure 3.4 illustrates the transaction rate of the conventional method (static transaction interval) and the PQI-CDI method. In the example, the number of the stream data sources is one for simplicity. The data source is a camera device and transmits image data to the processing computer.

We first explain the transaction rate under the static transaction interval. In the upper part of the figure, the transaction interval is static and is 100 [ms] for all transactions. The first transaction starts at 0 [ms], at which time the cameras transmit their recorded image data to the processing computer. After the processing computer finishes receiving the image data, it begins detecting objects in the image data. In this first transaction, these processes (shown in the blue area) finish after 40 [ms]. Therefore, the transaction time is 40 [ms], and the processing computer waits for another 60 [ms] before receiving the data for the next transaction. Therefore, transactions begin every 100 [ms]. For the second transaction, the image processing time is different from the first transaction owing to the

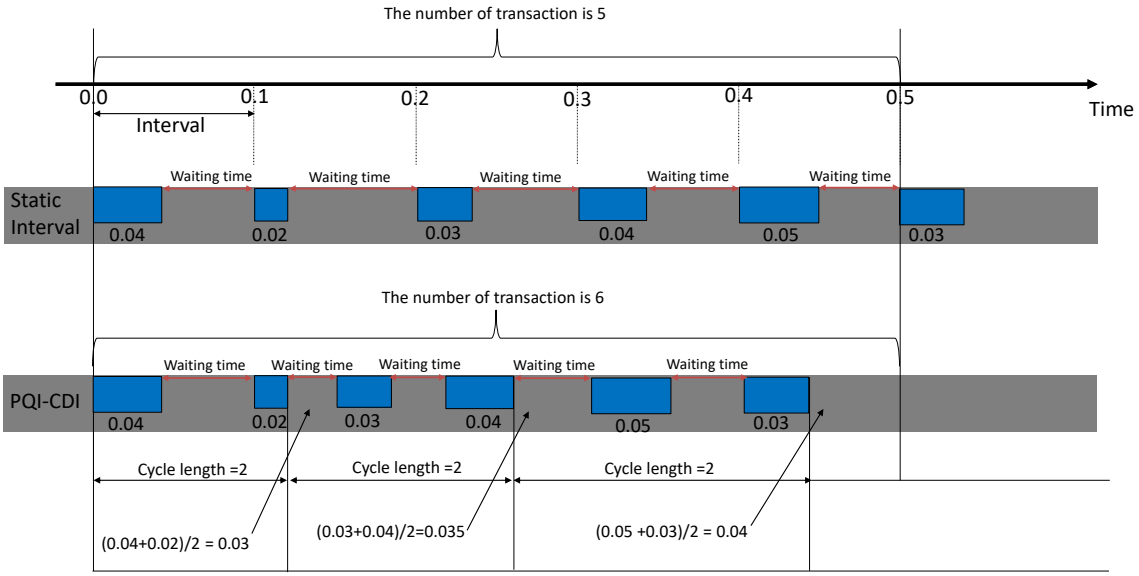


Figure 3.4: Transaction rate under the conventional method (static transaction interval) vs. the PQI-CDI Method

fluctuation of the processing power and is 20 [ms]. In this case, the transactions do not overlap, and the transaction rate is given by the inverse value of the transaction interval, which is 10.

The lower part of the figure shows the situation under the PQI-CDI method. In the PQI-CDI method, the processing computer calculates the average transaction time of the past C_n $(n = 1, \cdots, N)$ transactions for each camera device n, and sets the average value as the new transaction interval. To provide a simple example, we set the cycle length to two transactions (C_n=2). The processing computer requests to change the transaction interval at the end of each cycle. For example, for a case in which the first transaction time lasts 40 [ms] and the second last 20 [ms], the average transaction time is 30 [ms]. This value is then sent to the camera to influence the next transaction interval. From 0 to 500 [ms], in this example, the number of transactions under the PQI-CDI method is six. Thus, the transaction rate is $6000/500 = 12$ and is larger than that under the static interval.

3.4 Evaluation of the Transaction-rate Improvement Method

3.4.1 Evaluation Setup

In this subsection, we explain the evaluation setup.

3.4.1.1 Evaluation Parameters

In this evaluation, we assume the application described in Subsection 2.3.2 and use the parameters shown in Table 2.1. These parameters are similar to those used in Chapter 2, but we explain them again. The 'Input Bandwidth' is the input communication bandwidth of the processing computer, whose bandwidth is fairly shared among the data sources. The 'Output Bandwidth' is the output communication bandwidth of each data source. We set these parameters considering their reality. The 'Original Data Amount' is the amount of the original data that are processed in each transaction. We get this value as similar to Chapter 2. To simplify the evaluation results, we set the same data amount for all data items. The 'Processing Time Ratio' is the processing time divided by the data amount.

We set the same values for all $PProb_{n,p}(t)$ $(p = 1, \cdots, Q-1)$, i.e., $PProb_{n,p}(t) = FProb^{1/N}$. The meaning of the symbols $PProb_{n,p}(t)$ and $FProb$ is described in Subsections 2.3.3 and 2.5.2. We assume that the data for each quality is constructed by the first approach explained in Clause 2.4.2.3 as similar to Chapter 2. We simulate the stream processing system for 300 [s] to get the transaction time.

3.4.1.2 Performance Indexes

The primary evaluation item for PQI-CDI is the transaction rate and the transaction time. A drawback of adopting the dynamic interval is that there would be a fluctuation of the intervals between transactions which are not preferable for visual applications. To investigate this, we calculated the average jitter for all transactions which is represented as

$$\frac{\sum_{n=1}^{N}\sum_{t=2}^{T_n}|i_{n,t}-i_{n,t-1}|}{\sum_{n=1}^{N}(T_n-1)} \tag{3.2}$$

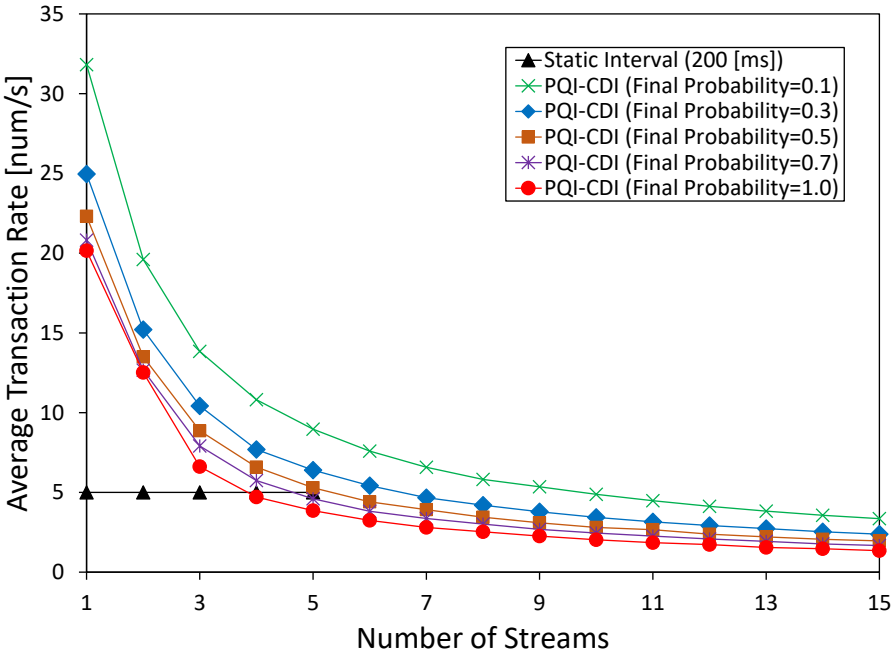


Figure 3.5: Average transaction rates and the number of streams

where N is the number of streams, T_n is the number of transactions executed in the n-th stream, and $i_{n,t}$ is the transaction interval from the previous transaction for t th transaction in the n th stream. A smaller average jitter indicates that the transaction intervals are less fluctuated.

3.4.2 Influence of Number of Streams

We measured transaction rates by changing the number of streams to investigate the effectiveness of our proposed PQI-CDI method.

In this experiment, we set the cycle length $C_n = 5$ because the jitter of the intervals was stable as shown in the next section (3.4.3). We set the number of quality levels to five, and the initial transaction interval is 200 [ms]. Figure 3.5 shows that the results of the average transaction rate changing the number of data streams. The horizontal axis is the number of streams, and the vertical axis is the average transaction rate.

The transaction rate under the PQI-CDI method decreases as the number of streams increases because the data amount that the processing computer communicates with the stream data sources increases. When the number of streams is small, the PQI-CDI method achieves a higher average transaction rate than the one with a static interval because the

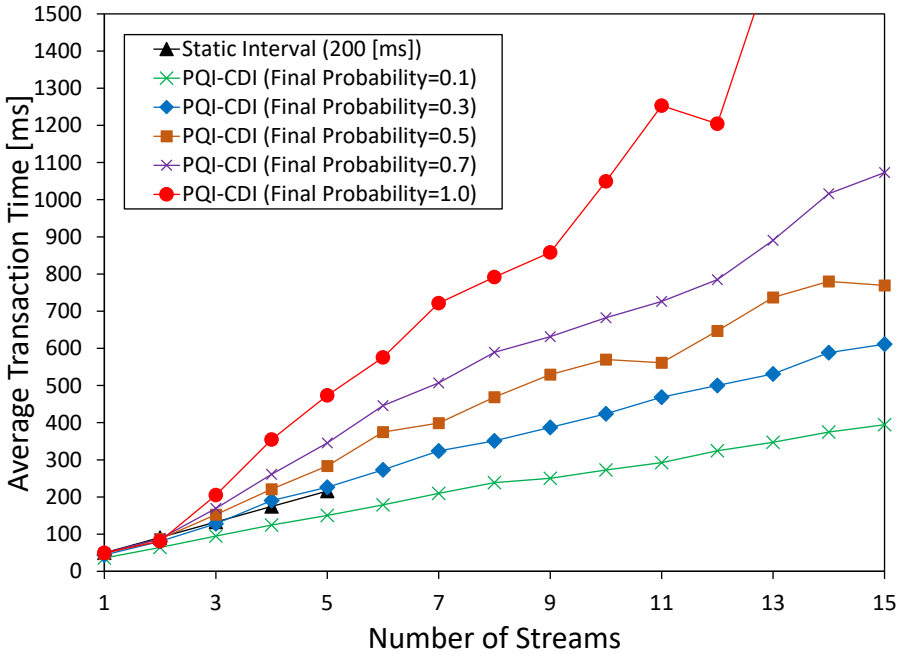


Figure 3.6: Average transaction time and the number of streams

network and the processing computer have extra capacity to improve the transaction rate compared with that of the case in which the transaction interval is fixed to 200 [ms]. The PQI-CDI method exploits this extra capacity by changing the interval dynamically. The average transaction rate increases as the final probability decreases because the average data to be transmitted from the data sources decreases with the final probability and the extra capacity increases. The static interval ends at a point where the number of the streams is five because the transaction times diverge when the number of data streams is greater than five.

Figure 3.6 shows the average transaction time. The horizontal axis is the number of streams, and the vertical axis is the average transaction time.

In the cases of other final probabilities, the average transaction time is longer than the static interval. This is because the updated transaction interval is given by the average transaction time in the previous cycle. However, the transaction time fluctuates for the reason that the processes probabilistically proceed to the processes for higher quality data and the transaction time can be longer than the updated transaction interval. Then, the transactions overlap and the transaction time increases. However, in the case that the final probability is 0.1, the PQI-CDI method provides a shorter average transaction time than

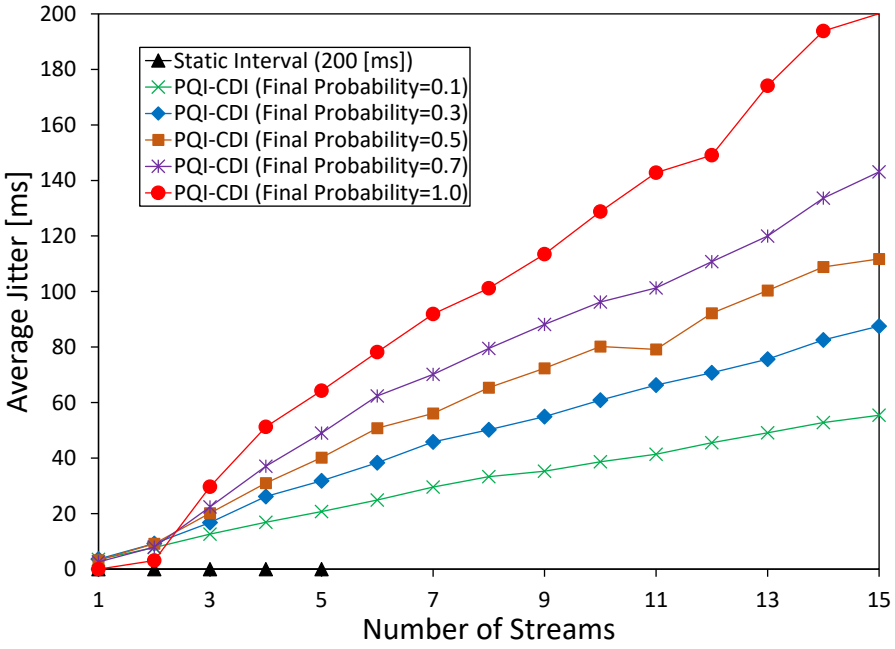


Figure 3.7: Average jitter and number of streams

the static interval because the network and the processing computer have sufficiently extra capacity to improve the transaction time. The line for the static interval stops at the point when the number of streams is five. The reason is the same as the result of the transaction rate and because the transaction times diverge when the number of data streams is greater than five.

Figure 3.7 shows the average jitter of the transaction intervals calculated by the formula (3.2). A smaller value indicates a less fluctuated transaction interval. The horizontal axis is the number of streams, and the vertical axis is the average jitter of transaction intervals.

In the static interval (200 [ms]) setting, as we can expect, the jitter values are zero. However, the the evaluation result for static interval ends at a point where the number of the streams is five because the transaction time is saturated.

The results that correspond to the PQI-CDI method show jitter values greater than zero because the intervals change dynamically. The jitter tends to increase as the number of data streams increase. This is because the data amount that is transmitted from all the data sources increases with the number of streams and the maximum transaction time lengthens. Therefore, the range of the transaction time becomes wider as the number of streams increases. As the transaction interval is given by the average transaction time in

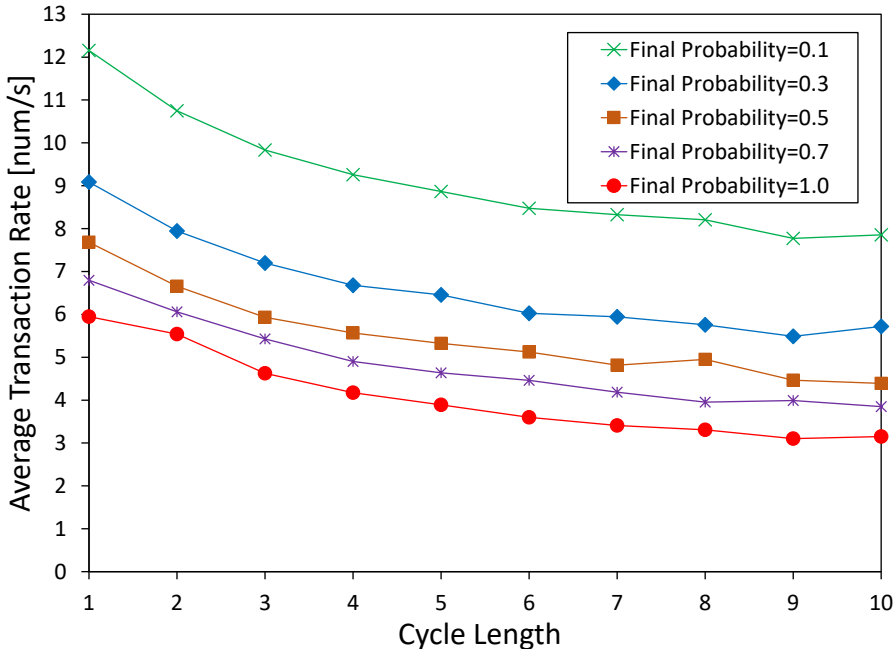


Figure 3.8: Average transaction rate and cycle length

the previous cycle under the PQI-CDI method, a wider range of transaction time gives a wider range of transaction intervals. Therefore, the jitter generally increases as the number of streams increases.

Moreover, the jitter increases with the final probability. The reason is similar to the reason that the jitter increases as the number of streams increases. The data amount that is transmitted from all the data sources increases with the final probability. Therefore, the range of the transaction interval increases with the final probability, and thus the jitter increases.

We can see the jitter is small when the final probability is 1.0 and the number of streams is one or two. When the final probability is getting closer to 1.0, the transaction time becomes stable as long as the load of the processor is small because the probability to process all quality levels in PQI increases. Because the load for the processing computer is rather small when the number of streams is less than three under the parameters used in the simulations, the average jitter became small.

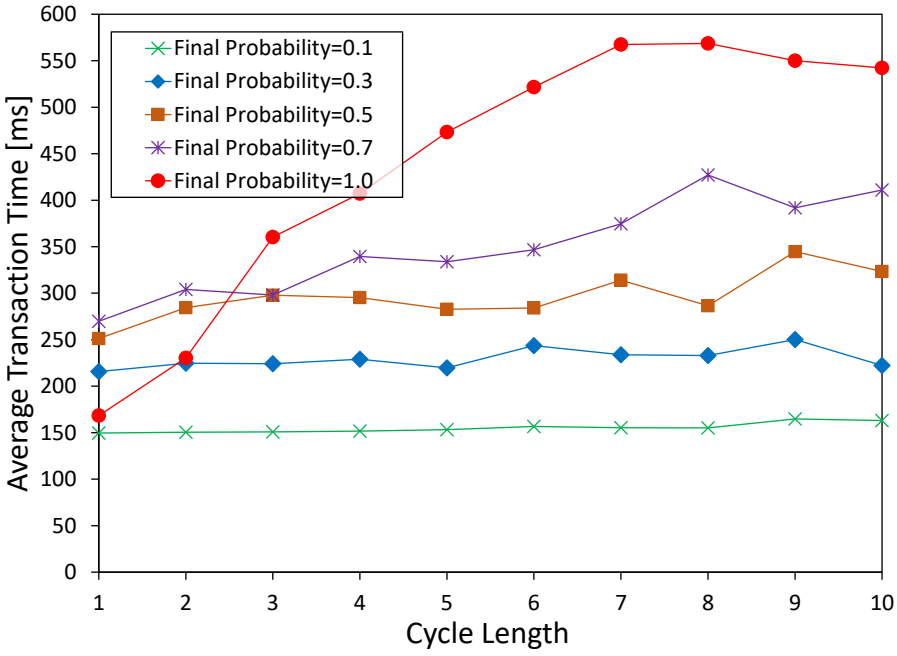


Figure 3.9: Average transaction time and cycle length

3.4.3 Influence of Cycle Length

In the PQI-CDI method, the cycle length influences performance. We investigated the influence changing the cycle length, $C_n(n = 1, \cdots, N)$, under different final probabilities. In this experiment, the cycle lengths for all streams are the same. The initial transaction interval is 200 [ms], the number of data streams is five, and the number of quality levels is five as example values.

Figure 3.8 shows the average transaction rate. The horizontal axis is the cycle lengths, and the vertical axis is the average transaction rate.

We can see that a shorter cycle length gives a higher average transaction rate because the interval is adjusted frequently with a shorter cycle length.

Figure 3.9 shows the average transaction time. The horizontal axis is 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 is longer because more transactions proceed to the final quality under a higher final probability. We can see that the average transaction times having lower final probabilities give shorter average transaction times. The average transaction time tends to increase a

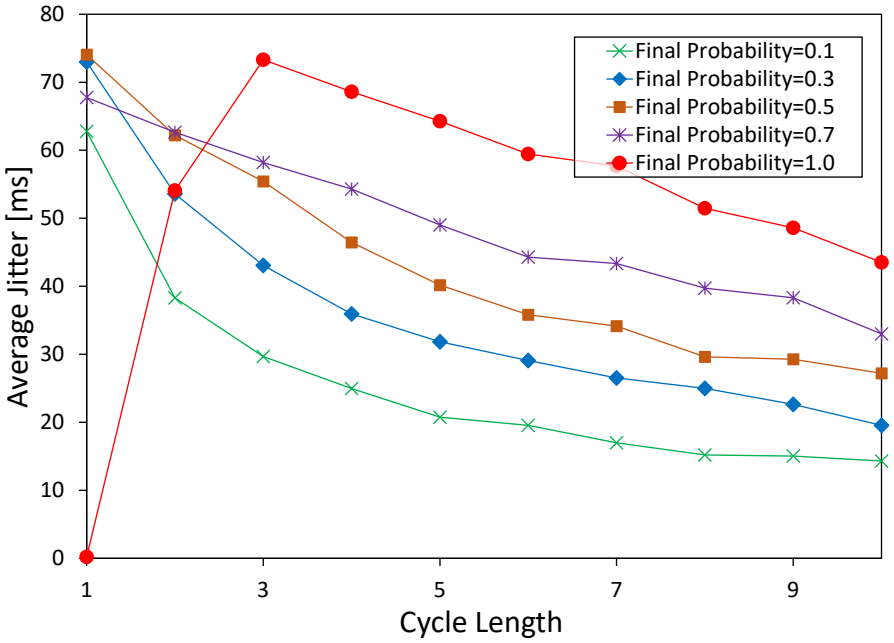


Figure 3.10: Average jitter and cycle length

little bit with the cycle length because the transaction interval does not change for a long period. This causes some transactions to overlap, and it lengthens the communication time and the processing time. Figure 3.10 shows the average jitter of the intervals under different final probabilities changing the cycle length. The jitter tends to decrease as the cycle length increases because the longer the cycle is, the less the intervals change frequently. We can see the average jitter is smaller than 50 [ms] when the cycle length is larger than 5 and the final probability is less than 0.7.

On the other hand, when the computation load is high, a longer cycle length increases the jitter value. This is because it takes time to converge the transaction interval in PQI-CDI when the computation load is high. In the simulation, the above mentioned situation occurred when the final probability is 1.0. When the final probability is 1.0 and the cycle length is 1, the average jitter is small because the transaction interval converges to the stable transaction time within a small number of interval changes. However, when the cycle length is two and three, the average jitter becomes large because PQI-CDI required several interval changes to converge to the actual transaction time.

As a whole, we confirmed that the PQI-CDI could alleviate the fluctuation of the transaction intervals in stream data processing by adopting an appropriate cycle.

3.5 Conclusion

The transaction rate is one of the main factors that need to be considered to improve the performance of stream data processing applications. To improve transaction rates, we proposed the PQI-CDI method, which dynamically changes the transaction interval under the PQI approach. In the PQI-CDI method, the processing computer changes the transaction intervals to be the same length as the average transaction time of the previous cycle. Moreover, the PQI-CDI method adopts the PQI approach to reduce communication time and processing time.

Our evaluation results revealed that the PQI-CDI method can achieve a higher transaction rate alleviating the fluctuation in the dynamic transaction intervals under the appropriate cycle length parameter settings.

In our assumed environment, the complexity and the priority of the processes are uniforms. However, in the actual environment, they vary depending on the applications. A potential extension of the PQI-CDI method is considering such heterogeneity of stream data processes. If the processing computer assigns computing resources according to the priority, the transaction rate may be improved. The current PQI-CDI method does not consider multi-processor environments that enable executing multiple stream data processes simultaneously on one or more processing computers. In this environment, the processing time is not changed until the number of simultaneous data processes reaches the number of processors of the stream data processes. Similar to the future work for the transaction time improvement in Chapter 2, the process assignment algorithms to improve transaction rate in such an environment is our future work.

Chapter 4

Implementation of Video Surveillance System with the PQI-CDI Method

4.1 Introduction

We first explain the background of the research in this chapter.

4.1.1 Background

In Chapter 3, we proposed the PQI-CDI method which improved the transaction time and the transaction rate for stream processing systems. In this chapter, we explain the implementation of a video surveillance system incorporated with the PQI-CDI method.

Our target application is perpetrator detection as described in Section 1.1. For instance, a video surveillance system in an office is installed to detect perpetrators in the image data recorded by surveillance cameras. Surveillance cameras are deployed in various places and are used worldwide [93–97].

Many researchers have examined the video surveillance systems with their recent proliferation. As discussed in Section 1.1, the main performance indexes are the transaction time and the transaction rate. To improve the transaction time and the transaction rate, various systems have been developed in [98–100]. However, the processing computer in the existing systems collects only the original quality data. By collecting the original quality data only when they are needed for processing, the average data amount of data collections is reduced. Therefore, we proposed the PQI approach and its extended PQI-CDI method in Chapters 2 and 3.

We used our developed simulator for the evaluation of our proposed approach because we could easily and rapidly test various situations by changing the evaluation parameters for the simulator. The simulator models the system shown in Section 1.1 and can simulate its behaviors. We set the parameters considering the average values in real situations and revealed the performance improvements established by the PQI-CDI method.

Although our developed simulator can simulate the behaviors of the modeled system, it is unclear whether an implemented system that adopts the PQI-CDI method give better performances than the systems without the PQI-CDI method. This is because it is unrealistic to create the perfect model of a specific implemented system that can give exactly same performances in the simulator. It is impossible to create a model that can reflect the performances of all video surveillance systems because the detail of the implementations depends on their provider. For example, the service provider has the choice of how to proceed the processes for video stream data. Also, he/she has the choice of how to implement the communication between the camera devices and the processing computer. Differences between the procedures in the simulator and those in real situations result in different performances (Subsection 4.1.2 discussed the details). Therefore, it is important for our research to investigate the differences between our developed simulator and an implemented system, and show the effectiveness of the proposed method in actual situations.

This chapter aims to investigate the differences in the results obtained in the simulations using the developed simulator and the experiments using an implemented system. We implement a video surveillance system incorporated with the PQI-CDI method. In the implemented system, three camera devices are used to serve as the stream data sources, and a laptop computer functions as the processing computer. We measure the transaction time, the transaction rate, and the jitters of the transaction intervals under real situations using our implemented system. The contributions of this chapter are summarized as follows:

(1) An implementation of the PQI-CDI method, which improves the transaction time and the transaction rate for stream data processing systems compared with the systems without the PQI approach. It is difficult to implement a video surveillance system with the PQI-CDI method because such systems have never been implemented. For the implementation, we need to design the system architecture (shown in Figure 4.5) and develop several software modules explained in Section 4.3. Especially, it is difficult to

develop the software module that produces the data needed to get qth quality data because it is required to search the delimiters for each quality in image files. There are no such existing software modules.

(2) An investigation of the differences in the results of the PQI-CDI method obtained using the developed simulator and the implemented system.

The remaining chapter is organized as follows. In Section 4.2, we introduce the related work. In Section 4.3, we explain the system design for the implementation of the video surveillance system with the PQI-CDI method. The implementation details are presented in Section 4.4, and experimental results are presented in Section 4.5. Finally, the chapter concludes in Section 4.6.

4.1.2 Performance Differences Caused by Actual Implementation

We focus on two aspects that cause the performance differences between those given by our developed simulator and by our implemented system.

First, our developed simulator does not strictly simulate the detailed design of how to proceed the processes for video stream data such as the processes on the stream data sources (the camera devices for surveillance systems) and the processes for starting/finishing the communications between them and the processing computer. This is because these procedures depend on implementations. If we modeled them strictly in the simulator, the simulation results would show a specific performance of an implementation.

Second, our developed simulator assumed that the communication channels for the camera devices and the processing computer were separated. Thus the processing computer receives the data for each transaction in parallel. In the surveillance systems in which the processing computer receives each quality data sequentially, the performances given by the simulator can be different from those of the implemented systems.

In this chapter, we investigate the differences caused by the above points. If our implemented system is similar to the system modeled by the simulator in detail, it is natural that the performances under the system are the same as those under the simulator. Simulators generally have some parameters for their configurations. For the reason of the errors of these parameters against actual situations, the absolute values of the performances obtained by simulators differ from those obtained by implemented systems although the tendencies of the changes in the values are similar. Therefore, we can further

confirm the effectiveness of our proposed approach if the implemented system shows better performances than the conventional approach even when it is not similar to the system modeled by the simulator.

4.2 Related Work for Implementation of Surveillance Systems

In this section, we first introduce several works for the implementation of surveillance systems. Our implemented system in this chapter adopts the PQI-CDI method which changes the transaction interval for the transaction rate improvement. Hence, after that, we introduce several systems considering transaction intervals and our work related to the implementation.

4.2.1 Surveillance Systems

H.264 is a well-known video encoding scheme to ensure a high video quality with a limited communication bandwidth. A surveillance system using H.264 was implemented in [101]. The authors evaluate the video quality and the bandwidth consumption. Although the authors confirm the reduction of the data amount transmitted from the camera devices to the viewer machines, they do not aim to reduce the time required to detect objects in the video surveillance systems.

A scheme to reduce the bandwidth consumption of video surveillance systems was proposed in [102]. In [102], the camera devices do not transmit data to the processing computer unless intrusions are detected. The processing computer verifies whether the intrusion actually occurres and send a notification to the system manager only when the intrusion occurres. This scheme can reduce the bandwidth consumption between the camera devices and the processing computer.

A simple approach to reduce the time required to detect objects is to limit the target area in the image to detect objects. In [103], the system identifies the background by using a background subtraction technique and reduces the object detection time by omitting the background area from the target area. Another approach in [104] set a threshold to clarify the background area. The method regards the pixels for which the difference from the previous frame image is less than the threshold as the background area. A larger threshold

yields a larger background area because the pixels thatare even largely different from that of the previous frame image are regarded as the background area. This method is similar to our proposed system in that the background is updated (progressive collection of the remaining data) when an object is detected. However, in the PQI-CDI method, the transaction interval is dynamically changed to enhance the transaction rate.

4.2.2 Systems considering Transaction Intervals

Most of the existing methods to modify the transaction interval adopt the approach in which the processing computers requested a change in the transaction intervals for the camera devices after a fixed number of transactions [105–109]. However, excessively short transaction intervals cause high computational loads on the processing computers, decreasing the transaction rates.

In [110], we proposed the PQI approach, which was explained in Chapter 2. In video surveillance systems using the approach, the camera devices produces several data having different quality levels. The lowest quality data are always transmitted to the processing computer. The processing computer progressively collects higher quality data only when it is needed. Furthermore, we proposed a method to improve the transaction rate in [111], which was explained in Chapter 3. The processing computer dynamically changes the transaction interval considering the transaction times of several past transactions. Although the simulator for the evaluation can simulate the behaviors of the modeled system, it is unclear whether an implemented system actually gives better performances than the systems without the proposed method. In this chapter, we investigate the differences between the results obtained by the simulator and that obtained by our implemented system [112].

4.3 Design of a Video Surveillance System

In this section, we explain the design for the implemented video surveillance system.

4.3.1 System Architecture

Figure 4.1 shows the system architecture. Some camera devices and a processing computer are connected to a computer network. The camera devices and the processing computer

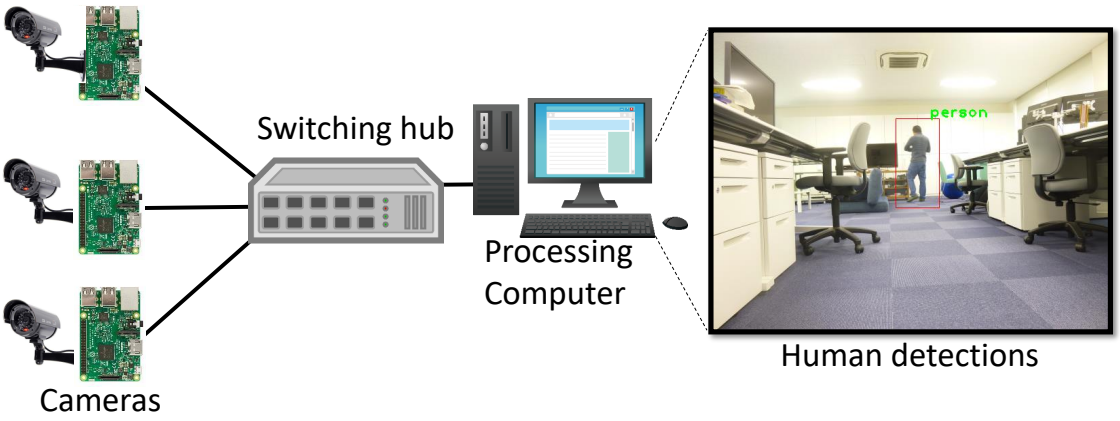


Figure 4.1: System architecture design

can communicate with one another. The computational resources of the camera devices are lower than that of the processing computer because we assume that compact devices serve as the camera devices such as Raspberry Pi devices equipped with the camera modules. The processing computer can be a laptop or a desktop computer, among other alternatives.

The camera devices obtain image data and produce image data that have different quality levels. For example, the Raspberry Pi devices obtain the JPEG image data from the equipped cameras. Notably, in the progressive JPEG format, the image data has several qualities (known as scans). Figure 4.2 shows an example of the JPEG image data having different qualities. $q_1, \cdots, q_{10}$ represents the quality levels. The lowest quality data corresponds to the smallest amount of data. These produced data are temporarily stored in the memory of the camera devices. The processing computer collects the stored image data from camera devices and executes image processing tasks such as object detection.

The camera devices and the processing computer adopt the PQI-CDI method to enhance the transaction time and rate. The details of the PQI-CDI method was explained in Chapter 3. The camera devices cyclically transmit the lowest quality data and transmit higher quality data only if required by the processing computer. The processing computer changes the transaction intervals of each stream data every time when a predetermined number of transactions finished.

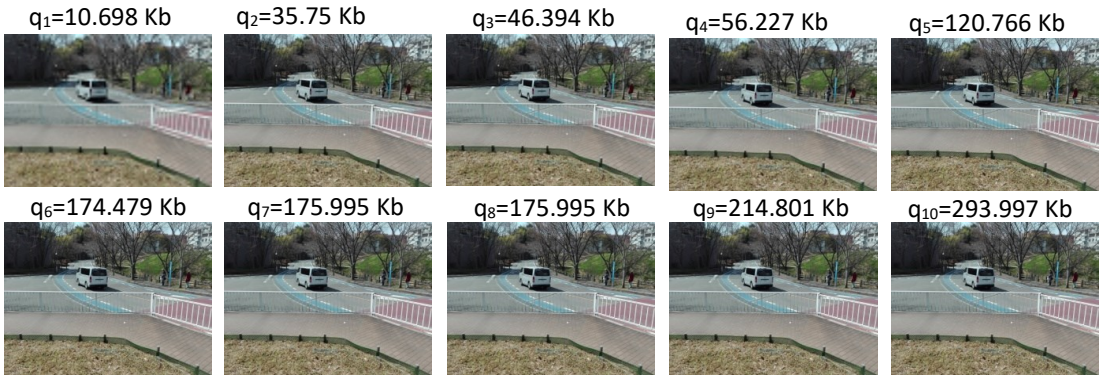


Figure 4.2: Example images with different quality levels

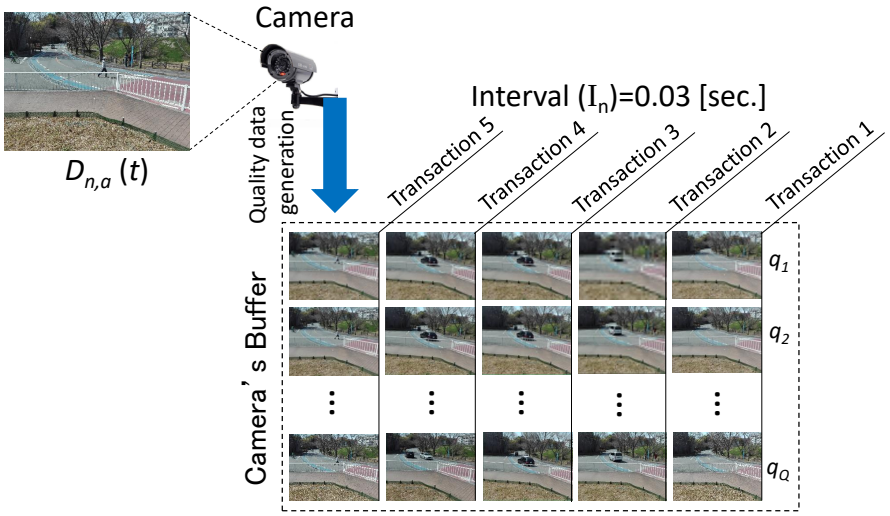


Figure 4.3: Generation image of each quality data in the cameras

4.3.2 Process Flow for Camera Devices

The detailed process flow of camera devices was described in Chapters 2 (Figures 2.2 and 2.3) and 3 (Figure 3.2). In this chapter, we briefly explain the process flow for camera devices designed for our implementation.

Figure 4.3 shows an image of the generation of each quality data in the camera devices. The transactions at the camera device n ($n = 1, \cdots N$) arise every time when a predetermined transaction interval elapses. Each camera device obtains $D_{n,a}(t)$ which is the original video frame image data of the camera device n at the t th transaction. The subscript a refers to the original data. After obtaining $D_{n,a}(t)$, each camera constructs $D_{n,q}(t)$ ($q = 1, \cdots, Q$) from $D_{n,a}(t)$ and temporarily stores the data to its storage. $D_{n,q}(t)$

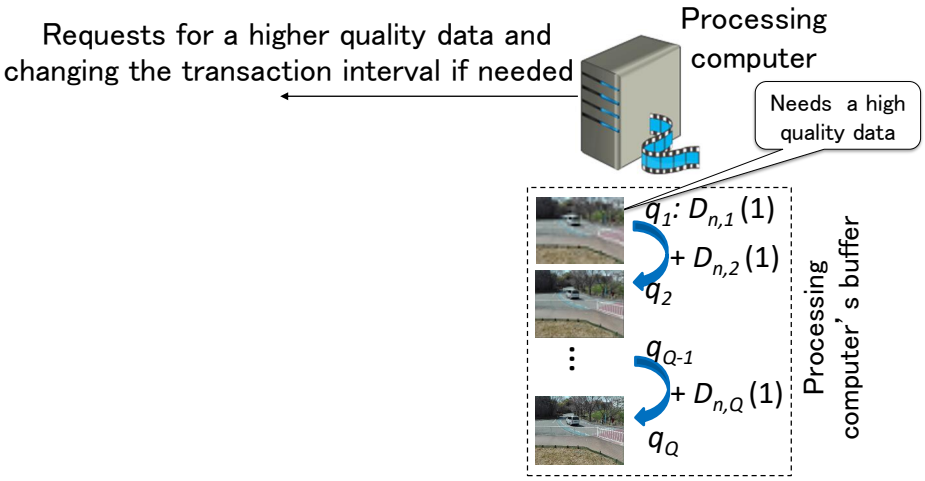


Figure 4.4: Processing image of the processing computer

represents the data needed to construct the qth quality data in combination with $D_{n,p}(t)$ $(p = 1, \cdots, q-1)$.

After generating $D_{n,q}(t)$ on each camera device, it transmits $D_{n,1}(t)$ to the processing computer and deletes $D_{n,1}(t)$ from its storage. After the transmission, the camera device waits for a request for higher quality data for the tth transaction. If the camera device receives the request for the qth quality data, it transmits $D_{n,q}(t)$ to the processing computer and deletes it from the buffer.

4.3.3 Process Flow for Processing Computers

Details of the process flow for the processing computers were presented in Chapters 2 (Figure 2.4) and 3 (Figure 3.3). In this chapter, we briefly explain the process flow for the processing computer designed for our implementation. Figure 4.4 illustrates the processing image of the processing computer. When the processing computer receives $D_{n,r}(t)$ $(r = 2, \cdots, Q)$, it starts generating the image data having the rth quality by combining it with $D_{n,s}(t)$ $(s = 1, \cdots, r-1)$. The first quality data is $D_{n,1}(t)$ itself and the processing computer does not need to combine it with other data. After the generation, the processing computer attempts to detect human faces in the image data by executing the object detection process. If faces are detected, the processing computer requests $D_{n,r+1}(t)$ to the camera device n. Otherwise, the tth transaction completes and the processing computer waits for the first quality data for the next transaction to be transmitted from the

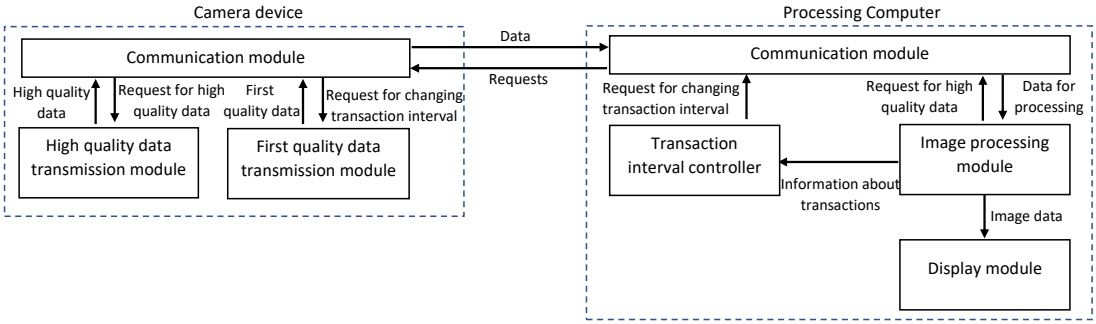


Figure 4.5: Software architecture of the implemented system

camera devices.

In the PQI-CDI method, the processing computer updates the transaction interval after a cycle finishes. The cycle for the camera device n includes C_n transactions. If C_n transactions are completed for the camera device n, the processing computer calculates the new transaction interval and sends the message to change the transaction interval with the new value to the camera device n.

4.3.4 Software Modules

Figure 4.5 shows the software modules. The system incorporates two types of installed software: one on the processing computer, and the other on the camera devices. We explain each module as follows.

4.3.4.1 Camera Devices

The software module for the camera devices consists of three parts.

- **Communication Module:** The communication module of each camera device mainly controls the communication with the processing computer. The module first creates the communication session with the processing computer. The communication between the camera device and the processing computer is performed on the created communication session. After the module created the communication session, it starts transmitting the stream data to the processing computer. In cases where the communication session breaks down unintentionally, the communication module tries to establish the handshaking again. The address for the processing

computer is designated to the camera devices by the user or the stream processing system manager.

When the communication module gets data from the first quality data transmission module or the high quality data transmission module in the program, it transmits the received data to the processing computer.

In the cases that the module receives a request for changing the transaction interval, it passes the request to the first quality data transmission module. In the cases that the module receives a request for higher quality data, it passes the request to the high quality data transmission module.

- **First Quality Data Transmission Module:** The first quality data transmission module controls the transmission of the first quality data. When a transaction starts, the module obtains the original image data from the camera and temporarily stores it in the buffer. After that, the module produces the data needed for getting qth quality data, $D_{n,q}(t)$ $(q = 1, \cdots Q)$, and stores it in the buffer. The transactions start every time when the transaction interval elapses.

  When the first quality data transmission module has produced the first quality data, $D_{n,1}(t)$, the module passes the data to the communication module and then the communication module transmits it to the processing computer.

  When the first quality data transmission module gets a request for changing the transaction interval, the module changes the transaction interval to the designated value.

- **High Quality Data Transmission Module:** The high quality data transmission module controls the transmission of higher quality data. When the module gets a request for higher quality data from the communication modules, it obtains the requested data from the buffer and passes it to the communication module.

4.3.4.2 Processing Computer

The software module for the processing computer consists of four parts.

- **Communication Module:** The communication module of a processing computer mainly controls the communication with the camera devices. The module starts

waiting for the creation of the communication session from camera devices when the system starts running. When the communication session with a camera device is created, it keeps the communication session for communicating with it. In cases where the communication session breaks down unintentionally, the module releases the communication session because another new communication session will be created by the camera device if needed.

When the communication module gets the request for changing transaction interval from the transaction interval controller, it transmits the request with the new transaction interval to the camera device.

In the cases that the communication module receives data from the camera devices, it passes the data to the image processing module.

- **Transaction Interval Controller:** The transaction interval controller manages the transaction intervals of each camera device based on the PQI-CDI method.

  When the controler receives the information about transactions from the image processing module, it counts up the number of the transactions from the start of the cycle (c_n in Clause 3.3.2.2) to check whether the number reaches the cycle length, C_n or not. If the number reaches C_n, the module calculates the new transaction interval and sends the request for changing the transaction interval to the communication module. The information about the transaction times needed to calculate the new transaction interval (transaction time and camera ID) is included in the information sent from the image processing module.

- **Image Processing Module:** The image processing module processes the received image data according to the user designated processes.

  When the module receives the data for processing, it starts executing the processes to the data and judges the necessity of higher quality data. If higher quality data is needed to proceed the transaction, the image processing module sends the request for it to the communication module. If the higher quality data is not needed or the received data is the original (highest) quality data, the transaction finishes. Moreover, the module sends the information about transactions to the transaction interval controller every time when a transaction finishes.

To check the behavior of the system, the image processing module passes the image data to the display module.

- **Display Module:** The display module receives image data from the image processing module and displays the data to the screen of the processing computer.

4.4 Implementation

In this section, we explain our implementation of a video surveillance system.

4.4.1 Equipment

This section describes the equipment used for the implementation. Figure 4.6 shows the devices used in the implementation. Three Raspberry Pi devices with camera modules and a laptop computer are connected to a network hub. An NTP (network time protocol) server is connected to the network for the synchronization of the times among the processing computers and the stream data sources. Time synchronization is required to measure the transaction time. The devices can communicate with one another via the computer network. The desktop computer functions as the processing computer. Table 4.1 lists the specifications of the implemented system.

4.4.2 Programs

We created two softwares. One is the software for the camera devices, which runs on each Raspberry Pi device in the system. The program consists of three parts as described in Clause 4.3.4.1. The Raspberry Pi devices do not originally contain a function to record different quality data. To produce image data that have different qualities, a function embedded in OpenCV (version 3.2) is adopted to convert the original image data to the progressive JPEG format. The converted progressive JPEG images have 10 scans. The processing computer can produce image data that have several qualities by combining some scans. For example, by combining five scans, the image data can be regarded to consist of two qualities.

Another software is for the processing computers, which runs on the laptop computer in the system. The program consists of four parts as described in Clause 4.3.4.2. The

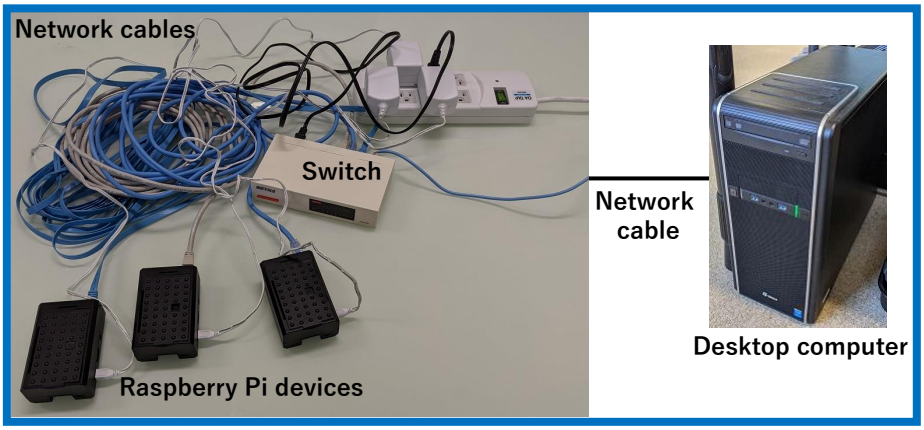


Figure 4.6: Devices used in the implementation

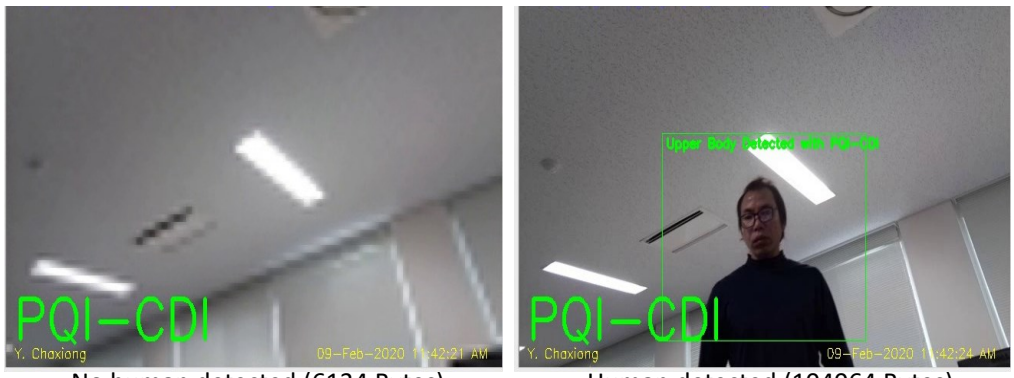
Table 4.1: Specifications of the implemented system

Items	Details
Recording computer	Raspberry Pi 3, 1 [GB] Memory
Camera device	Raspberry Pi NoIR Camera Module V2.1
Processing computer	Intel Core i7-1065 (1.3 [GHz] quad-core), 32 [GB] Memory
Network	100BASE-TX/1000BASE-T

image processing is performed by OpenCV. For the detection of human faces, we used the function 'Haar Feature-based Cascade Classifier', which uses Haar-like features for human face detection. We confirmed that human faces can be detected also in the lowest quality image data although the image is unclear.

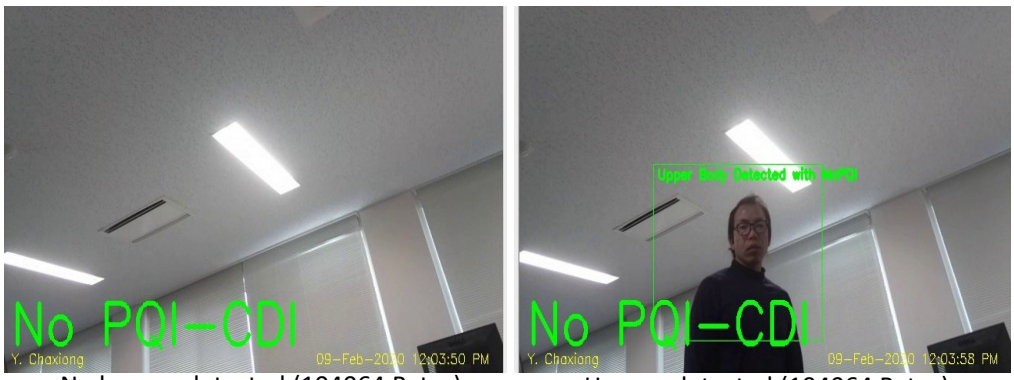
4.4.3 Sample Images

Figure 4.7 shows an example of images displayed under the PQI-CDI method. The left-sided image shows the case where the processing computer does not detect humans. In this case, the processing computer does not request a higher quality data. On the other hand, the right-sided image shows the case where the processing computer detects humans. In this case, the processing computer requests the data with the highest quality to obtain high-resolution image data. Therefore, the resolution of the left-sided image is



No human detected (6134 Bytes) Human detected (104064 Bytes)

Figure 4.7: Example images under the PQI-CDI method



No human detected (104064 Bytes) Human detected (104064 Bytes)

Figure 4.8: Example images when the PQI-CDI method is not used

lower than that of the right-sided image.

Figure 4.8 shows example images displayed under the conventional approach. Each camera device transmits its original image data to the processing computer without producing other quality data. The processing computer executes the human face detection processes against the clear image, which has the highest quality. The figure shows that each image shows a clear image even when no face is detected. In this case, the data sizes for each image are large compared with the case under the PQI-CDI method.

4.5 Evaluation of the Implemented System

To investigate the differences between the simulated performances and the real performance in this section, we compare the simulation results and actual results obtained using the

implemented system.

4.5.1 Evaluation Environments

We measured the performances of our implemented system changing the cycle length and the final probability. If we use live videos for measurements, it is impossible to strictly control final probabilities because the values depend on the probabilities that human faces are recorded in the videos. Therefore, we stored the images recorded by each camera device to its storage and used the images for getting the evaluation results under different cycle lengths. We measured the performances when the number of the camera devices is one and three. Based on the setting for the simulation in previous chapters, we set the bandwidth between the camera devices and the processing computer to 10 [Mbps] by using the QoS controller function of the Windows operating system. To get the performances using our developed simulator, we use the same parameter values shown in Tables 2.1. We ran the implemented system for one minute to get each plot in the results and calculated the average value. The other detail of our implemented system is explained in Subsection 4.4.1

We compare the performances of our implemented system for the cases with and without using the PQI-CDI method. Under the PQI-CDI method, the processing computer dynamically adjusts the transaction interval and adopts the PQI approach. Since some conventional surveillance systems dynamically change the transaction interval, as the comparison method, we use the method that does not adopt the PQI approach and dynamically adjusts the transaction interval every time a fixed number (cycle length) of transactions finishes. We denote the method by *No PQI-CDI* method in the following sections. The No PQI-CDI method is a special case of the PQI-CDI method where the number of the data qualities is one.

4.5.2 Transaction Time

Transaction time is one of the main performance indexes for surveillance systems. In the PQI-CDI method, the processing computer changes the transaction interval of each data source every time when the number of the finished transactions reaches the value of the cycle length. Therefore, we measured the transaction time by changing the cycle length to evaluate the performance of our implemented system.

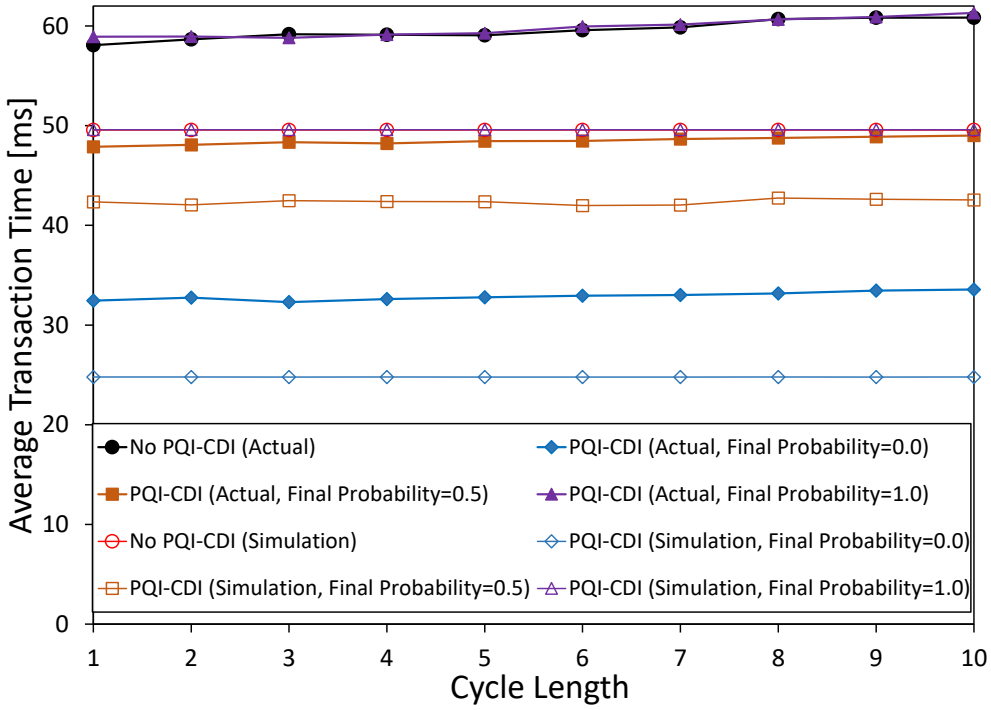


Figure 4.9: Average transaction time when one camera is used

Figure 4.9 shows the average transaction time when the number of the camera devices is one. The horizontal axis is the cycle length and the vertical axis is the average transaction time. 'Actual' in the legend indicates the average transaction time of our implemented system and 'Simulation' indicates the average transaction time measured by our developed simulator.

In both the actual case and the simulation case, the average transaction time under the conventional No PQI-CDI method is longer than that under our proposed PQI-CDI method when the final probability is 0.0 or 0.5. This is because the PQI-CDI method reduces the average data amount to be transmitted from the camera devices. Since the average data amount decreases with the final probability, the average transaction time under a lower final probability is shorter. When the final probability is 1.0, the average transaction time under the PQI-CDI method is almost the same as that under the No PQI-CDI method. This is because the data amount transmitted and processed is almost the same. The average transaction times in both the actual case and the simulation case do not change largely even when the cycle length changes. This is because the transaction time does not change largely even when the cycle proceeds to the next since the transaction

interval of each cycle, which is given by the average transaction time of the previous cycle, is almost the same as the transaction time in the cycle. Thus, the transaction time is not largely influenced by the cycle length when the number of the camera devices is one. The average transaction time increased as the cycle length got longer in the simulation results in Chapter 3 because the number of the stream data sources was five in the chapter.

The average transaction time in the actual case is longer than that in the simulation case under each final probability. This is because the simulator does not strictly simulate the communication and the processing procedures implemented in our system. In the implemented system, the camera devices grab the images, convert them to the progressive JPEG format, and create the data for each quality. After that, they transfer the data to the communication buffer for the transmission and wait for starting it. On the processing computer side, it moves the received data in the communication buffer to the main memory and constructs each quality data. Although our developed simulator simulates the processing time for each quality data, it ignores the above procedures. It takes some time for executing them in the implemented system, and thus the average transaction time in the actual case is longer than that in the simulation case. When the number of the camera devices is three, the transactions are interfered by other video stream data. Thus, the transaction time is predicted to depend on the cycle length because there is a possibility that the transaction time changes when the cycle proceeds to the next one. Therefore, we measured the transaction time when the number of the camera devices is three. Figure 4.10 shows the result. In both the actual and the simulation cases, as we expected, the average transaction time changes along with the cycle length. This is because the transaction interval does not change for a long period under a longer cycle length, and thus several transactions overlap when the transaction interval is shorter than the transaction time, causing a longer transaction time.

Similar to the result of one camera device, the average transaction time under the No PQI-CDI method is longer than that under our proposed PQI-CDI method when the final probability is 0.0 or 0.5. The average transaction time in the actual case is longer than that of the simulation cases under each final probability. This is the same reason as the result when the number of the camera devices is one. That is because our developed simulator did not strictly simulate the communication and the processing procedures implemented to our system.

When the cycle length is small (approximately smaller than three), the average

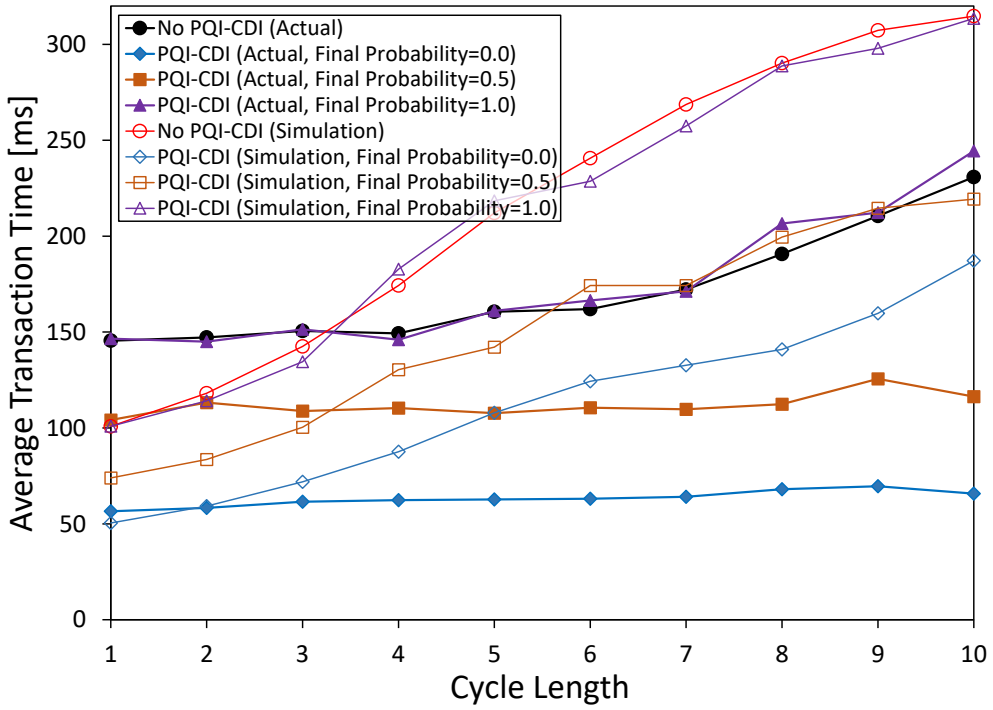


Figure 4.10: Average transaction time when three cameras are used

transaction time in the actual case is longer than that in the simulation case under each final probability. This is the same reason as the result when the number of the camera devices is three. That is because our developed simulator did not strictly simulate the communication and the processing procedures implemented to our system.

In the simulation case, the average transaction time increases further than that in the actual case as the cycle length increases. This is because the simulated communication time is longer than that of the implemented system when the transactions overlap with others. Our developed simulator assumed that communication channels for the camera devices and the processing computer were separated, and thus the processing computer received the data for each transaction in parallel. On the other hand, the processing computer received each quality data sequentially in our implemented system. Since it takes a longer time to receive data by parallel communications, the communication times in the simulation cases are longer than those in the actual cases when the transactions overlap. Since a longer cycle length gives a higher probability that the transactions overlap, the transaction times further increases in the simulation cases than those in the actual cases.

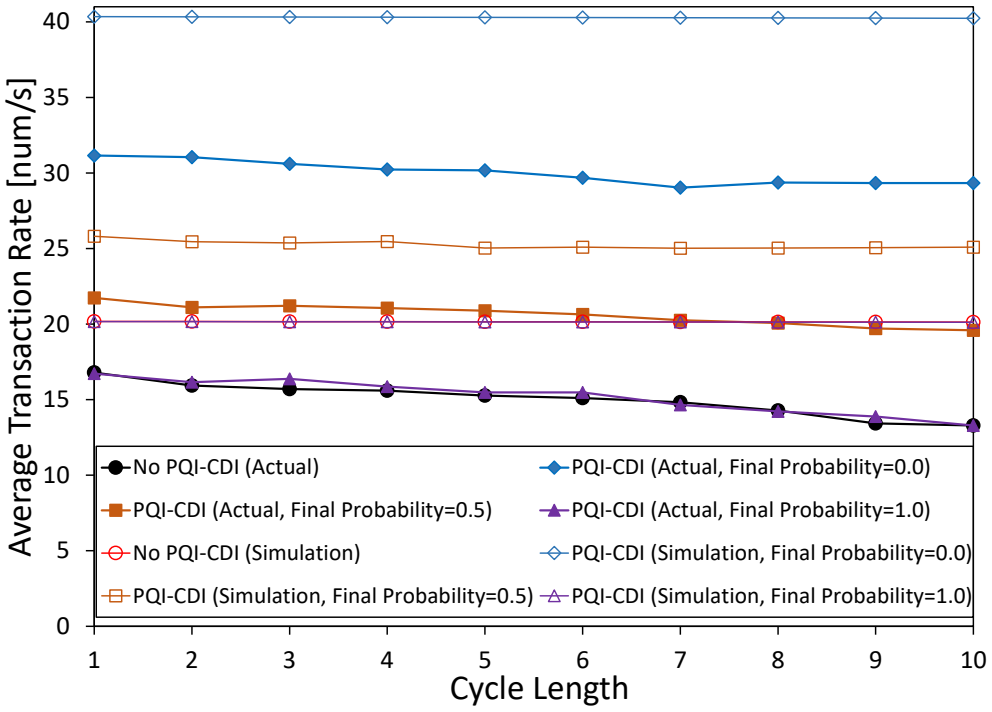


Figure 4.11: Average transaction rate when one camera is used

For example, in the case that the number of the camera devices is one, the average transaction time under the PQI-CDI method is 48 [ms] when the final probability is 0.5 at minimum (cycle length is 1). The average transaction time under the No PQI-CDI method and the same final probability is 58 [ms]. That is, our implemented system achieves 17% shorter average transaction time in this situation. In the case that the number of the camera devices is one, the average transaction time under the PQI-CDI method is 101 [ms] when the final probability is 0.5 at minimum (cycle length is 1). The average transaction time under the No PQI-CDI method and the same final probability is 147 [ms] (31% reduction).

4.5.3 Transaction Rate

The transaction rate is one of the main performance indexes for surveillance systems. To evaluate the performance of our implemented system, we measured the transaction rate changing the cycle length.

Figure 4.11 shows the average transaction rate when the number of the camera devices

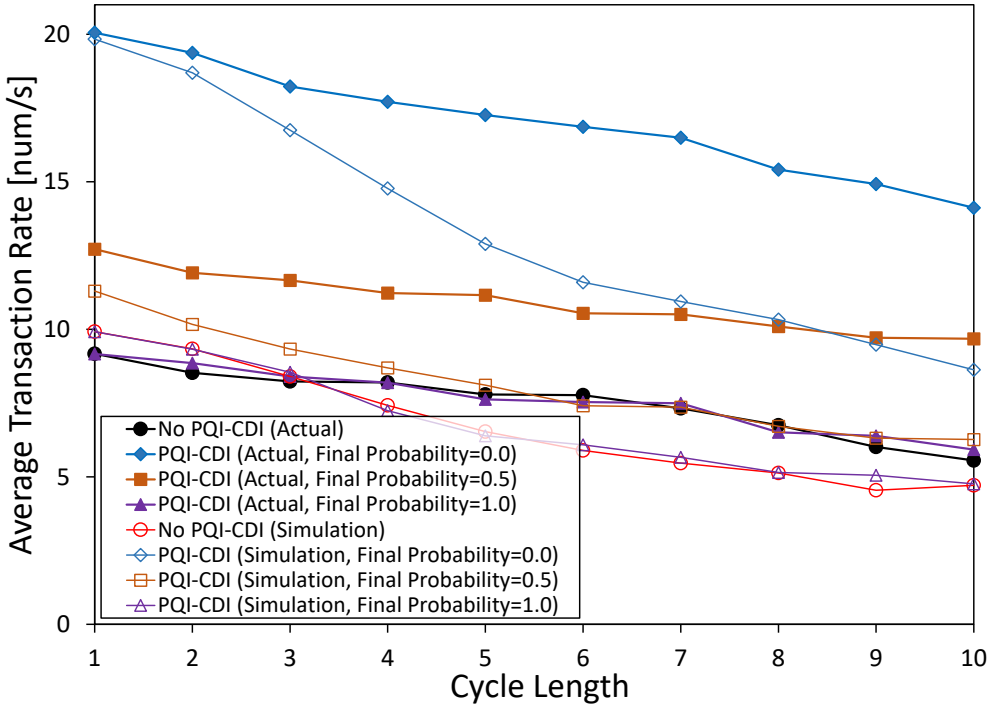


Figure 4.12: Average transaction rate when three cameras are used

is one. The horizontal axis is the cycle length and the vertical axis is the average transaction rate. The legends and the other experimental configurations are the same as the previous subsection.

In the actual case and the simulation case, the average transaction rate under the conventional No PQI-CDI method is smaller than that under our proposed PQI-CDI method when the final probability is 0.0 or 0.5. This is because the PQI-CDI method attempts to adjust the transaction interval so that the transactions do not overlap with others. When the final probability is 1.0, the average transaction rate under the PQI-CDI method is almost the same as that under the No PQI-CDI method because the data amount transmitted and processed is the same. The average transaction rates in both the actual case and the simulation case do not change largely even when the cycle length changes. The reason is the same as the reason that the transaction times when the number of the camera devices is one do not change largely.

Figure 4.12 shows the average transaction rate when the number of the camera devices is three. The average transaction rates in both the actual case and the simulation case decrease as the cycle length increases because the transaction interval does not change

for a long period under a longer cycle length as similar to the reason explained in the previous subsection. The decreasing rate in the simulation case is larger than that in the actual case because the processing computer received the data for each transaction in parallel in the simulator. The detailed reason is the same as the case for the transaction time and was explained in the previous subsection.

The average transaction rate under the No PQI-CDI method is smaller than that of our proposed PQI-CDI method when the final probability is 0.0 or 0.5 as similar to the result of one camera device. The average transaction rate in the actual case is smaller than that in the simulation case under each final probability. This is because the simulator did not strictly simulate the communication and the processing procedures implemented to the actual system.

For example, in the case where the number of the camera devices is one, the average transaction rate under the PQI-CDI method is 22 [num/s] when the final probability is 0.5 at maximum (cycle length is 1). The average transaction rate under the conventional No PQI-CDI method and the same final probability is 17 [num/s]. That is, our implemented system achieves 29% higher average transaction rate in the evaluation situation. In the case that the number of the camera devices is three, the average transaction rate under the PQI-CDI method is 12 [num/s] when the final probability is 0.5 at maximum (cycle length is 1). The average transaction time under the conventional No PQI-CDI method and the same final probability is 9.2 [num/s] (30% improvement).

4.5.4 Jitter of Transaction Intervals

We evaluated the jitter of transaction intervals in Chapter 3 as a drawback of dynamic interval. We measured the jitters in our implemented system.

Figure 4.13 shows the average jitter when the number of the camera devices is one. The horizontal axis is the cycle length and the vertical axis is the average jitter. The legends and the other experimental configurations are the same as the previous subsection.

In the actual cases, the average jitter under the conventional No PQI-CDI method is longer than that under our proposed PQI-CDI method when the final probability is 0.0 or 0.5 (except for the case that the cycle length is one) because the values of the transaction intervals under these final probabilities are smaller than that under the No PQI-CDI method as shown in Figure 4.9. The average jitter when the final probability is 0.5 and the

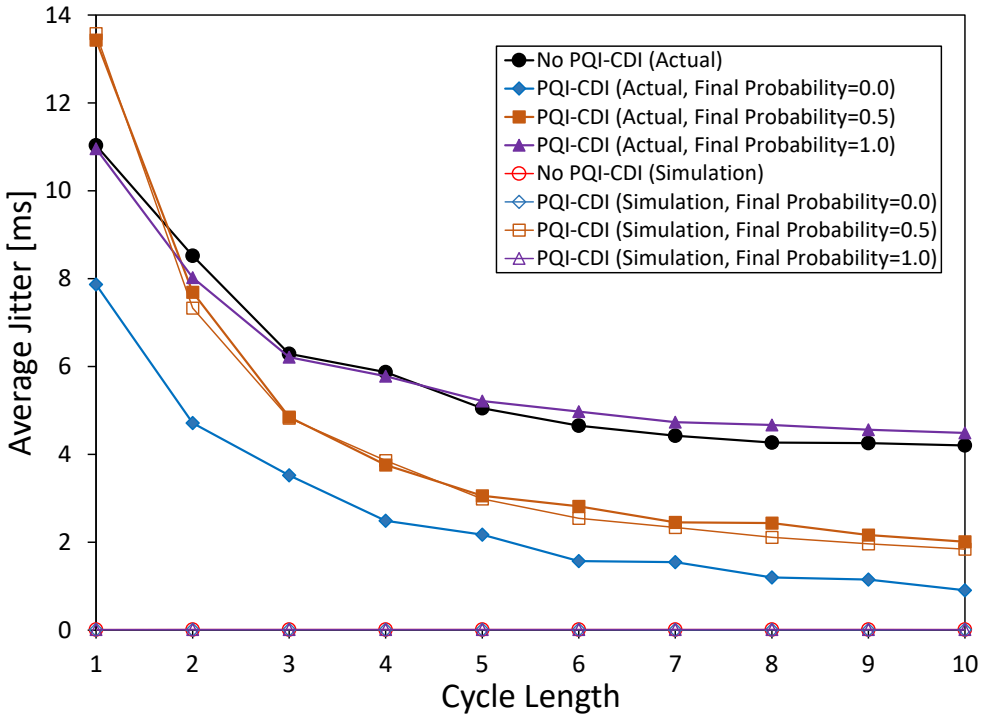


Figure 4.13: Average jitter when one camera is used

cycle length is one under the PQI-CDI method is larger than the jitter under other final probabilities. This is because the transaction time when the processing computer does not request the second quality data is shorter than the case when it requests the second one. Since the range of the transaction time is large compared with the range when the final probability is 0.0 or 1.0, the average jitter increases. The average jitter in the actual case decreases as the cycle length increases because a longer cycle length less changes the transaction interval.

In the simulation case, the average jitter is almost zero except for the case where the final probability is 0.5. This is because the number of the camera devices is one and the transaction interval of each cycle is almost the same as the transaction time in the cycle. Therefore, the transaction time does not change even when the cycle proceeds to the next one, and thus the transaction interval does not change. The average jitters increased in the results in Chapter 3 because the number of the stream data sources was five in the chapter. The average jitter when the final probability is 0.5 in the simulation case is larger than those under other final probabilities because the transaction time changes depending on whether the processing computer requests the second quality data or not.

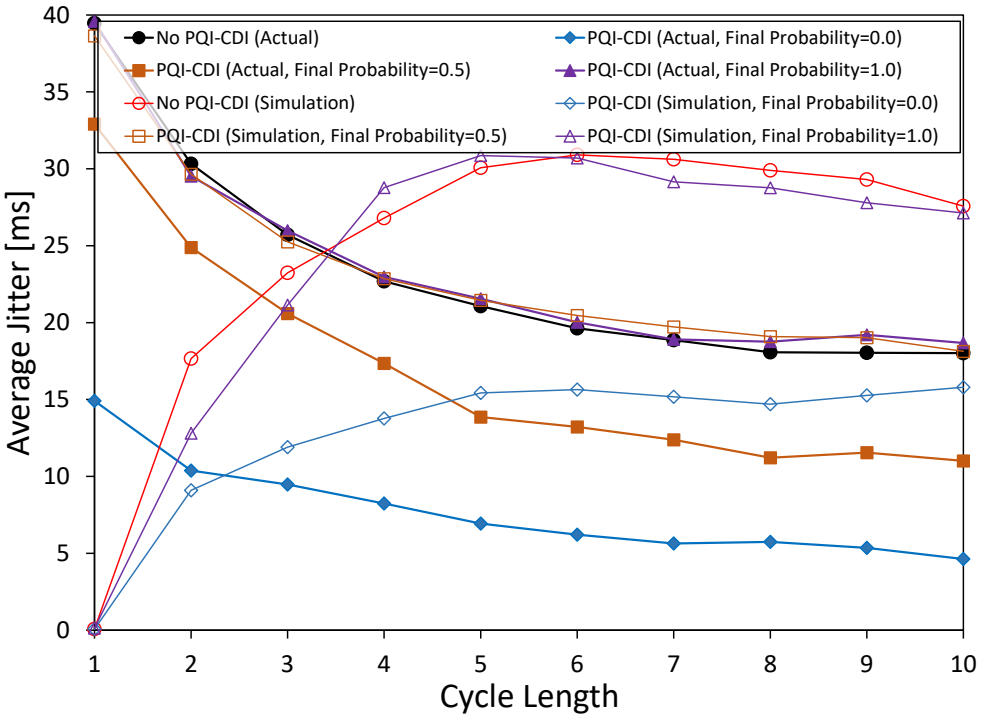


Figure 4.14: Average jitter when three cameras are used

Figure 4.14 shows the average jitter when the number of the camera devices is three. In the actual case, the average jitter under the conventional No PQI-CDI method is longer than that under our proposed PQI-CDI method when the final probability is 0.0 or 0.5 because the values of the transaction intervals under these final probabilities are smaller than that under the No PQI-CDI method. The average jitter in the actual case tends to decrease as the cycle length increases because a longer cycle length less frequently changes the transaction intervals as similar to the result of one camera device.

In the simulation case, the average jitter under the No PQI-CDI method increases when the cycle length is small (approximately less than six). Also, the average jitter under the PQI-CDI method increases when the cycle length is small and the final probability is 0.0 or 1.0. This is because a small number of interval changes leads the convergence of the transaction interval when the cycle length is 1.0 in these cases. However, several interval changes are required to converge the transaction interval when the cycle length gets larger. The average jitter converges to a certain value as the cycle length increases when the cycle length is large (approximately larger than five in these cases) because the transaction interval less changes. This phenomenon also appeared in the result when

the final probability is 1.0 in Figure 3.10. The converged average jitter when the final probability is 0.0 is smaller than that when the final probability is 1.0 because the value is smaller. When the final probability is 0.5, the average jitter does not increase even when the cycle length is small because the average jitter converges after several cycles are elapsed. Therefore, the average jitter in this case decreases along with the cycle length.

4.5.5 Discussion

Our implemented system has two different points from the system modeled by the simulator as explained in Subsection 4.1.2.

When the number of the camera devices is small (one camera device in the evaluation results), the influence caused by the parallel communication in the simulator is not so large. Therefore, the influence caused by the difference of transaction procedures leads the differences in the absolute performance values between our implemented system and our developed system as shown in Figures 4.9 and 4.11. However, we confirmed that the tendencies of the changes in the values under the implemented system, i.e., the performances under the PQI-CDI method are better than the conventional No PQI-CDI method when the final probability is less than 1.0, are similar to those under the simulator.

When the number of the camera devices is large (three camera devices in the evaluation results), the influence caused by the parallel communication in the simulator is also large. Therefore, the performances under the simulator depend on the cycle length although those under the implemented system does not depend on it as shown in Figures 4.10 and 4.12. However, we confirmed that the PQI-CDI method gives better performances than the No PQI-CDI method when the final probability is less than 1.0 even where the implemented system is not similar to the modeled system by the simulator.

4.6 Conclusion

The simulator we used in previous chapters did not strictly simulate the detailed design of how to proceed the processes for video stream data. Also, our developed simulator assumed that communication channels for the camera devices and the processing computer were separated. Therefore, in this chapter, we implemented a video surveillance system and investigated the differences in the results obtained using our developed simulator

and the implemented system. The implemented system adopted our proposed PQI-CDI method, which reduced the transaction time and improved the transaction rate. We measured the transaction time, transaction rate, and the jitters of the transaction intervals under the implemented system for the investigation.

We confirmed that our implemented system properly worked and realized the implementation of the PQI-CDI method. Moreover, we found that the performance of the implemented system were different from the simulated results in two points. First, the overheads caused by the differences between the procedures for the simulator and them in our implemented system deteriorated the performance. Second, the parallel communications between the processing computer and the camera devices in the simulator deteriorated the performances compared with the implemented system when the number of the camera devices is large. Although the simulation results were different from the results obtained by our implemented system, we confirmed that the PQI-CDI method can improve the transaction time and the transaction rate in actual situations.

One of the remaining issues in the chapter is adopting the developed system to the data other than video data. In this chapter, we developed a video surveillance system with the PQI-CDI method. However, the method does not depend on the contents of the data as far as we can construct several quality data such as LiDAR (the example scenario is described in Section 2.4). The amount of the data, the time required for processing them, or the probability to proceed to the processing of the next quality data depends on the content of the data. Hence, the performance investigations for other data are our future work. Another remaining issue is the consideration of communication errors such as data losses or disconnections. In the case where the processing computer cannot receive the requested data, it needs to skip the transaction or request the data again. This causes a lower application performance. However, our developed system does not consider such communication errors that can deteriorate the application performance.

Chapter 5

Conclusion

5.1 Concluding Remark

Stream data processing systems have attracted considerable research attention. In some systems, a processing computer processes stream data transmitted by remote stream data sources. Two key indexes could be considered to evaluate the performance of such stream data processing systems, namely, the transaction time and the transaction rate. A smaller transaction time and higher transaction rate correspond to an enhanced application performance. This study aimed at enhancing these performances and the following key conclusions were derived.

In Chapter 1, we explained the importance of stream data processing and identified the research issues. We explained the research objectives, related work, and the content of this research in the chapter.

In Chapter 2, we described our proposed PQI approach aimed to reduce the transaction time of stream data processing systems of which input data come from remote stream data sources. In the PQI approach, each stream data source constructs several data of that qualities are lower than the original quality, e.g., low-resolution image data. Only in the cases where higher quality data are needed for processing, the processing computer progressively collect them. By reducing the average data amount of data collections and for processing, the PQI method reduces the average transaction time. Our simulation evaluation revealed that our proposed method could reduce the transaction time while keeping the application performances compared to the conventional No PQI method for the cases that the final probability is small. Here, the final probability is the probability that the processing computer finally processes the original quality data in each transaction.

In Chapter 3, we described the PQI-CDI method aimed to improve the transaction rate by changing the transaction intervals dynamically under the PQI approach. In the PQI-CDI method, the processing computer changes the transaction intervals to be the same length as the average transaction time of the previous cycle. Moreover, the PQI-CDI method adopts the PQI approach to reduce communication and processing times. Our evaluation results revealed that the PQI-CDI method could achieve a higher transaction rate than a conventional approach (static transaction interval). However, when the cycle length was excessively very short, the processing computer would request too many interval changes, resulting in less fair and inconsistent transaction intervals. If the cycle length was too long, changing the transaction interval did not cause noticeable improvements.

In Chapter 4, we described the implementation of a video surveillance system with the PQI-CDI method and its performance evaluation in actual situations. In the implementation, three camera devices and one processing computer were connected via a local area network. We developed two software modules for the camera devices and the processing computer. We confirmed that our implemented system properly worked and realized the implementation of the PQI-CDI method. Moreover, we investigated the differences between the performances obtained by our developed simulator and those by our implemented system. We confirmed that the PQI-CDI method could improve the transaction time and the transaction rate in actual situations although the performances were not always similar to the simulation results.

Overall, the proposed methods based on the PQI approach could reduce the transaction time and improve the transaction rate in some situations. The target system of this study is stream data processing systems in which a processing computer processes stream data transmitted by remote stream data sources. Existing such systems did not consider the necessity of the original quality data that are obtained at the data sources. We opened up new vista to the research field and proposed a pioneer PQI approach focusing on this point. In the approach, the processing computer progressively collects higher quality data from the data sources only in cases when they are needed. The approach, which can be applied for recently attractive and worldwidely used abovementioned stream processing systems, brings a large contribution to our lives.

5.2 Future Work

Compared to the conventional approach, the PQI approach could reduce the transaction time under a low final probability, and the PQI-CDI method could achieve a higher transaction rate. However, certain challenges remain to be addressed.

In the targeted system in this study, one processing computer executes the stream data processing. One processing computer systems are easy to construct. If the system management organization owns some other processing computers or exploits virtual computers provided by cloud services, the computational resources can be enhanced by using multiple processing computers. In such cases, the transaction time can be further reduced by distributing the processing load that is concentrated on one processing computer in this sturdy to others. Therefore, efficient load distribution for the stream processing system in the cases that the remote stream data sources transmit their data to some processing computers is a future direction for our research.

In this study, we did not consider the power consumption both of the processing computer and the stream data sources. As introduced in Subsection 1.3.3, the stream processing system with the PQI approach can apply several power consumption reduction schemes. However, the transaction rate degrades by omitting several data transmissions and processes for the power consumption reduction. Therefore, reduction of the transaction time and the improvement of the transaction rate considering the power consumption is a future direction of this study.

Furthermore, we implemented a video surveillance system with three camera devices and one processing computer. They are connected with each other via a switching hub. However, recent video surveillance systems could equip more camera devices that are connected to the Internet due to the recent IoT trends. In such cases, the communication bandwidths between the camera devices and the processing computer are not stable. Thus, the transaction time tends to change largely. This causes inefficient adjustment of the transaction interval by the PQI-CDI method because the method uses the average transaction time as the new transaction interval. Therefore, the transaction rate improvement under unstable communication bandwidth situations is a future direction for our research.

Acknowledgements

Upon the completion of this dissertation, I would like to take this opportunity to express my sincerest gratitude to those who have done their best to support me.

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Shinji Shimojo, who not only accepted me as his student but also assisted me to overcome the limitation of my knowledge as well as providing me valuable advice and guidance throughout my doctoral research work.

I would like to express my deepest appreciation to Associate Professor Tomoki Yoshihisa at the Cybermedia Center, Osaka University, for his patient and enthusiastic guidance along with my studies. I would not be able to work on this challenging and interesting topic without his suggestions and I could also not have come this far without his supports. He continuously and convincingly conveyed a spirit of adventure and excitement to this study. His excellent advice based on his deep knowledge always provided a unique perspective and new meaning to this study.

I would like to thank Assistant Professor Tomoya Kawakami at University of Fukui and Dr. Yuuichi Teranishi at National Institute of Information and Communications Technology. This dissertation would not have been possible without their valuable advice and guidance. Also thank you for all their trust in my sometimes spontaneous and transient research ideas.

I would also like to acknowledge the committee members of my dissertation, Professor Toru Fujiwara and Professor Takahiro Hara. Their insightful and constructive comments considerably improved the quality of the dissertation.

I would like to express my appreciation to Professor Makoto Onizuka and Professor Yasuyuki Matsushita at the Department of Multimedia Engineering of the Graduate School of Information Science and Technology of Osaka University, and Professor Kaname Harumoto at Osaka University Institute for Datability Science for their innovative suggestions and warm student life support.

I wish to express my acknowledgment to JICA AUN/SEED-Net for awarding me the scholarship with the financial support for the doctoral degree within 3 years. Furthermore, I extend my sincere appreciation to Osaka University for giving me the great opportunity to do research in a warm and friendly environment.

My grateful acknowledgment goes also to all professors and supporting staffs in both Shimojo laboratory and IST office, who always help and give me guidelines and conveniences during the whole period of my study.

I would like to thank all the students at the Shimojo laboratory for helping me with some cases about my life in Japan, especially Japanese communication, and for all the fun we had. It was a great pleasure working with you all. I would like to recognize all my friends in the Laboratory for the enjoyable and stimulating atmosphere that they provide with their companion and friendship.

Last but not least, I would like to offer my hearty thanks to my family who always appreciate my every decision, share my good or bad times, and provide me with their greatest concerns and cares.

References

[1] W. K. Wong and K.T. Leow, “Wireless Webcam based Car Burglar Detection System,” in Proc. International Conference on Intelligent and Advanced Systems (ICIAS), pp. 1-4, June 2014.

[2] C. N. Bhagwat, Y. N. Krishnan, and K. Badrinath, “Cloud based Intruder Detection System,” in Proc. International Conference on Electronics and Communication Systems (ICECS), pp. 1244-1246, February 2015.

[3] J. Zhang and L. Yue, “The Design of Multi-Zone Wireless Burglar Alarm System,” in Proc. International Conference on Cyberspace Technology (CCT), pp. 469-473, November 2013.

[4] P. Şchiopu and A. Costea, “Design of Anti Burglar Alarm Systems Detection and Signaling,” in Proc. IEEE International Symposium for Design and Technology in Electronic Packaging (SIITME), pp. 111-115, October 2011.

[5] G. Long, “Design of Wireless Burglar Alarm based on Mcu,” in Proc. International Conference on Multimedia Technology, pp. 3032-3034, July 2011.

[6] P. Choorat, C. Sirikornkarn, and T. Pramoun, “License Plate Detection and Integral Intensity Projection for Automatic Finding the Vacant of Car Parking Space,” in Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pp. 1-4, June 2019.

[7] C. Ahn, B. Lee, S. Yang, and S. Park, “Design of Car License Plate Area Detection Algorithm for Enhanced Recognition Plate,” in Proc. International Conference on Computer Applications and Information Processing Technology (CAIPT), pp. 1-4, August 2017.

[8] A. Menon and B. Omman, "Detection and Recognition of Multiple License Plate from Still Images," in Proc. International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), pp. 1-5, December 2018.

[9] O. Khin, M. Phothisonothai, and S. Choomchuay, "License Plate Detection of Myanmar Vehicle Images Captured from the Dissimilar Environmental Conditions," in Proc. International Conference on Advanced Computing and Applications (ACOMP), pp. 127-132, December 2017.

[10] L. Li, S. Yoon, J. Liu, and J. Yi, "Multi-Scale Car Detection and Localization using Contour Fragments," in Proc. IEEE International Conference on Image Processing (ICIP), pp. 1609-1613, October 2014.

[11] H. Sarıbaş, H. Çevıkalp, and S. Kahvecıoğlu, "Car Detection in Images Taken from Unmanned Aerial Vehicles," in Proc. Signal Processing and Communications Applications Conference (SIU), pp. 1-4, July 2018.

[12] G. Li, X. Fang, K. Khoshelham, and S. O. Elberink, "Detection of Cars in Mobile Lidar Point Clouds," in Proc. IEEE International Conference on Intelligent Transportation Engineering (ICITE), pp. 259-263, October 2018.

[13] R. Cortés, X. Bonnaire, O. Marin, and P. Sens, "Stream Processing of Healthcare Sensor Data: Studying User Traces to Identify Challenges from a Big Data Perspective," Procedia Computer Science, Elsevier, Vol. 52, pp. 1004-1009, June 2015.

[14] S. Shao, J. Woo, K. Yamamoto, and N. Kubota, , "Elderly Health Care System Based on High Precision Vibration Sensor," in Proc. IEEE International Conference on Machine Learning and Cybernetics (ICMLC 2019), pp. 1-6, July 2019.

[15] A. De, P. Joao, B. Herfort, A. Brenning, and A. Zipf, "A Geographic Approach for Combining Social Media and Authoritative Data Towards Identifying Useful Information for Disaster Management," International journal of geographical information science, Taylor & Francis, Vol. 29, No. 4, pp. 667-689, Febrary 2015.

[16] M. Jahanian, T. Hasegawa, Y. Kawabe, Y. Koizumi, A. Magdy, M. Nishigaki, T. Ohki, K.K. Ramakrishnan, "DiReCT: Disaster Response Coordination with Trusted

Volunteers, IEEE International Conference on Information and Communication Technologies for Disaster Management (ICT-DM 2019), pp. 1-8, December 2019.

[17] X. Lu, C. Ye, J. Yu, and Y. Zhang, "A Real-Time Distributed Intelligent Traffic Video-Surveillance System on Embedded Smart Cameras," in Proc. International Conference on Networking and Distributed Computing, pp. 51-55, October 2014.

[18] N. Abbas and F. Yu, "Design and Implementation of A Video Surveillance System for Linear Wireless Multimedia Sensor Networks," in Proc. IEEE International Conference on Image, Vision and Computing (ICIVC), pp. 524-527, October 2018.

[19] P. Anghelescu, I. Serbanescu, and S. Ionita, "Surveillance System using IP Camera and Face-Detection Algorithm," in Proc. International Conference on Electronics, Computers and Artificial Intelligence (ECAI), pp. 1-6, October 2013.

[20] M. N. Saadat, H. Kabir, Z. A. Long, H. Sofian, and M. F. A. Zuhairi, "Efficient Face Detection and Identification in Networked Video Surveillance Systems," in Proc. International Conference on Ubiquitous Information Management and Communication (IMCOM), pp. 1-9, February 2020.

[21] Y. Ge, X. Liang, Y. C. Zhou, Z. Pan, G. T. Zhao, and Y. L. Zheng, "Adaptive Analytic Service for Real-Time Internet of Things Applications," in Proc. IEEE International Conference on Web Services (ICWS), pp. 484-491, July 2016.

[22] R. Bansod, S. Kadarkar, R. Virk, M. Raval, R. Rashinkar, and M. Nambiar, "High Performance Distributed In-Memory Architectures for Trade Surveillance System," in Proc. International Symposium on Parallel and Distributed Computing (ISPDC), pp. 101-108, June 2018.

[23] A. Akbar, G. Kousiouris, H. Pervaiz, J. Sancho, P. Ta-Shma, F. Carrez, and K. Moessner, "Real-Time Probabilistic Data Fusion for Large-Scale IoT Applications," in IEEE Access, Vol. 6, pp. 10015-10027, February 2018.

[24] J. A. Colmenares, R. Dorrigiv, and D. G. Waddington, "A Single-Node Datastore for High-Velocity Multidimensional Sensor Data," in Proc. IEEE International Conference on Big Data (Big Data), pp. 445-452, December 2017.

[25] F. Gao, Z. Huang, Z. Wang and S. Wang, "An Object Detection Acceleration Framework based on Low-Power Heterogeneous Manycore Architecture," in Proc. IEEE World Forum on Internet of Things (WF-IoT), pp. 597-602, December 2016.

[26] W. Huang, Y. Kang, and S. Zheng, "An Improved Frame Difference Method for Moving Target Detection," in Proc. Chinese Automation Congress (CAC), pp. 1537-1541, October 2017.

[27] S. Kameyama and Y. Miura, "High Performance of Moving-Object Detection by Gpgpu based on Pipelining," in Proc. IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), pp. 9-10, June 2017.

[28] S. Luo, W. Yao, Q. Yu, J. Xiao, H. Lu, and Z. Zhou, "Object Detection based on Gpu Parallel Computing for Robocup Middle Size League," in Proc. IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 86-91, December 2017.

[29] Y. Tajima, K. Ito, T. Aoki, T. Hosoi, S. Nagashima, and K. Kobayashi, "Performance Improvement of Face Recognition Algorithms using Occluded-Region Detection," in Proc. International Conference on Biometrics (ICB), pp. 1-8, June 2013.

[30] H. Cao, M. Brown, L. Chen, R. Smith, and M. Wachowicz, "Lessons Learned from Integrating Batch and Stream Processing using IoT Data," in Proc. International Conference on Internet of Things, Systems, Management and Security (IoTSMS), pp. 32-34, October 2019.

[31] Y. Lee, M. Lee, M. Lee, S. J. Hur, and O. Min, "Design of A Scalable Data Stream Channel for Big Data Processing," in Proc. International Conference on Advanced Communication Technology (ICAT), pp. 537-540, July 2015.

[32] D. Sun and S. Hwang, "DSSP: Stream Split Processing Model for High Correctness of Out-of-Order Data Processing," in Proc. IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), pp. 193-197, September 2018.

[33] A. Dey, K. Stuart, and M. E. Tolentino, "Characterizing the Impact of Topology on IoT Stream Processing," in Proc. IEEE World Forum on Internet of Things (WF-IoT), pp. 505-510, February 2018.

[34] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive Clustering for Dynamic IoT Data Streams," in IEEE Internet of Things Journal, Vol. 4, No. 1, pp. 64-74, February 2017.

[35] A. Nassar, A. Mostefaoui, and F. Dessables, "Improving Big-Data Automotive Applications Performance Through Adaptive Resource Allocation," in Proc. IEEE Symposium on Computers and Communications (ISCC), pp. 1-7, July 2019.

[36] Z. Guo, Y. Gao, Y. Jiang and G. Xue, "GRIB Parallel Design of Civil Aviation Meteorological Data Processing System," in Proc. International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), pp. 34-37, August 2015.

[37] K. B. Mistree, A. Dutt, and S. V. Kothiya, "Real Time Object Tracking for High Performance System using Gpgpu," in Proc. International Conference on Information Processing (ICIP), pp. 529-534, December 2015.

[38] B. Bosek, L. Horwath, G. Matecki, and A. Pawlik, "High Performance Gpgpu based System for Matching People in A Live Video Feed," in Proc. International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 448-454, October 2012.

[39] V. Alieksieiev, "One Approach of Approximation for Incoming Data Stream in IoT based Monitoring System," in Proc. IEEE Second International Conference on Data Stream Mining & Processing (DSMP), pp. 94-97, August 2018.

[40] H. Shao, J. Hsu, C. Ngo, and C. Kweon, "Progressive Transmission of Uncompressed Video over mmW Wireless," in Proc. IEEE Consumer Communications and Networking Conference, pp. 1-5, January 2010.

[41] C. Komar and C. Ersoy, "Detection Performance Improvement using Risk Assessment Framework," in Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 1888-1893, September 2010.

[42] L. Dao Thi Hue, D. T. Duong, and X. Hoangvan, "Hevc based Distributed Scalable Video Coding for Surveillance Visual System," in Proc. NAFOSTED Conference on Information and Computer Science, pp. 314-318, November 2017.

[43] R. Ge, Z. Shan, and H. Kou, "An Intelligent Surveillance System based on Motion Detection," in Proc. IEEE International Conference on Broadband Network and Multimedia Technology, pp. 306-309, February 2012.

[44] H. Kim and H. Lee, "A Low-Power Surveillance Video Coding System with Early Background Subtraction and Adaptive Frame Memory Compression," in IEEE Transactions on Consumer Electronics, Vol. 63, No. 4, pp. 359-367, November 2017.

[45] U. A. Agrawal and P. V. Jani, "Performance Analysis of Real Time Object Tracking System based on Compressive Sensing," in Proc. International Conference on Signal Processing, Computing and Control (ISPCC), pp. 187-193, September 2017.

[46] U. A. Agrawal and P. V. Jani, P. Liu, L. Zhao and Y. Ma, "Compressive Sensing of Multispectral Image based on Pca and Bregman Split," in Proc. IEEE International Geoscience and Remote Sensing Symposium, pp. 2558-2561, July 2013.

[47] L. Kong and R. Dai, "Object-Detection-based Video Compression for Wireless Surveillance Systems," in IEEE Multimedia, Vol. 24, No. 2, pp. 76-85, April 2017.

[48] L. Liu, Z. Li, and E. J. Delp, "Efficient and Low-Complexity Surveillance Video Compression using Backward-Channel Aware Wyner-Ziv Video Coding," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 19, No. 4, pp. 453-465, April 2009.

[49] A. Ortega and M. Khansari, "Rate Control for Video Coding over Variable Bit Rate Channels with Applications to Wireless Transmission," in Proc. International Conference on Image Processing, Vol.3, pp. 388-391, October 1995.

[50] O. D. Incel and B. Krishnamachari, "Enhancing the Data Collection Rate of Tree-based Aggregation in Wireless Sensor Networks," in Proc. IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 569-577, June 2008.

[51] F. Xhafa, V. Naranjo, S. Caballé, and L. Barolli, "A Software Chain Approach to Big Data Stream Processing and Analytics," in Proc. International Conference on Complex, Intelligent, and Software Intensive Systems, pp. 179-186, July 2015.

[52] G. Z. Papadopoulos, N. Pappas, A. Gallais, T. Noel, and V. Angelakis, "Distributed Adaptive Scheme for Reliable Data Collection in Fault Tolerant WSNs," in Proc. IEEE World Forum on Internet of Things (WF-IoT), pp. 116-121, December 2015.

[53] Y. Huang and S. Cheng, "Reduction Scheme for Sensor-Data Transmission on A Big Data Streaming Platform," in Proc. International Conference on Ubiquitous and Future Networks (ICUFN), pp. 244-249, August 2018.

[54] X. Peng, Y. He, and L.Tian, "Communication Reduction for Continuous Extreme Values Monitoring over Distributed Data Streams," in Proc. Workshop on Power Electronics and Intelligent Transportation System, pp. 181-187, September 2008.

[55] M. Ramachandra, "Optimization of the Data Transactions and Computations in IoT Sensors," in Proc. International Conference on Internet of Things and Applications (IoTA), pp. 358-363, January 2016.

[56] H. Kanzaki, K. Schubert, and N. Bambos, "Video Streaming Schemes for Industrial Iot," in Proc. International Conference on Computer Communication and Networks (ICCCN), pp. 1-7, August 2017.

[57] P. Zhao, W. Yu, X. Yang, D. Meng, and L. Wang, "Buffer Data-Driven Adaptation of Mobile Video Streaming over Heterogeneous Wireless Networks," in IEEE Internet of Things Journal, Vol. 5, No. 5, pp. 3430-3441, October 2018.

[58] J. Wu, C. Yuen, M. Wang, and J. Chen, "Content-Aware Concurrent Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks," in IEEE Transactions on Parallel and Distributed Systems, Vol. 27, No. 3, pp. 710-723, March 2016.

[59] S. Yu, T. Tian, J. Zhou, and H. Guo, "An Adaptive Packet Transmission Model for Real-Time Embedded Network Streaming Server," in Proc. International Conference on Audio, Language and Image Processing, pp. 848-853, July 2008.

[60] W. U. Rahman, Y. Choi, and K. Chung, "Performance Evaluation of Video Streaming Application over CoAP in IoT," in IEEE Access, Vol. 7, pp. 39852-39861, March 2019.

[61] J. Xu, Y. Andrepoulos, Y. Xiao, and M. Van Der Schaar, "Non-Stationary Resource Allocation Policies for Delay-Constrained Video Streaming: Application to Video over Internet-of-Things-Enabled Networks," in IEEE Journal on Selected Areas in Communications, Vol. 32, No. 4, pp. 782-794, March 2014.

[62] S. Molina-Giraldo, H. D. Insuasti-Ceballos, C. E. Arroyave, J. F. Montoya, J. S. Lopez-Villa, A. Alvarez-Mez, and G. Castellanos-Dominguez, "People Detection in Video Streams using Background Subtraction and Spatial-based Scene Modeling," in Proc. Symposium on Signal Processing, Images and Computer Vision (STSIVA), pp. 1-6, September 2015.

[63] S. V. Rao and M. Dakshayini, "Priority based Optimal Resource Reservation Mechanism in Constrained Networks for IoT Applications," in Proc. International Conference on Wireless Communications, Signal Processing and Networking (WISPNET), pp. 1228-1233, March 2016.

[64] O. Kulitsa, D. Okladnoy, V. Tverdokhleb, and A. Hahanova, "The Development Method for Evaluating the Saturation of Video Frame Blocks to Reduce The Processing Time of the Video Stream," in Proc. IEEE East-West Design & Test Symposium (EWDTS), pp. 1-3, January 2017.

[65] J. Lee, G. Yoon, and H. Choi, "Monitoring of IoT Data for Reducing Network Traffic," in Proc. International Conference on Ubiquitous and Future Networks (ICUFN), pp. 395-397, August 2018.

[66] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming Video over the Internet: Approaches and Directions," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, pp. 282-300, March 2001.

[67] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On Reducing IoT Service Delay Via Fog Offloading," in IEEE Internet of Things Journal, Vol. 5, No. 2, pp. 998-1010, April 2018.

[68] T. Buddhika and S. Pallickara, "NEPTUNE: Real Time Stream Processing for Internet of Things and Sensing Environments," in Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 1143-1152, May 2016.

[69] J. C. Beard and R. D. Chamberlain, "Use of Simple Analytic Performance Models for Streaming Data Applications Deployed on Diverse Architectures," in Proc. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 138-139, April 2013.

[70] Q. Hu, S. Paisitkriangkrai, C. Shen, A. Van Den Hengel, and F. Porikli, "Fast Detection of Multiple Objects in Traffic Scenes with A Common Detection Framework," in IEEE Transactions on Intelligent Transportation Systems, Vol. 17, No. 4, pp. 1002-1014, April 2016.

[71] Y. S. Reddy and K. K. Pattanaik, "A Reply Cache Mechanism to Reduce Query Latency of WSN in IoT Sensory Environment," in Proc. IEEE International Symposium on Nanoelectronic and Information Systems (INIS), pp. 38-42, December 2016.

[72] T. Theodorou and L. Mamatas, "Software Defined Topology Control Strategies for the Internet of Things," in Proc. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 236-241, November 2017.

[73] T. Theodorou and L. Mamatas, "CORAL-SDN: A Software-Defined Networking Solution for the Internet of Things," in Proc. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 1-2, November 2017.

[74] J. Navarro-Ortiz, P. Ameigeiras, J. M. Lopez-Soler, J. Lorca-Hernando, Q. Perez-Tarrero, and R. Garcia-Perez, "A QoE-Aware Scheduler for HTTP Progressive Video in OFDMA Systems," in IEEE Communications Letters, Vol. 17, No. 4, pp. 677-680, April 2013.

[75] W. Liu, L. Dong, and W. Zeng, "Motion Refinement based Progressive Side-Information Estimation for Wyner-Ziv Video Coding," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 12, pp. 1863-1875, December 2010.

[76] M. H. Taieb, J. Chouinard, K. Loukhaoukha, D. Wang, and G. Huchet, "Progressive Distributed Video Coding with Multiple Passes for Side Information Update," in Proc. International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), pp. 453-459, March 2012.

[77] Y. Shen, H. Cheng, J. Luo, Y. Lin, and J. Wu, "Efficient Real-Time Distributed Video Coding by Parallel Progressive Side Information Regeneration," in IEEE Sensors Journal, Vol. 17, No. 6, pp. 1872-1883, March 2017.

[78] S. Nazir, V. Stankovic, H. Attar, L. Stankovic, and S. Cheng, "Relay-Assisted Rateless Layered Multiple Description Video Delivery," in IEEE Journal on Selected Areas in Communications, Vol. 31, No. 8, pp. 1629-1637, August 2013.

[79] S. Yu, X. Chen, W. Sun, and Deping Xie, "A Robust Method for Detecting and Counting People," in Proc. International Conference on Audio, Language and Image Processing, pp. 1545-1549, July 2008.

[80] E. Tan and C. T. Chou, "A Frame Rate Optimization Framework for Improving Continuity in Video Streaming," in IEEE Transactions on Multimedia, Vol. 14, No. 3, pp. 910-922, June 2012.

[81] J. C. -. Ju, Y. Chen, and S. Y. Kung, "A Fast Rate-Optimized Motion Estimation Algorithm for Low-Bit-Rate Video Coding," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 7, pp. 994-1002, October. 1999.

[82] J. A. Colmenares, R. Dorrigiv, and D. G. Waddington, "A Single-Node Datastore for High-Velocity Multidimensional Sensor Data," in Proc. IEEE International Conference on Big Data (Big Data), pp. 445-452, December 2017.

[83] G. M. Dias, M. Nurchis, and B. Bellalta, "Adapting Sampling Interval of Sensor Networks using On-Line Reinforcement Learning," in Proc. IEEE World Forum on Internet of Things (WF-IoT), pp. 460-465, December 2016.

[84] U. A. Agrawal and P. V. Jani, "A Real-Time Object Detecting System using Compressive Sensing," in Proc. International Conference on Signal Processing and Communication (ICSPC), pp. 68-74, July 2017.

[85] M. Zeng, Z. Wei, H. He, and X. Yang, "Application of Improved Bhattacharyya Coefficient based Multi-Object Detection and Tracking Integrated Strategy," in Proc. International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp. 60-64, August 2018.

[86] A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya, "Object Detection Algorithms for Video Surveillance Applications," in Proc. International Conference on Communication and Signal Processing (ICCSP), pp. 0563-0568, April 2018.

[87] S. T. Ali, K. Goyal, and J. Singhai, "Moving Object Detection using Self Adaptive Gaussian Mixture Model for Real Time Applications," in Proc. International Conference on Recent Innovations in Signal Processing and Embedded Systems (RISE), pp. 153-156, October 2017.

[88] X. Zhang, "Research on Vehicle Object Detection in Traffic Video Stream," in Proc. IEEE Conference on Industrial Electronics and Applications, pp. 1874-1878, June 2010.

[89] R. D. Sharma, S. L. Agrwal, S. K. Gupta, and A. Prajapati, "Optimized Dynamic Background Subtraction Technique for Moving Object Detection and Tracking," in Proc. International Conference on Telecommunication and Networks (Tel-Net), pp. 1-3, August 2017.

[90] A. Hryvachevskyi, S. Fabirovskyy, I. Prudyus, L. Lazko, and J. Matuszewski, "Method of Increasing the Object Detection Probability by the Multispectral Monitoring System," in Proc. IEEE International Conference on the Experience of Designing and Application of Cad Systems (CADSM), pp. 77-80, March 2019.

[91] S. Jin, D. Kim, T. T. Nguyen, D. Kim, M. Kim, and J. W. Jeon, "Design and Implementation of A Pipelined Datapath for High-Speed Face Detection using Fpga," in IEEE Transactions on Industrial Informatics, Vol. 8, No. 1, pp. 158-167, February 2012.

[92] J. C. Beard and R. D. Chamberlain, "Analysis of A Simple Approach to Modeling Performance for Streaming Data Applications," in Proc. IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 345-349, August 2013.

[93] M. A. Usman, M. R. Usman, and S. Y. Shin, "An Intrusion Oriented Heuristic for Efficient Resource Management in End-to-End Wireless Video Surveillance Systems," in Proc. IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1-6, January 2018.

[94] L. Han, Y. Zhao, S. Yu, B. Zhao, J. Li, and J. Wu, “A General Solution for Multi-Thread based Multi-Source Compressed Video Surveillance System,” in Proc. IEEE International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), pp. 95-99, October 2014.

[95] D. Whitman, “The Need and Capability of A Surveillance Data Distribution System,” in Proc. Integrated Communications, Navigation and Surveillance Conference, pp. 1-6, May 2009.

[96] T. Tsai and C. Chang, “A High Performance Object Tracking Technique with An Adaptive Search Method in Surveillance System,” in Proc. IEEE International Symposium on Multimedia, pp. 353-356, December 2014.

[97] H. Wang, W. Cai, J. Yang, and Q. Chen, “Design of Hd Video Surveillance System for Deep-Sea Biological Exploration,” in Proc. IEEE International Conference on Communication Technology (ICCT), pp. 908-911, October 2015.

[98] M. Wang, B. Cheng and C. Yuen, “Joint Coding-Transmission Optimization for a Video Surveillance System With Multiple Cameras," in IEEE Transactions on Multimedia, vol. 20, no. 3, pp. 620-633, March 2018.

[99] Y. A. Yimamuaishan.Abudoulikemu, Y. Huang and C. Ye, “A Scalable intelligent service model for video surveillance system based on RTCP," in Proc. International Conference on Signal Processing Systems, 2010, pp. V3-346-V3-349, August 2010.

[100] L. Kong and R. Dai, “Object-Detection-Based Video Compression for Wireless Surveillance Systems," in IEEE MultiMedia, vol. 24, no. 2, pp. 76-85, June 2017.

[101] D. Chu, C. Jiang, Z. Hao, and W. Jiang, “The Design and Implementation of Video Surveillance System based on H.264, SIP, RTP/RTCP and RTSP,” in Proc. International Symposium on Computational Intelligence and Design, pp. 39-43, October 2013.

[102] I. R. Khan, A. Hassan, S. Ahsan, S. Alshomrani, and G. Iqbal, “Remote Surveillance with Reduced Transmission Overhead,” in Proc. International Conference on Frontiers of Sensors Technologies (ICFST), pp. 435-439, April 2017.

[103] K. C. Lai, Y. P. Chang, K. H. Cheong, and S. W. Khor, "Detection and Classification of Object Movement - An Application for Video Surveillance System," in Proc. International Conference on Computer Engineering and Technology, pp. 17-21, April 2010.

[104] L. Wu, Z. Liu, and Y. Li, "Moving Objects Detection based on Embedded Video Surveillance," in Proc. International Conference on Systems and Informatics (ICSAI), pp. 2042-2045, May 2012.

[105] A. Premadi, B. Ng, M. S. Ab-Rahman, and K. Jumari, "Real Time Optical Network Monitoring and Surveillance System," in Proc. International Conference on Computer Technology and Development, pp. 311-314, November 2009.

[106] X. Wang, A. Jabbari, R. Laur, and W. Lang, "Dynamic Control of Data Measurement Intervals in A Networked Sensing System using Neurocomputing," in Proc. International Conference on Networked Sensing Systems (INSS), pp. 77-80, June 2010.

[107] J. Han, J. Wang, and P. Wang, "on Robust Stability of Dynamic Interval Systems with Multiple Time-Delays," in Proc. International Conference on Systems and Informatics (ICSAI), pp. 239-243, May 2012.

[108] M. Li, W. Tang, and M. Yuan, "Reliability-based Topology Optimization of Interval Parameters Structures with Dynamic Response Constraints," in Proc. IEEE International Conference on Mechatronics and Automation, pp. 469-474, August 2014.

[109] S. Siddiqui, A. A. Khan, and S. Ghani, "Investigating Dynamic Polling Intervals for Wireless Sensor Network Applications with Bursty Traffic," in Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 448-451, November 2017.

[110] C. Yukonhiatou, T. Yoshihisa, Y. Teranishi, Y. Ishi, T. Kawakami, and S. Shimojo, "A Scheme to Improve Stream Data Analysis Frequency for Real-Time IoT Applications," in Proc. Multimedia, Distributed, Cooperative and Mobile (DICOMO) Symposium, pp. 1205-1211, July 2018.

[111] C. Yukonhiatou, T. Yoshihisa, Y. Teranishi, Y. Ishi, T. Kawakami, and S. Shimojo, "A Scheme to Improve Stream Transaction Rates for Real-Time IoT Applications," in Proc. International Conference on Advanced Information Networking and Applications (AINA), Vol 926, P. 787-798. March 2019.

[112] C. Yukonhiatou, T. Yoshihisa, T. Kawakami, Y. Teranishi, S. Shimojo, "An Implementation of Surveillance Systems with Dynamic Transaction Intervals Under PQI Approach," in IPSJ SIG Techical Report, Vol. 2019-DPS-181, No. 3, pp. 1-6, December 2019.