A Research on Privacy Protection and Large-Scale Distribution Technologies for Public Camera Video Streaming

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2025

Satoru MATSUMOTO

List of publication

Journal papers

1. Satoru Matsumoto, Tomoki Yoshihisa, Tomoya Kawakami, and Yuuichi Teranishi, ”Feature Data Distribution Methods for Person Re-identification using Multiple Cameras,” *International Journal of Informatics Society (IJIS)*, Vol. 15, No.3, pp. 123–131, February 2024.

2. Satoru Matsumoto, Tomoki Yoshihisa, Tomoya Kawakami, and Yuuichi Teranishi, ”A Scalable Video-on-Demand System on Edge Computing Environments,” *International Journal of Informatics Society (IJIS)*, Vol. 14, No.1, pp. 13–21, June 2022.

3. Satoru Matsumoto, Tomoki Yoshihisa, Tomoya Kawakami, and Yuuichi Teranishi, ”A Distributed Internet Live Broadcasting System for Multi-Viewpoint Videos,” *International Journal of Informatics Society (IJIS)*, Vol. 11, No.2, pp. 117–124, October. 2019.

4. Satoru Matsumoto, Tomoki Yoshihisa, Shimojo Shinji, ”A Wireless-Broadcast-Type Video-on-Demand System for Interruption Time Reduction,” *Information Processing Society of Japan Transactions on Digital Content (DCON)*, Vol. 10, No.1, pp. 28–38, February 2022 (in Japanese).

Refereed Conference Papers

1. Satoru Matsumoto, Tomoki Yoshihisa, Hideyuki Shimonisi, Tomoya Kawakami, and Yuichi Teranishi, ”Implementation and Evaluation of a Facial Image Obscuring Method for Person

Identification to Protect Personal Data," in *Proceedings of 2024 IEEE 21th Consumer Communications Networking Conference (CCNC)*, pp. 214–217, January 2024.

2. Satoru Matsumoto, Tomoki Yoshihisa, "Different Worlds Broadcasting: A Video Data Distribution Method for Flexible Bandwidth Allocation in Hybrid Broadcasting Environments, " in *Proceedings of 33rd International Conference on Advanced Information Networking and Applications (AINA 2019)*, pp. 799–809, March 2019.

3. Satoru Matsumoto, Y. Ishi, Tomoki Yoshihisa, Tomoya Kawakami, and Yuich Teranishi, "Different Worlds Broadcasting: A Distributed Internet Live Broadcasting System with Video and Audio Effects," in *Proceedings of IEEE Interna- tional Conference on Advanced Information Network- ing and Applications (AINA)*, pp. 71–78, March 2017.

4. Satoru Matsumoto, Tomoki Yoshihisa, Shinji Shimojo, "A Data Scheduling Method for Video-on-Demand Systems on Radio Broadcasting Environments, " in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, DOI : 10.1109COMPSAC51774.2021.00084, September 2021.

Non-Refereed Technical Papers

1. Satoru Matsumoto, Tomoki Yoshihisa, Hideyuki Shimonisi, Tomoya Kawakami, and Yuichi Teranishi, "Implementation and Evaluation of a Person Identification System Considering Face Image Recognition Accuracy and Privacy Protection," *IPSJ Technical Report (Web)* , vol. 2024-CSEC-104, no. 13, pp. 1–6, March 2024 (in japanese).

2. Satoru Matsumoto, Tomoki Yoshihisa, Tomoya Kawakami, Yuichi Teranishi, " A Study on Live Video Processing Time Reduction Methods for Privacy-oriented Video Management ," *Technical Report of IEICE (IA2021-44)*, vol. 121, no. 300, pp. 65–66, December 2021 (in Japanese).

3. Satoru Matsumoto, Tomoki Yoshihisa, Tomoya Kawakami, Yuichi Teranishi, "Proposal of privacy-oriented video management method for live camera video," *Information Processing*

Society of Japan DICOMO 2021, pp. 813–816, October 2021 (in Japanese).

Preface

In future societies, communication through video is expected to become increasingly widespread. For instance, it is becoming common to broadcast video logs (vlogs) featuring oneself in real-time to a large and unspecified audience.

Platforms such as YouTube enable content creators to deliver live streams or provide on-demand pre-recorded videos to their viewers. Similarly, viewers widely adopt practices like watching live broadcasts or consuming pre-recorded content asynchronously.

The traditional broadcasting model, where identical content is delivered unidirectionally to all viewers, such as in conventional television, is undergoing a transformation. The emerging model prioritizes two-way communication that considers individual privacy during video collection and delivers personalized content efficiently to meet diverse needs during distribution.

This study proposes the construction of a large-scale video distribution service utilizing public cameras. The service is based on footage captured by numerous cameras installed in public spaces, enabling large-scale video distribution. The video distribution infrastructure is designed to widely and in real-time deliver urban video information for diverse public services, such as smart cities, while addressing the critical issue of privacy protection for pedestrians who may be unintentionally captured. To this end, blurring will be applied to the facial images of pedestrians captured by these cameras.

This proposal also provides a service for large-scale distribution of footage captured by numerous public cameras, built upon this video distribution infrastructure. In this service, video distributors can convert footage into VideoLon, VOD, or similar formats and distribute their own facial images captured by public cameras. Distributors can preregister their facial features, enabling the unblurring of their own faces for distribution purposes. Furthermore, as an extension of this service, the system can also provide advanced features such as multi-view video and real-time video editing, in addition to blurring and unblurring processes.

However, there are several challenges to realizing this service. First, creators must upload their facial feature data to the cloud, posing significant privacy risks. Facial features can be used to reconstruct the original face using techniques such as GANs, leading to risks of impersonation and misuse of personal information. Since facial features are unique to individuals, even a single leak can result in irreversible and persistent harm.

Furthermore, the system must support efficient video distribution to serve numerous creators simultaneously. It must also handle extremely large-scale distributions, such as during broadcasts by highly popular YouTubers or in times of major disasters.

To address these challenges, this study defines four technical requirements:

Develop a mechanism that prevents the reconstruction of original face images from leaked facial features, which must also be scalable, lightweight, and fast to execute. Provide a dynamic video processing system tailored to the needs of individual creators, enabling personalized optimization. Enhance system efficiency by leveraging edge computing to optimize bandwidth usage and enable data reuse. Ensure stable large-scale video distribution by combining broadcast waves and internet technologies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In future societies, communication through video is expected to become increasingly widespread. For instance, it is becoming common to broadcast video logs (vlogs) featuring oneself in real-time to a large and unspecified audience.

Platforms such as YouTube enable content creators to deliver live streams or provide on-demand pre-recorded videos to their viewers. Similarly, viewers widely adopt practices like watching live broadcasts or consuming pre-recorded content asynchronously.

The traditional broadcasting model, where identical content is delivered unidirectionally to all viewers, such as in conventional television, is undergoing a transformation. The emerging model prioritizes two-way communication that considers individual privacy during video collection and delivers personalized content efficiently to meet diverse needs during distribution.

1.2 Overview of This Research and Positioning of Each Chapter

This study proposes the construction of a large-scale video distribution service utilizing public cameras. The service is based on footage captured by numerous cameras installed in public spaces, enabling large-scale video distribution. The video distribution infrastructure is designed to widely and in real-time deliver urban video information for diverse public services, such as smart cities, while addressing the critical issue of privacy protection for pedestrians who may be unintentionally captured. To this end, blurring will be applied to the facial images of pedestrians captured by these cameras.

This proposal also provides a service for large-scale distribution of footage captured by numerous public cameras, built upon this video distribution infrastructure. In this service, video distributors can convert footage into VideoLog, VOD, or similar formats and distribute their own facial images captured by public cameras. Distributors can pre-register their facial features, enabling the unblurring of their own faces for distribution purposes. Furthermore, as an extension of this service, the system can also provide advanced features such as multi-view video and real-time video editing, in addition to blurring and unblurring processes.

However, there are several challenges to realizing this service. First, creators must upload their facial feature data to the cloud, posing significant privacy risks. Facial features can be used to reconstruct the original face using techniques such as GANs, leading to risks of impersonation and misuse of personal information. Since facial features are unique to individuals, even a single leak can result in irreversible and persistent harm.

Furthermore, the system must support efficient video distribution to serve numerous creators simultaneously. It must also handle extremely large-scale distributions, such as during broadcasts by highly popular YouTubers or in times of major disasters.

To address these challenges, this study defines four technical requirements:

Develop a mechanism that prevents the reconstruction of original face images from leaked facial features, which must also be scalable, lightweight, and fast to execute. Provide a dynamic video processing system tailored to the needs of individual creators and viewers, enabling personalized optimization. Enhance system efficiency by leveraging edge computing to optimize bandwidth usage and enable data reuse. Ensure stable large-scale video distribution by combining broadcast waves and internet technologies.

Fig 1.1 illustrates the conceptual diagram of the integrated system.

The subsequent chapters of this paper are positioned as follows.

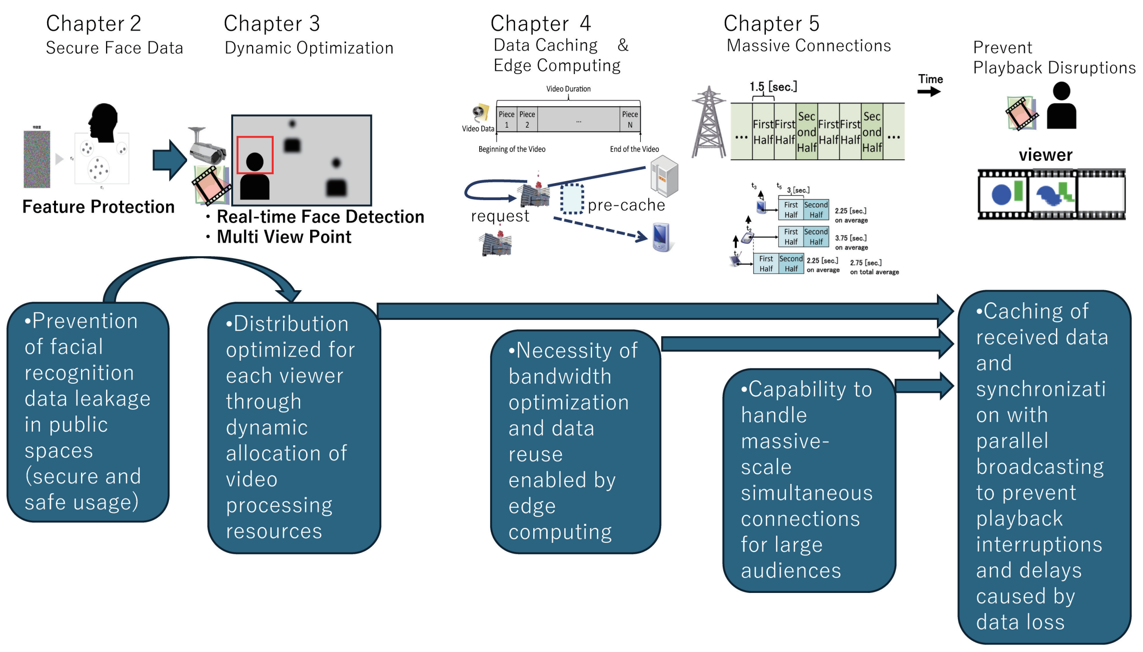

Chapter 2
Secure Face Data
Chapter 3
Dynamic Optimization
Chapter 4
Data Caching & Edge Computing
Chapter 5
Massive Connections
Prevent Playback Disruptions
Feature Protection
・Real-time Face Detection
・Multi View Point
request
pre-cache
1.5 [sec.]
Time
viewer
•Prevention of facial recognition data leakage in public spaces (secure and safe usage)
•Distribution optimized for each viewer through dynamic allocation of video processing resources
•Necessity of bandwidth optimization and data reuse enabled by edge computing
•Capability to handle massive-scale simultaneous connections for large audiences
•Caching of received data and synchronization with parallel broadcasting to prevent playback interruptions and delays caused by data loss

Figure 1.1: the conceptual diagram of the integrated system.

1.3 Chapter 2: Privacy Protection for Facial Features Using Noise Addition

The objective of the proposed method is to enable accurate personal identification while preventing privacy violations caused by information leakage during the registration of facial feature data to the cloud by pre-registered users. To address this issue, the system incorporates a mechanism in which the users themselves add noise to their feature data.

This noise addition method allows for accurate recognition of registered users while ensuring that the original facial image cannot be reconstructed. Compared to homomorphic encryption, it is faster and more efficient, making it suitable for larger-scale processing. Furthermore, unlike reversible methods such as unitary transformations, this approach significantly reduces risks in the event of data leakage.

The proposed method adds noise to facial feature data, similar to differential privacy. However, while differential privacy aims to maintain the aggregate characteristics of the data, this method must enable individual facial recognition even after noise is added. To achieve this, the noise addition is designed to maintain the accuracy of facial recognition while ensuring a high level of personal privacy. Through this approach, the proposed method not only establishes robust privacy protection but also enhances the safety of services utilizing public cameras.

1.4 Chapter 3: Optimization of Dynamic Video Processing Resources

The proposed method aims to achieve stable and high-quality video streaming by reducing network load, minimizing data loss, and preventing delays during simultaneous streaming by multiple users. To achieve this, edge computing technology is utilized.

Edge computing enables efficient use of network bandwidth and facilitates data reuse. This approach distributes and synchronizes data across edge servers, improving the overall efficiency and reliability of the system.

As a result, it ensures uninterrupted streaming and provides a stable, high-quality viewing experience. This method serves as a critical technological foundation for enhancing the quality of video streaming services.

1.5 Chapter 4: Efficient Bandwidth Utilization via Edge Computing

The goal of this method is to reduce network load and minimize data loss and delivery delays, ensuring stable and high-quality video distribution. Edge computing technology is employed to achieve this. By utilizing edge computing, the system efficiently uses network bandwidth and enables data reuse. This mechanism realizes data distribution and synchronization between edge servers, enhancing overall system efficiency and reliability. As a result, uninterrupted and stable high-quality viewing experiences are provided, forming a critical technical foundation for video delivery service quality improvement.

1.6 Chapter 5: Large-Scale Video Delivery Using Wireless Communication

The proposed method aims to achieve large-scale and stable video streaming, even in scenarios requiring massive distribution, such as during broadcasts by highly popular YouTubers or in the event of major disasters. To this end, a hybrid distribution method combining broadcast waves and the internet is proposed.

This method utilizes the VHF band to ensure wide-area and stable distribution. It incorporates a mechanism to divide content and optimize synchronization between transmission and reception. This approach reduces initial buffering delays and guarantees stable streaming, even in environments with massive audiences.

The method is particularly effective in high-load scenarios, such as disaster situations or streaming by popular YouTubers. It significantly enhances the scalability and reliability of video streaming services.

1.7 Integration of Proposed Methods

The proposed methods (1) through (4) are designed to comprehensively address two critical requirements in VLOG distribution services utilizing public cameras: privacy protection and large-scale distribution. Each method addresses specific challenges while playing a complementary role in the system. Together, they form the foundation of a next-generation video distribution service that balances efficiency and reliability.

Chapter 2

Privacy-Preserving Real-Time Facial Recognition with Epsilon Noise

In future societies, communication through video is expected to become increasingly prevalent. For example, it is becoming common to broadcast video logs (vlogs) in real-time to an unspecified audience as a form of communication.

Platforms such as YouTube enable content creators to deliver live streams or provide viewers with pre-recorded on-demand videos. Similarly, viewers have widely adopted practices such as watching live broadcasts and consuming asynchronously stored videos.

In response to this trend, this chapter focuses on developing a real-time facial recognition system designed to protect privacy during video collection, particularly in live streaming scenarios. The proposed system can identify individuals appearing in videos in real time, transmit only the faces of pre-registered individuals to specific recipients, and mask unauthorized faces. These features enable privacy-conscious video streaming.

2.1 Introduction

A privacy-preserving real-time facial recognition system is proposed, focusing on the necessity of protecting privacy in facial recognition systems while maintaining identification accuracy. To address the high computational burden of traditional encryption methods, this study adopts a method that adds epsilon noise to feature data, reducing the risk of facial image reconstruction while enabling personal identification. This approach ensures a critical balance for real-time processing and effectively addresses the challenges of privacy protection in public spaces.

that use cameras installed in public places to identify people in real time are attracting attention. In most of these systems, the user sends facial feature data to the system in advance. Examples include entry systems and attendance confirmation systems.

In these systems, the higher the accuracy of person identification, the more advantageous the system as a means to prevent intrusion by suspicious persons or to furnish solid evidence that a particular person was present. To improve the accuracy of person identification, the user may send a large amount of feature data with a large data size, and the system may make use of the detailed features. On the other hand, if the system understands the detailed feature values, it may be possible to obtain a clear facial image. If feature values are leaked to a malicious party, user privacy may be violated due to unauthorized use. Examples of user privacy violations due to unauthorized use are illustrated in Fig. 2.1. This figure highlights cases such as the redistribution of feature values against the user's intention, unintentional location identification through public cameras, creation of 3D surfaces, electronic lock release, and the generation of fake images and videos by AI.

In identifying individuals in public places, the identifier seeks to obtain detailed feature data necessary for accurate identification, while the identified person wishes the sharing of such data to be minimized, to protect their privacy. The challenge lies in finding the best way to balance these conflicting needs.

The authors have adopted an approach that utilizes obscured facial features and have advanced their research on this method. In this approach, only a specific noise intensity and the obscured facial features are transmitted to a camera edge server installed in public places, ensuring that

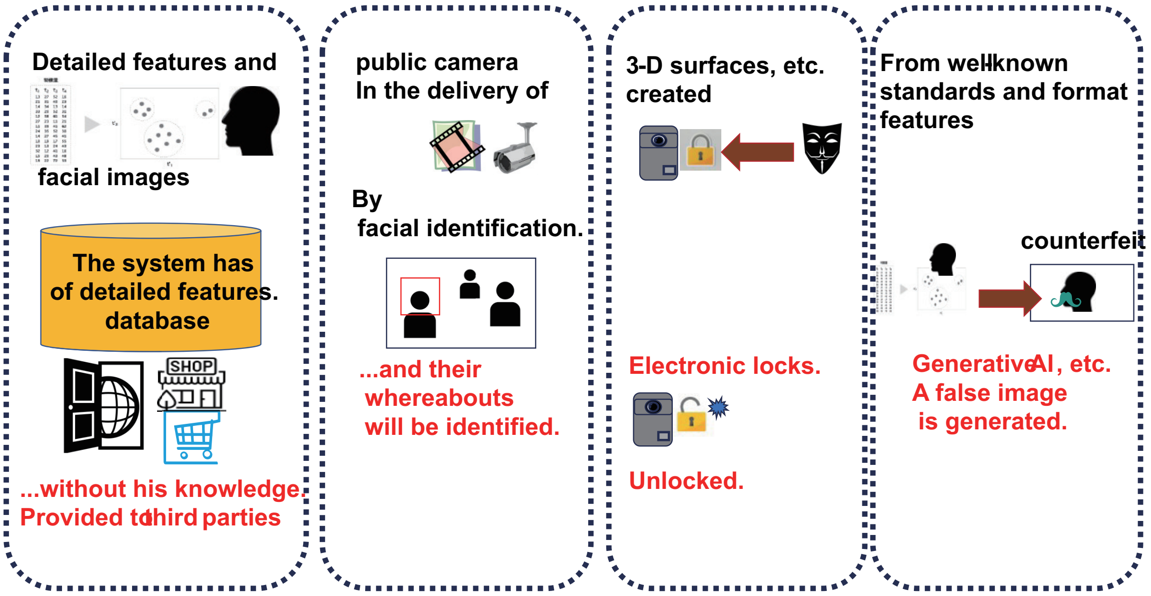

Detailed features and
facial images
The system has of detailed features. database
...without his knowledge. Provided to third parties
public camera
In the delivery of
By
facial identification.
...and their whereabouts will be identified.
3-D surfaces, etc. created
Electronic locks.
Unlocked.
From well-known standards and format features
counterfeit
Generative AI, etc.
A false image is generated.

Figure 2.1: Overview chart of Examples of user privacy violations.

clear facial image data does not pass through the public network. This enables secure and privacy-protective person authentication.

This paper presents methods and evaluates the effectiveness of adding noise to obscure feature data in a person identification system aimed at protecting privacy. The system identifies individuals from camera images without providing detailed facial images or feature data. The accuracy of identification is evaluated by adding noise below a certain intensity to the feature values of facial images provided by the users.

Alternatively to the approach taken by the authors, there is a method of concealing data through encryption, known as quasi-homomorphic encryption. This technique encrypts the data in a form that allows computations to be performed on them. Unlike simple encryption, this method enables operations to be carried out on encrypted data without the need to encrypt and decrypt with each computation, making it more robust for handling data. However, it requires more complex algorithms and additional computational steps, resulting in longer processing times. Furthermore, since the data are encrypted, it is not possible to pre-classify the data for efficient feature comparison as in the unencrypted state, requiring brute-force comparisons, which increases processing time. The authors of this paper prioritize the performance of real-time identification. In this paper, the computational speed of the approach using homomorphic encryption is evaluated in comparison to the approach that adds noise.

Similarly, a method using quasi-homomorphic encryption to conceal feature values is also considered, and a comparison of identification accuracy and processing speed between these two methods is conducted. In addition, protection is necessary to prevent malicious reconstruction using AI from detailed feature data. If such data are leaked, there is a risk that a clear facial image could be reconstructed. The authors evaluated this risk and assessed the accuracy of person identification from the reconstructed images by varying the strength of the noise added to the feature data.

In addition, protection is necessary to prevent malicious reconstruction using AI from detailed feature data. If such data are known, there is a risk that a clear facial image could be reconstructed. The authors assessed the level of risk and evaluated the accuracy of person identification from reconstructed images by varying the strength of the noise added to the feature data.

Following on from this introductory section, the remainder of this paper is structured as follows. Section 2 describes related research. Section 3 describes the proposed method, Section 4 details the authors' proposed NGVlog system, Section 5 evaluates the privacy-aware method, and Section 6 concludes the paper with a summary of the main points.

2.2 Related work

As part of previous research projects, the authors have developed a next-generation video blogging system (NGVlog system), which enables users to upload blogs in real time in a hands-free manner using cameras installed in public spaces [1]. In this system, cameras located at tourist spots or similar locations filmed all passersby, and the faces of individuals other than the vlogger were obscured in the footage. This allowed the vlogger to generate vlog content without filming it themselves, while ensuring the privacy of third parties.

To realize this NGVlog system, the authors conducted research that evaluated the effectiveness of a system that efficiently identifies individuals from video data captured in public spaces, while also providing appropriate privacy protection. The NGVlog system developed in this previous research was created for modeling purposes, such as theoretical verification and early-stage design simulations. In addition, this system only evaluates when noise is added to the feature values.

The system satisfied the following requirements:

(1) The system did not preregister detailed facial images or facial features of the vlogger, ensuring the vlogger's privacy.

(2) The facial images of all individuals except the vlogger were blurred to protect the privacy of third parties.

(3) The system identified the vlogger's face, ensuring that only their face remained unblurred in the video.

In this research, the authors proposed a method that utilized obscured facial features. With this approach, only a certain intensity of noise was sent for noise generation and the obscured facial features to the camera edge servers installed in public spaces, preventing clear facial image data and facial features from passing through the public network.

As a result, even if the facial image data and feature values prepared for identification of the vlogger themselves were more detailed than the resolution of the streaming images, the risk of misuse due to leakage through the public network was reduced, thus allowing users to use the system with peace of mind. However, it was essential to establish a secure mechanism that prevented the edge servers of public cameras from transmitting real-time facial image features over the public network.

The paper suggested that these possibilities could be realized. The NGVlog system, an evaluation environment developed in previous research, was created for the purpose of modeling, such as theoretical verification and initial stage design simulation. At the time of the previous research, the NGVlog system could only evaluate the case where noise was added to the feature values.

In this paper, we continue to develop the NGVlog approach as an evaluation environment, and conduct a detailed investigation and evaluation of optimal noise intensity, and compare it to the use of quasi-homomorphic encryption. Additionally, we examine the extent to which a person can be identified even if feature values with added noise are unintentionally exposed due to leakage, are used by generative AI for image reconstruction.

2.2 Related work

In order to prevent detailed feature data from flowing through network paths, it is essential to manage storage locations carefully. In recent years, the concept of Self-Sovereign Identity (SSI) has been evaluated domestically and internationally from the perspective of privacy protection [2]. We have adhered to the concept of SSI and have adopted a policy of storing data accordingly. Unlike conventional ID systems in which a third party, such as a service provider, manages and controls personal information, SSI allows individuals to own and manage their personal information data in a storage location under their control. For example, to protect privacy, it is possible to limit data items to the minimum necessary for multiple services and to instruct or control the sharing of such items.

In building the system, the authors followed the concept of SSI and placed clear facial data and detailed facial features in a storage location under the control of the data owner. Detailed features may be used illegally; without SSI, personal data may be linked to secondary systems related to the service used without the user's knowledge or control, and facial images and feature data may be used for secondary purposes. Related research on the problems caused by providing detailed facial image features includes the following. If detailed facial image features are provided to a system that uses features in a general format that can be handled by model data available for deep learning, the system may create false images using generative AI. These can lead to privacy violations [3–5]. Differential privacy [6] is a technique to prevent detailed information about an individual's data from being revealed in data analysis and model learning. Specifically, it is a method that prevents the disclosure of detailed information about individual data in a dataset.

In this paper, the authors conduct research and evaluation within an experimental environment that protects data based on the concept of SSI, using their proposed system. Additionally, following the principles of differential privacy, the authors investigate and assess the appropriate intensity of noise to be applied, using a model similar to the identification model used in the feature extraction part of the NGVlog system. This research contributes to understanding how to protect data and determine the optimal level of noise to add for effective privacy protection.

Yamanaka et al. analyzed the role of noise addition in achieving differential privacy and mathematically examined the sufficient conditions for noise distributions that satisfy differential privacy [7]. They proposed several new noise distributions and compared their sizes based on percentile points. In particular, they focused on epsilon noise, which controls the strength of privacy protection, and evaluated the impact of noise addition according to the epsilon parameter. Their results revealed that while alternative distributions can potentially meet differential privacy requirements, Laplace noise excels in maintaining data utility while offering flexibility in settings determined by the epsilon parameter. The study highlights that the sharp peak and rapid decay characteristics of the Laplace distribution make it optimal for balancing privacy protection and data usability. This analysis underscores the importance of carefully selecting the type of noise, especially in sensitive applications such as facial recognition features, to achieve a balance between privacy protection and data utility.

Relation Between the Proposed Method and Differential Privacy The relationship between the proposed method and the concept of differential privacy is discussed below.

Differential privacy is a framework that guarantees that the inclusion or exclusion of an individual's data in a dataset does not significantly affect the aggregated results. To achieve this, random noise is added to the data, reducing the risk that third parties can infer specific personal information from the output.

One of the key parameters in differential privacy is epsilon, which controls the intensity of the added noise. A smaller epsilon provides stronger privacy protection by introducing more noise, reducing the likelihood of identifying specific individuals. However, this comes at the cost of decreased data utility. Conversely, a larger epsilon improves the accuracy and utility of the data but weakens privacy protection. This trade-off between privacy protection and data utility is a fundamental aspect of many privacy-preserving systems.

Traditional applications of differential privacy primarily focus on protecting statistical query responses by adding noise to aggregated data. The goal is to prevent sensitive personal information from being leaked through repeated queries. In contrast, the proposed method in this study applies small epsilon-noise directly to individual facial features, thereby preventing the leakage of personal identification information while maintaining recognition accuracy. While the granularity of noise addition differs, both approaches share the common principle of privacy protection through noise addition.

The proposed method is not a direct implementation of the theoretical framework of differential privacy. However, the core concept of adding noise to make personal identification more difficult aligns with the primary objective of differential privacy. Therefore, the proposed method can be regarded as a practical application of differential privacy principles in a different context.

In summary, while traditional differential privacy aims to protect statistical query responses, the proposed method emphasizes privacy protection in real-time facial recognition systems. The difference lies in the application domain. However, the fundamental principle of preventing personal information leakage by adding noise remains consistent across both approaches.

The balance between privacy protection and recognition accuracy achieved by the proposed method highlights its practical applicability in real-world systems. Unlike traditional differential privacy, which often sacrifices usability for stronger privacy protection, the proposed method aims to maintain identification performance while protecting personal information. A detailed comparison between these approaches is provided in Table 3, which illustrates the trade-offs in privacy and utility.

Table 2.1: **Comparison of Privacy Protection and Identification Accuracy**

Method	**Privacy Protection Level (ϵ)**	**Identification Accuracy**	**PSNR (dB)**
Differential Privacy (Traditional)	Small ϵ	Low	-
Proposed Method (This Study)	Moderate ϵ	High	60.7

The proposed method adds noise to facial features, similar to differential privacy. However, while differential privacy aims to preserve group-level characteristics, the proposed method ensures individual face recognition remains possible after noise addition. To achieve this, the study examines noise addition that maintains facial recognition accuracy while ensuring a high level of privacy protection. This approach establishes privacy protection and enhances the safety of services utilizing public cameras.

Maekawa et al. proposed a novel method for generating protected biometric templates, such as those used in facial recognition systems, through the application of random unitary transformations [8]. By employing random unitary matrices, their method safeguards templates (feature data) while preserving important geometric properties, including inner product, Euclidean distance, and correlation coefficients, thereby ensuring the concealment of visual information. Furthermore, the proposed approach maintains the reusability and security of the templates. Templates generated via unitary transformations do not degrade the learning performance of Support Vector Machines (SVM). Moreover, their method addresses potential security risks, such as key or template leakage, by incorporating strategies like the use of multiple keys and periodic key updates. The effectiveness of this method was validated through experiments conducted on the Extended Yale Face Database B, demonstrating that the approach does not adversely affect authentication performance metrics, such as the False Acceptance Rate (FAR) and False Rejection Rate (FRR).

Unitary transformations are reversible, which poses a risk of reconstructing the original data with high accuracy. To address this, the authors propose adding Laplace noise as an effective method to negate reversibility and reduce the risk of data reconstruction. This approach enhances personal data security and prevents critical risks in data protection.

While some proposals use random unitary matrices or keys for secure operations, the authors express concerns about the complexity of handling keys and matrices manually or transmitting them over networks. Instead, they adopt a noise generation approach that does not rely on complete reconstruction keys but rather transmits only the strength of the noise.

The comparison between noise addition and unitary transformation is summarized in Table 2.2, highlighting their respective characteristics and applications.

Noise addition, such as using Laplace distribution, works by blurring features through random noise. While this method is generally applicable to any type of data, excessive noise can degrade data usability. It is computationally efficient, as noise generation and application are relatively fast. Privacy risks arise when noise levels are too low, potentially allowing reconstruction of original data. To enhance security, noise distributions and their magnitudes can be dynamically adjusted. This method is commonly used for general data privacy protection, including differential privacy.

In contrast, unitary transformation geometrically transforms the entire dataset using random unitary matrices, preserving geometric properties like Euclidean distance and inner product. These properties ensure minimal impact on analysis accuracy and model performance, making this approach suitable for tasks requiring geometric data structures, such as support vector machines

(SVMs). However, its applicability is limited to algorithms that depend on these geometric properties. Computationally, generating random unitary matrices is resource-intensive, but once generated, transformations can be performed efficiently. Privacy risks stem from the potential leakage of the unitary matrix, which could allow complete data reconstruction. Security can be enhanced by using multiple keys, random matrices, or periodic template updates. This method is particularly suitable for applications in face recognition and biometric authentication.

Thus, noise addition and unitary transformation differ in their applicability, computational efficiency, and approaches to privacy and security. Each method has its own strengths and is suitable for different use cases.

The proposed method adds noise to facial features, similar to differential privacy. However, while differential privacy aims to preserve group-level characteristics, the proposed method ensures that individual face recognition remains possible after noise addition. To achieve this, the study examines noise addition strategies that maintain facial recognition accuracy while ensuring a high level of privacy protection. This approach establishes robust privacy safeguards and enhances the security of services utilizing public cameras.

Furthermore, some research has described the basic concept of differential privacy and the usefulness of using epsilon noise to prevent the leakage of sensitive information [9]. There is also related research that uses cryptographical features or makes the system robust to protect privacy, such as the work by Narayanan and Shmatikov on linkedness vulnerabilities, where individual data points in a dataset can be linked to external data sources to derive an image. They discuss avoiding this possibility [10].

Following the approach of the study mentioned above, in this research, the authors decided to use epsilon noise, which allows for easy adjustment of the noise intensity, in their evaluation experiments. The evaluation section of this paper will furthermore demonstrate which type of noise distribution is most effective.

The research by Fredrikson et al. also describes attack techniques related to data inversion to avoid the possibility of backcalculating individual data from the output of a model [11]. When using Euclidean distance as a similarity metric for face recognition, some studies have used quasi-homomorphic encryption techniques, which allow operations to be performed in encrypted form.

Table 2.2: **Comparison of Noise Addition and Unitary Transformation**

Characteristic	**Noise Addition**	**Unitary Transformation**
Data Protection Method	Adding random noise (e.g., Laplace distribution) to blur features.	Applying random unitary matrices to transform the entire dataset geometrically.
Preservation of Data Usability	Excessive noise may degrade data usability.	Geometric properties such as Euclidean distance and inner product are preserved, ensuring minimal impact on analysis accuracy and model performance.
Scope of Application	Applicable to any data.	Limited to algorithms that utilize geometric properties of data (e.g., SVM).
Privacy Risks	Low noise levels may allow reconstruction of the original data.	Leakage of the unitary matrix may lead to complete reconstruction of the original data.
Computational Efficiency	Noise generation and addition are relatively fast.	Generating random unitary matrices is computationally expensive, but once generated, template transformation is efficient.
Methods for Enhancing Security	Dynamic adjustment of noise distribution and amount to improve security.	Using multiple keys and random matrices, as well as periodic template updates, to strengthen security.
Practical Examples	General data privacy protection (e.g., differential privacy).	Applicable to tasks requiring geometric properties, such as face recognition and biometric authentication.

Yang et al. [12] attempt to improve privacy protection in cloud-based face recognition systems using quasi-isomorphic encryption techniques. Using the CKKS scheme, which allows floating-point operations to be performed during encryption, the majority of errors in face recognition are reduced to a very low rate of less than 0.02%.

However, this method requires comparison between all features, which increases the processing time when the number of target features increases.

Nakanishi et al. [13] are developing a system that integrates the face recognition engines of various companies and uses quasi-homomorphic encryption techniques to reduce the risk of data leakage. As the use of cryptography increases response times, they identify that it is important to limit the number of registered users to less than 120 and use a clustering strategy to achieve a practical level of response time.

The authors propose an approach that considers differential privacy and uses obscured data with epsilon noise (epsilon-noise) added to the features, before comparing facial image features for person identification. This method allows for obscuring, does not demand the processing time needed for encryption and decryption, and is expected to have a lower computational load than that of quasi-isomorphic encryption.

In differential privacy, epsilon-noise is often used to protect individual data while minimizing its impact on overall data analysis results. By appropriately adding epsilon-noise, the statistical trends and patterns of the entire dataset can be preserved, while making it difficult to identify individual data points. Thus, epsilon is a parameter that balances the strength of privacy protection and data utility. Generally, a smaller epsilon adds more noise, making it harder to identify personal data, thus strengthening privacy protection. However, more noise can reduce the accuracy of the data. Conversely, a larger epsilon adds less noise, increasing the accuracy of the data but weakening privacy protection. Since the relationship between noise and privacy protection is not linear, the intensity of noise is defined as the inverse of epsilon. By defining noise in this way, we can prevent an excessive increase in noise and maintain both privacy protection and data utility within a practical range.

2.3 Research Objectives and Proposed Methodology

In this study, we address the challenge of balancing accurate identification of individuals with the protection of their privacy. The following subsections outline the research objectives and the proposed methodology for developing a system that leverages epsilon-noise to protect personal information while ensuring reliable person identification in public spaces.

2.3.1 Research Objectives

In today's society, the identification of individuals in public places is an important issue from the perspective of both security and the protection of personal privacy. In the case of large data sets of people from all over the world, it is technically and ethically difficult to identify individuals based on individual characteristics. However, identification of individuals by comparing characteristics within a limited population is a more realistic and manageable approach.

The purpose of this research is to develop a system for identifying individuals in video data captured in public places using feature data sets obtained from a limited population. In addition, we verify the effectiveness of a method of adding noise to feature data to protect privacy while maintaining the accuracy of individual identification.

The main objective of this paper is, from a privacy protection perspective, to prevent users from leaking clear facial features to public networks; and, furthermore, to clarify the balance to be struck between disambiguation methods and identification accuracy. In other words, the aim is to disambiguate the features of an individual's facial image and provide them to a person identification system so that the individual can be accurately identified even when receiving disambiguated data.

2.3.2 Proposed Methodology

As the first step of the proposed method, feature data of facial images are collected in advance from a limited population of registered persons, and epsilon-noise is added to the feature data to create an ambiguous data set that protects personal information based on the principle of differential privacy. Individuals are then identified by calculating the closest Euclidean distance of the feature data. By adding noise to the feature data, we explore how to minimize the impact on identification accuracy while enhancing personal privacy protection, and finally propose a technical solution that strikes a balance between public safety and personal privacy.

The formula for adding epsilon-noise is shown below; where X is the original feature vector, let X_{noisy} be a feature vector with noise-added function as in (1). In the differential privacy process of the proposed method, the noisy dataset X_{noisy} , which is the dataset X plus epsilon-noise, is defined using the Laplace distribution as follows.

$$X_{\text{noisy}} = X + \text{Laplace}\left(0, \frac{1.0}{epsilon}\right) \tag{2.1}$$

This Laplace$(0, \frac{1.0}{epsilon})$ is center $\mu = 0$, scale parameter $\beta = \frac{1.0}{epsilon}$. The noise from the Laplace distribution of the feature is shown and its probability density function $f(x|\mu, \beta)$ is expressed as (2).

$$f(x|\mu, \beta) = \frac{1}{2\beta} \exp\left(-\frac{|x-\mu|}{\beta}\right), \tag{2.2}$$

where $\beta = \frac{1.0}{epsilon}$ is the parameter controlling the privacy intensity epsilon is taken to be the inverse of a. The cosine similarity is calculated using equation (3).

$$\text{cosine similarity}(A, B) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|A\|\|B\|} \tag{2.3}$$

Based on the pilot modeling alluded to earlier in this paper, the authors are currently building a system that allows individuals to create VideoLogs using cameras installed in open places. Here, a large number of cameras clustered at specific tourist spots can be used to capture the faces of passers-by at a distance sufficient to identify them, blur the faces of those who do not want to be seen, and send the images to the VideoLog(Vlog) system in real time. However, spending a lot of computation time on blurring delays the process of transmitting the camera image and causes frustration for the viewer, as the viewing image is provided intermittently.

The authors have devised a method to add epsilon-noise to facial image features and use noise-obscured facial image features, by adding only a certain intensity of noise, to reduce computational load and prevent transmission and reception of clear and detailed images of facial features belonging

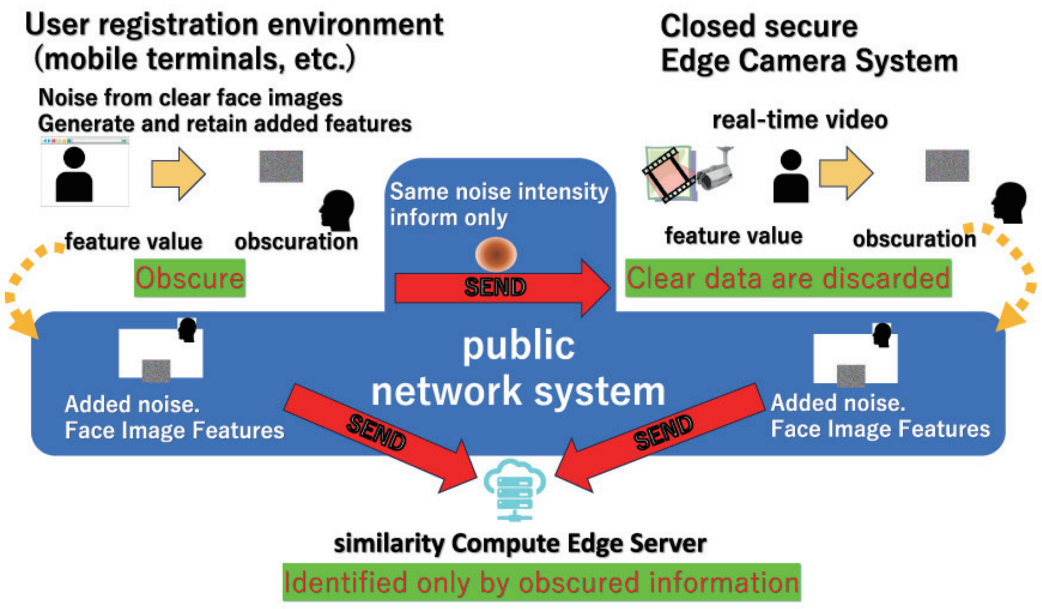


Figure 2.2: Overview chart of research objectives from the privacy protection brief.

to an individual on public network paths. The paper presents a method for preventing the transmission and reception of facial images and detailed facial image features of particular individuals over public network paths.

In this paper, in particular, noise is added to the features calculated from an individual's facial image in order to obscure them, and it is evaluated to what extent the features can be minimally obscured so as to identify the individual. Figure 2.2 shows a conceptual diagram of research objectives in terms of privacy protection.

2.3.3 NG Vlog System

This section describes the features of the NGVlog system. Figure 2.3 shows a schematic diagram of the system in construction. The system consists of a registration system for the feature values of the facial images on the user side, a camera that is paired with an edge server in a sightseeing spot, a camera edge server, and multiple camera edge servers that are connected to a network to transmit feature values. The system is designed to store the user's personal information, such as clear facial

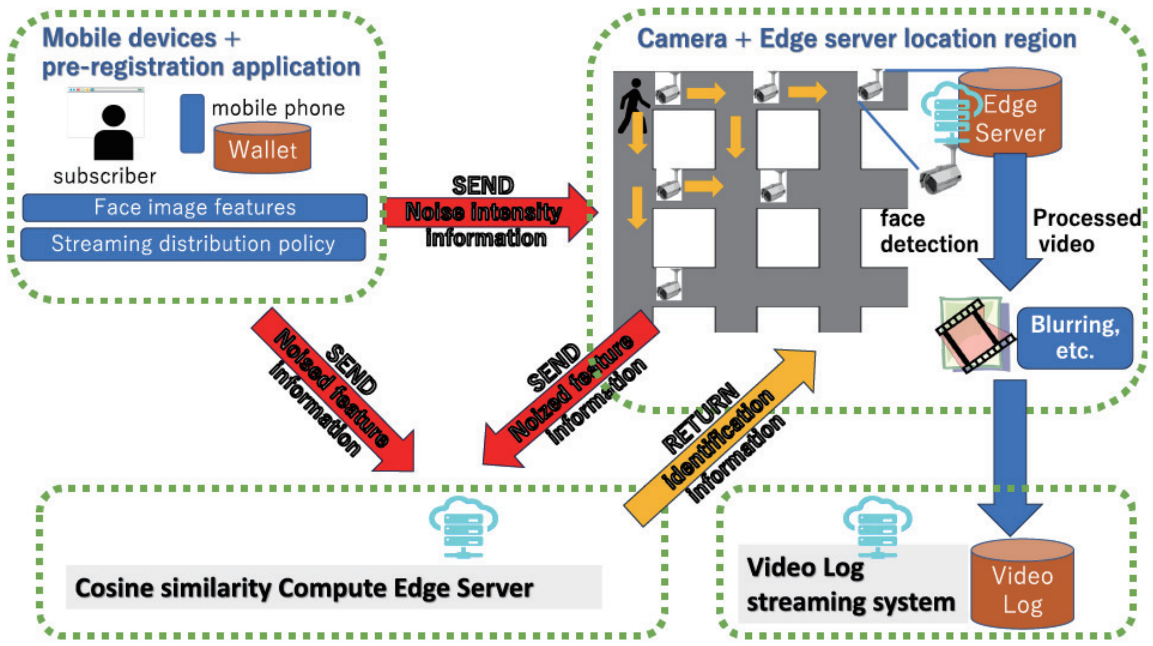

Mobile devices + pre-registration application
subscriber
mobile phone
Wallet
Face image features
Streaming distribution policy
SEND Noise intensity information
Camera + Edge server location region
Edge Server
face detection
Processed video
Blurring, etc.
SEND Noised feature information
SEND Noized feature information
RETURN Identification information
Cosine similarity Compute Edge Server
Video Log streaming system
Video Log

Figure 2.3: Overview schematic diagram of the system.

images and detailed feature values, as much as possible at the storage location specified by the user or on a portable terminal, following the SSI.

Figure 2.4 below provides an overview of how this system is used. The following is an example. First, the user uses an application to save a high-quality, detailed face image from a PC or mobile terminal to a destination specified by the user. Next, the user saves the face image and facial features with noise. At this point, the user can select and specify in advance the relationship between the user and the viewer on the social networking service, as well as the policy for publishing the Vlog itself.

After the system user arrives at the sightseeing spot, he/she instructs the system to commence deploying from his/her mobile terminal. Each edge camera server in the sightseeing spot receives the feature data in a form that has been obscured from the storage location specified by the user or the storage location of the mobile terminal and relays the data to the edge servers in the neighborhood that are linked to the system for distribution.

Then, for each frame of the real-time video, all detected facial images are extracted, and the

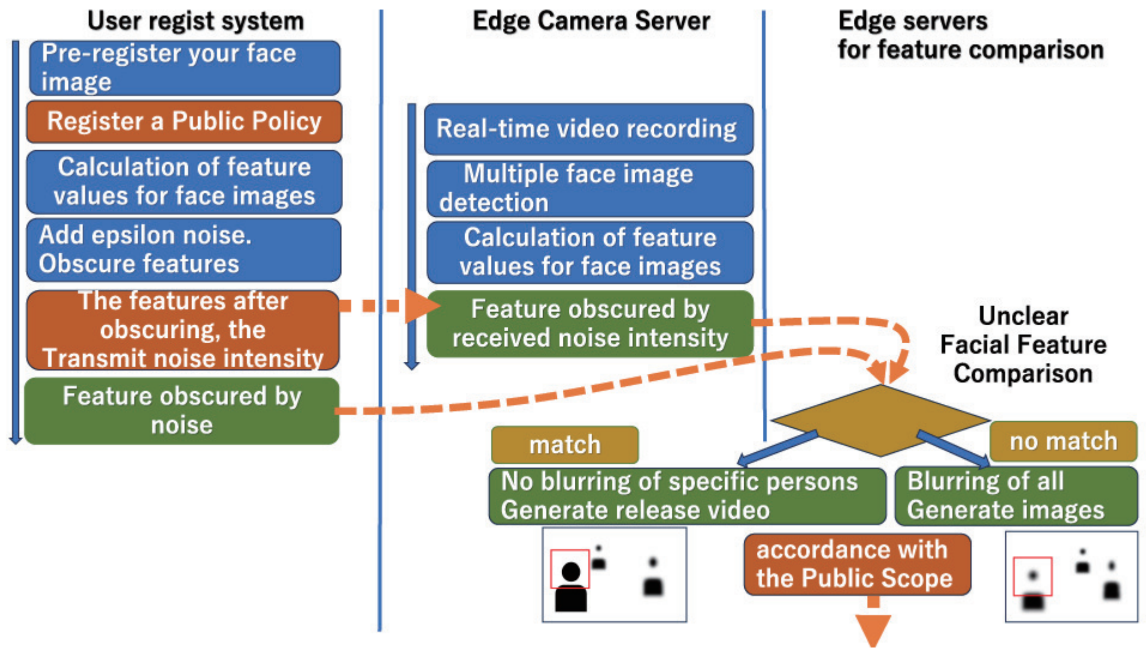

User regist system
Pre-register your face image
Register a Public Policy
Calculation of feature values for face images
Add epsilon noise. Obscure features
The features after obscuring, the Transmit noise intensity
Feature obscured by noise
Edge Camera Server
Real-time video recording
Multiple face image detection
Calculation of feature values for face images
Feature obscured by received noise intensity
Edge servers for feature comparison
Unclear Facial Feature Comparison
match
no match
No blurring of specific persons Generate release video
Blurring of all Generate images
accordance with the Public Scope

Figure 2.4: Overview of how this system is used.

quantity of obscured features is calculated. Then, using the obscured features for a specific user and the obscured features obtained from the real-time video, the system calculates the similarity and blurs all face images except those of the identifiable persons and distributes them as a stream. When distributing this stream via social networking services and similar, the stream is distributed according to the pre-selected distribution policy described above.

2.4 Evaluation

The system used in the evaluation identifies individuals in video data captured in public places using feature data sets obtained from a limited population. In this evaluation, we test the effectiveness of a method that adds noise to feature data to protect privacy while maintaining the accuracy of individual identification.

2.4.1 Evaluattion Method

We evaluated the discrimination accuracy of the system using public video data from a video sharing service. The evaluation system was used to determine whether or not a registered individual was present in the video by matching it with a feature data set stored on an edge server. An overview of the evaluation methods is shown in Fig. 2.5. The details of the data flow and evaluation methods regarding Fig. 2.5 are as follows. This evaluation aims to create a system that protects privacy while identifying individuals within specific groups, such as at tourist destinations, and first evaluates whether it is possible to prevent the generation of original face images based on the feature values of individual face images.

In particular, this research evaluated the accuracy of discrimination against the noise added for obscuration. Because the system was not fully constructed, part of the evaluation was conducted through simulation. In the simulation, to obscure the user's original data, an appropriate level of noise was added to the features, and the edge server calculated the obscured features for all facial images extracted from each frame. The user does not send clear images or features through the public network, but only features that have already been obscured.

The camera edge server identifies individuals by comparing the shortest Euclidean distance between the obscured features and the calculated real-time video obscured features by noise intensity, with a common information quasi-encryption key sent in advance.

The evaluation procedure for the proposed method is as follows. The first half of this evaluation, Section B,C, is a preliminary research study to determine how much noise to add in order to obtain a valid effect, based on the Cosine Similarity and K-means methods.

The second half of the evaluation, Sections D-E, compares the accuracy and processing time of the proposed method with those of using quasi-homomorphic encryption.

Furthermore, we will attempt to reconstruct the original images using facial features, and investigate the extent to which the identification of individuals is hindered by adding noise to facial features. In sections F-I, we will discuss the results of a detailed analysis of these effects. The authors provided detailed explanations to the participants, from whom they requested facial features for use in the camera system, ensuring sufficient privacy protection.

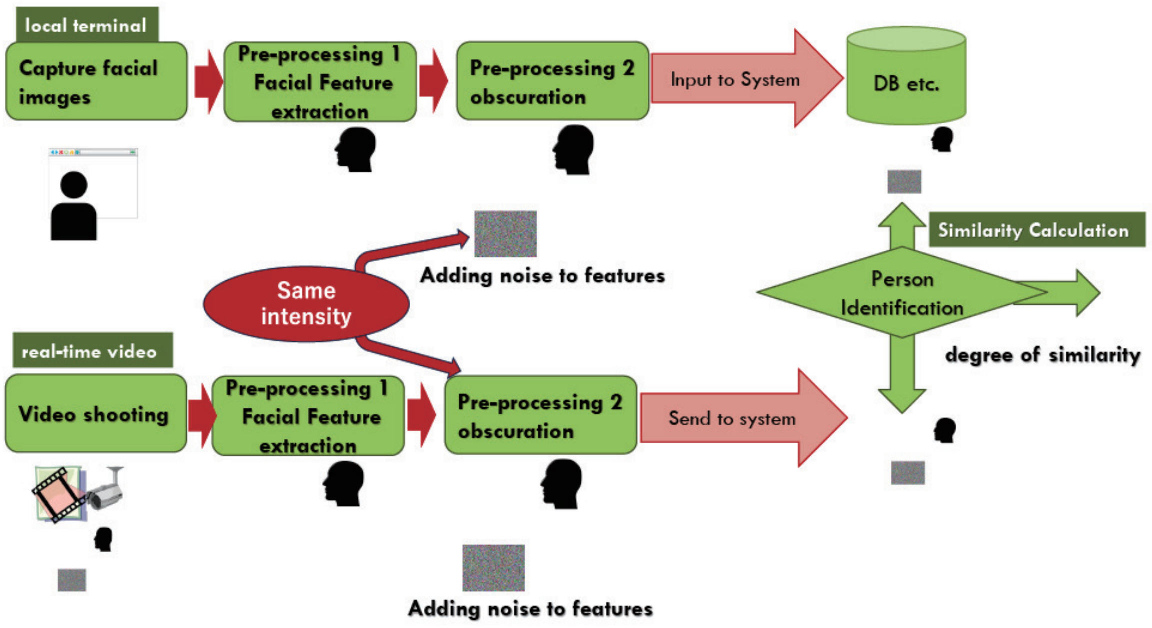


Figure 2.5: Overview of the evaluation methods.

2.4.2 Evaluation Environment

The environment and equipment specifications are as follows:

- a PC with a 12th generation Intel(R)Core(TM) i7 2.30 GHz CPU and an Nvidia RTX4070 Laptop GPU, and
- a PC with a 14th generation Intel(R)Core(TM) i9-14900K 3.20 GHz CPU and an Nvidia RTX4090 GPU.

We used Windows 11 Professional 64-bit OS, running Python from the command prompt. We used the fast Insightface library for facial image detection [14]. In the evaluation environment, the facial images used were at a resolution of 693 × 630 pixels, the same as the resolution required for the facial images used for official identification purposes (320dpi).

2.4.3 Evaluation of Noise-Added Feature Data for Person Identification

This section focuses on the evaluation of the accuracy of person identification when epsilon-noise is added to facial features, detailing the comparison between clear and noise-added features, and the effect of noise on clustering and identification accuracy using K-means. Comparison of PSNR (Peak Signal-to-Noise Ratio) values and cosine similarity, as well as the method for processing the real-time images, is included here.

In this research, we envision a system that classifies individuals based on facial image features to identify an individual among multiple people. The system targets a number of people in a scene captured by a camera and identifies individuals by comparing the features in facial images provided by the user with those extracted from the real-time video. In particular, we focus on the difference in identification accuracy when epsilon-noise is added to the features according to the privacy disambiguation method, whereby noise is intentionally added for protection or not, and the effect of the noise is evaluated. The privacy protection of facial features processed by the camera edge server is also taken into account. In the identification process, the features sent to the edge server are examined to evaluate the feasibility of user identification through a comparison with the features of real-time face videos. The effectiveness of this approach was validated and evaluated with video data from 150 individuals, including video images sized 80 horizontal by 100 vertical pixels or larger, showing facial images from YouTubeFacesDatabase [15].

The YouTubeFacesDatabase is very useful for research into facial recognition and has been used by many researchers. This paper used 7,500 frames of images from the database to create class information for person identification. The name labels and the data for the center position of the face in each frame were used as teacher information, and the video folders in which the face images were easy to identify were used as correct data. The authors then evaluated whether individuals could be classified and identified using sample data where there were larger movements and smaller, slightly less visible facial images than in the correct data. The authors used the K-means function of the Python Scikit-learn library for classification. This function divides the data into a fixed number of classes by progressively minimizing the Euclidean distance from the center of gravity of each data point until convergence is reached. In person identification validation, classes were first defined, and the K-means function was used to determine the center of features for each class. The features of the person-labeled faces were then extracted from the face detection of the sample frames and compared to the center of features for each class based on similarity, from which the closest class was derived. The distance asymptotically minimized by the K-means function is expressed as (4).

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \|x_i - \mu_j\|^2 \tag{2.4}$$

The authors compared the results found for detailed and obscured features to check for similarity. 1) The features of the samples to which no epsilon-noise was added were compared with the centers of each class for which classification calculations were performed without epsilon-noise. For each class for which classification calculations were performed without the addition of epsilon-noise, a principal component analysis of the 512 dimensional features was performed. Figure 2.6 shows the classification projected onto the two axes. 2) Comparison of the centers of the features of the epsilon-noise-added samples with the centers of each noise-added class for which classification was computed with epsilon-noise added. A principal component analysis of the 512-dimensional features was performed for each class for which classification was computed with epsilon-noise added. Figure 2.7 shows the projection of the classification on the two axes. The similarity is not much different from that in Fig. 2.7. Comparison of Figs. 2.6 and 2.7 shows that the centers of the cluster distributions are relatively diffuse after adding noise, and the center points are also shifted.

2.4.4 Performance Comparison of Quasi-Homomorphic Encryption and Noise-Based Methods

This section will compare the performance of quasi-homomorphic encryption with the proposed noise-based method, discussing processing times, correct response rates, and practical feasibility. Here will be located the findings related to the speed and accuracy of these methods and their potential for practical application.

The authors experimented with the CKKS scheme, which allows floating-point operations to be performed in encrypted form, using Microsoft Seal, a fast library widely used for quasi-homomorphic cryptographic computations. The correct response rate for each number of registrants is shown in

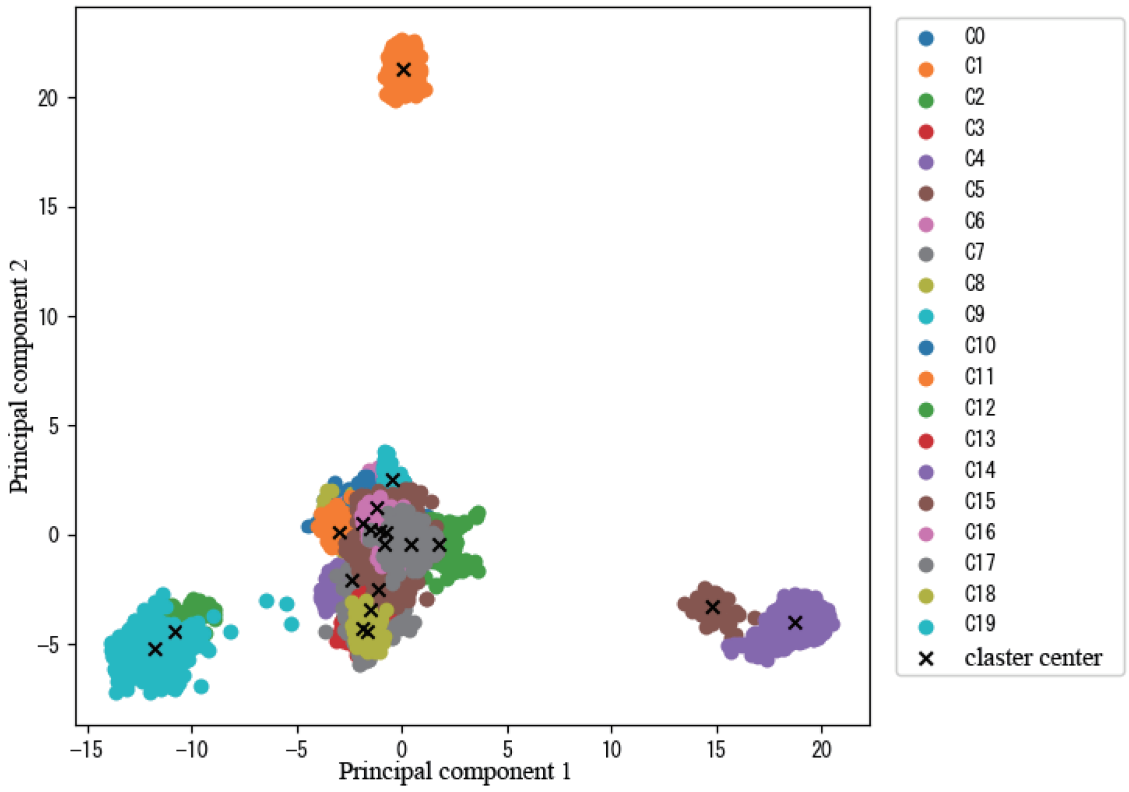


Figure 2.6: Principal component analysis projected on two axes.

Table 2.3. The correct response rate was measured as success or failure in identification based on the YoutubeFace label information. The correct response rate with quasi-isomorphic encryption and without noise is almost the same, and when noise is added, the correct response rate drops by 0.005 accuracy. However, the situation is such that the correct response rate exceeds 90%, and there is a possibility of practical use if the system is made multifactoral with another identification factor and systematically assisted.

The author tried the CKKS method, which can perform floating point operations in encrypted form, using Microsoft Seal, a fast library widely used for quasi-homomorphic cryptographic computations. The percentage of correct answers before and after the addition of epsilon-noise and in 100 registered users of quasi-isomorphic encryption is shown in Table 2.3. The correct answer rate

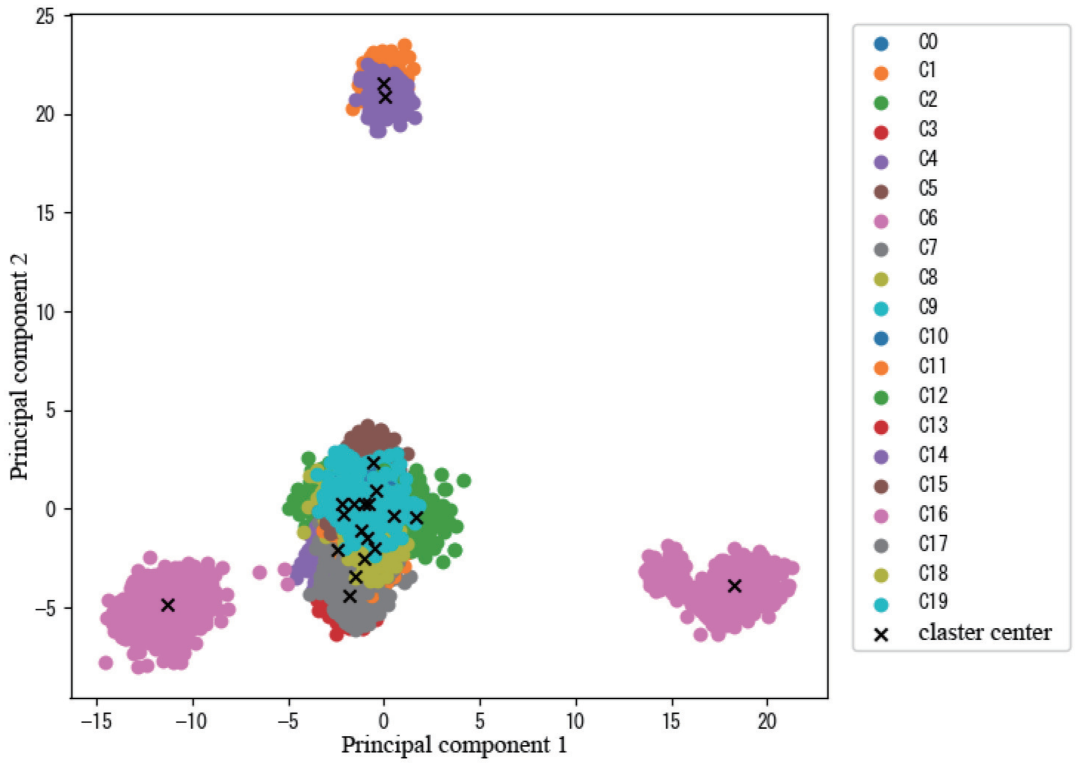


Figure 2.7: Principal component analysis projected on two axes when noise is added.

was measured by the success or failure of the identification based on the YoutubeFace label information. The correct response rates for the cases with quasi-homomorphic encryption and without adding noise were almost the same, while the correct response rate decreased by 0.005 accuracy when noise was added. However, the situation is such that the correct response rate exceeds 90%, and a multifactor authentication system that combines another identification factor and systematic aids could be practical. A paired t-test was conducted to compare the FRR values between the Homomorphic encryption method and the proposed noise-based method. The results showed no statistically significant difference between the two methods (t = -1.49, p = 0.158). This indicates that the proposed noise-based method achieves a comparable identification accuracy to the Homomorphic encryption method. The analysis of variance (ANOVA) is a statistical method used to determine whether there are significant differences between the means of multiple groups. By comparing the variance between groups and within groups, ANOVA calculates the F-statistic and p-value to assess whether observed differences are statistically significant. In this study, ANOVA was applied to evaluate the impact of varying noise intensities (epsilon) on the True Acceptance Rate (TAR) across different systems, helping to identify the optimal balance between identification accuracy and privacy protection. An ANOVA test was conducted to evaluate the impact of different noise intensities (epsilon) on the Peak Signal-to-Noise Ratio (PSNR). The groups were divided based on the epsilon values as follows: Group 1 (epsilon = 0.5 to 2.0), Group 2 (epsilon = 3.5 to 5.0), Group 3 (epsilon = 5.5 to 7.0), and Group 4 (epsilon = 7.5 to 9.5). The results showed a statistically significant difference between the groups (F = 32.62, p ¡ 0.05), indicating that the noise intensity significantly affects the PSNR values. This suggests that higher noise levels result in improved PSNR, demonstrating the robustness of the proposed noise-based method.

In an initial evaluation, the features of the face images were obscured by adding effective epsilon-noise to the differential privacy, as described in the related work in Section 2. The accuracy of person identification was then checked by varying the intensity of the noise. Five hundred real-time images were used for this evaluation. The strength of the noise was varied and the predictive similarity between the input image with added noise and the real-time image was plotted. The dB values of the PSNR of the 512-dimensional matrix of features before and after adding noise to the input image are also shown in the figure.

The vertical axis is the similarity, the dB value of the PSNR of the 512-dimensional feature matrix before and after adding noise, and the horizontal axis is the epsilon intensity. It emerged that the epsilon values above 2.0 did not subjectively identify the person correctly when several people visually confirmed it. The figure shows that when the value of epsilon is greater than 2.0, the cosine similarity is greater than 0.6 and the PSNR value is greater than 50 dB. In the evaluation, the identification was considered successful when the PSNR value was greater than 50 dB and the cosine similarity was greater than 0.6. We used the Laplace distribution when adding epsilon-noise. When Laplace distribution noise is added with epsilon set below 2.0, individuals can be identified while preserving privacy, and in the case of epsilon-noise, epsilon can be fine-tuned. To improve differential privacy, we considered an approach to add epsilon-noise and evaluate it before processing facial landmark positions in the feature extraction process. The feature extraction model used in this research was described using the widely-adopted ONNX model format, which allowed us to decode the computational process and calculate the feature values. We also explored the possibility of using the Buffalow-l model from Insightface, in its ONNX format, to identify a person based on facial features.

However, due to the extreme complexity of the Buffalow-l model, it proved challenging to fully implement a method to effectively suppress the inference of feature data points by appropriately adding epsilon-noise before the convolution process. Developing a VAE or GAN (Generative Adversarial Network) specialized for this model posed a significant challenge. As a result, we chose to conduct the evaluation using ResNet50, which serves as the foundation of the Buffalow-l model.

We aim to develop a module that generates the original image using a process similar to the feature extraction method used in this research. This module will be evaluated and described in Sections F-H. To benchmark appropriate noise parameters, epsilon-noise was added to both the final calculated feature values and those extracted from the original face image. We compared the PSNR between the feature value matrices and verified the FRR to estimate the strength of the noise.

In the subsequent H-G section, we employed UNet, ResNet50, and GAN to generate images similar to the original. We compared the PSNR with the original image and evaluated the ease of identifying individuals when clear feature values were provided.

The processing time versus the number of registrants is also shown in Fig. 2.8. The upper figure shows the results without noise and with quasi-homomorphic encryption and noise added, and the lower figure is an enlarged version of the results without noise and with noise added. It emerges that the quasi-homomorphic encryption case takes a considerable amount of computation time, 1 second per 120 registrants, which makes it difficult to put into practice. The reason for this is the nature of the quasi-homomorphic encryption, which currently allows only four simple arithmetical operations with general calculation libraries if calculations are performed with the data still encrypted.In addition, since the feature data are transmitted in encrypted form, it cannot be sorted in advance. It cannot be calculated at high speed by making predictions of when the Euclidean distance is likely to be the shortest. When determining the Euclidean distance, all registrants must be examined one-to-one, making it difficult to utilize a fast algorithm for identification.

On the other hand, our method with noise was fast. It processed almost as quickly as the computation speed without noise. The correct response rate was also at a manageable level for practical use. Additionally, the authors demonstrated the possibility of identifying a specific person. This was done by classifying the features of the user's facial image and the features from the real-time camera image, which had been obscured by adding epsilon-noise of a common intensity. The upper graph of the percentage of correct responses when varying the epsilon from 0.5 to 9.5 dB in 100 registrants is shown in Table 2.4.

As shown in Fig. 2.8 the lower graph, again in this experiment, the correct practical response rate was reached where the epsilon exceeded 2.0. With 50 registered users, Fig. 2.8 the lower graph also shows a computation time of 0.004[ms], which may be considered sufficiently practical. The vertical right axis is the dB value of the PSNR of the 512-dimensional feature matrix before and after adding noise, and the horizontal axis is the epsilon intensity. epsilon values above 2.0 did not subjectively identify the person correctly when several people visually confirmed it. The figure shows that when the value of epsilon is greater than 2.0, the cosine similarity is greater than 0.6 and the PSNR value is greater than 50 dB. In the evaluation, the identification was considered successful when the PSNR value was greater than 50 dB and the cosine similarity was greater than 0.6. With a small number of users, the use of the Laplace transformation with the epsilon below 2.0 added noise and allowed for identification while maintaining privacy. In order to improve differential privacy, an approach to add epsilon-noise before image processing of facial landmark positions in the feature extraction process was also considered, and the facial features of the ONNX (Open Neural Network Exchange) models form of the insight face Buffalo-l model was used to identify the face of a person.

In the previous experiment, where the number of registrants was increased to 100 to estimate the range of epsilon values, rough predictions of the parameters were obtained. However, to acquire more accurate data, the measurement method was changed, and a new experiment was conducted. In this experiment, the number of registered participants was increased to 150 and the Euclidean distance was used to identify individuals rather than cosine similarity. Test data linked to specific individuals were used to calculate the FRR, and the results are shown in Fig. 2.9. From this figure, it was observed that unless the epsilon value is below 0.6, the FRR does not reach 80%, rendering it impractical.

As an additional point, In the trial experiment conducted before the above evaluation, when the

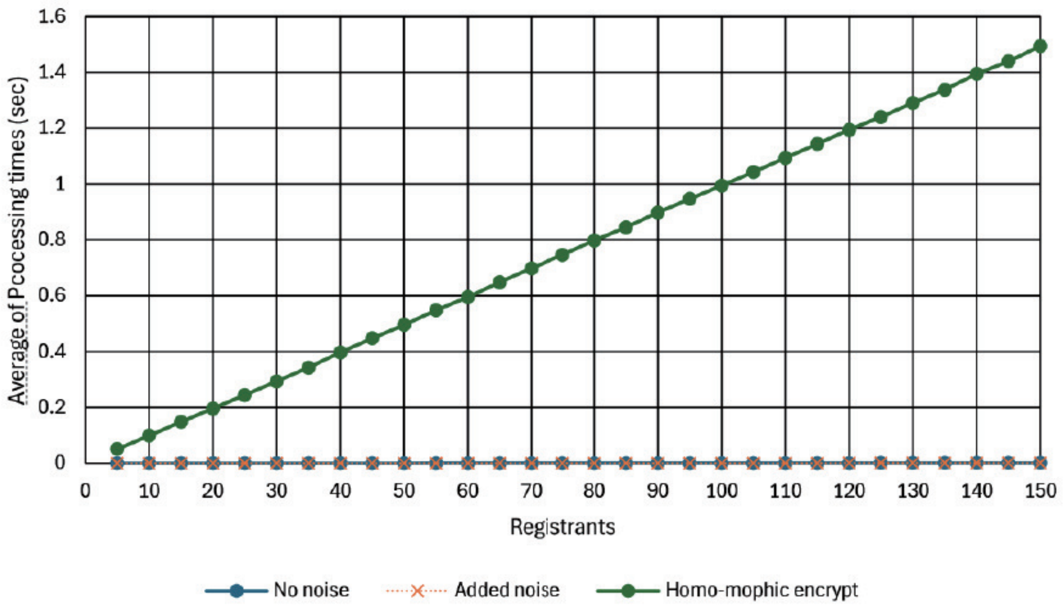
Table 2.3: **Peak Signal-to-Noise Ratio of 100 registrants**

epsilon	Homomorphic (FRR)	No noise (FRR)	Added noise (FRR)	Added noise (PSNR)
0.5	0.064	0.064	0.932	39.1298
1.0	0.064	0.064	0.38	45.15144
1.5	0.064	0.064	0.138	48.65487
2.0	0.064	0.064	0.084	51.17488
3.5	0.064	0.064	0.082	53.08487
4.0	0.064	0.064	0.068	54.68275
4.5	0.064	0.064	0.07	56.01378
5.0	0.064	0.064	0.068	57.19966
5.5	0.064	0.064	0.07	58.21072
6.0	0.064	0.064	0.066	60.70892
6.5	0.064	0.064	0.062	61.40307
7.0	0.064	0.064	0.064	62.04894
7.5	0.064	0.064	0.068	62.63217
8.0	0.064	0.064	0.064	63.22105
8.5	0.064	0.064	0.070	63.74255
9.0	0.064	0.064	0.066	64.22459
9.5	0.064	0.064	0.066	64.60249

Table 2.4: **A summary diagram of the time required for the preparation process**

Method	Average feature calculation (Time)	Average generation key (Time)	Average adding noise (Time)	Average encryption (Time)	Average transmission data (Time)	Total
Added noise	34.95	none	0.03	none	1.0	35.98
Homomorphic	34.95	87.51	none	2.55	1.0	126.01

Time:[msec]



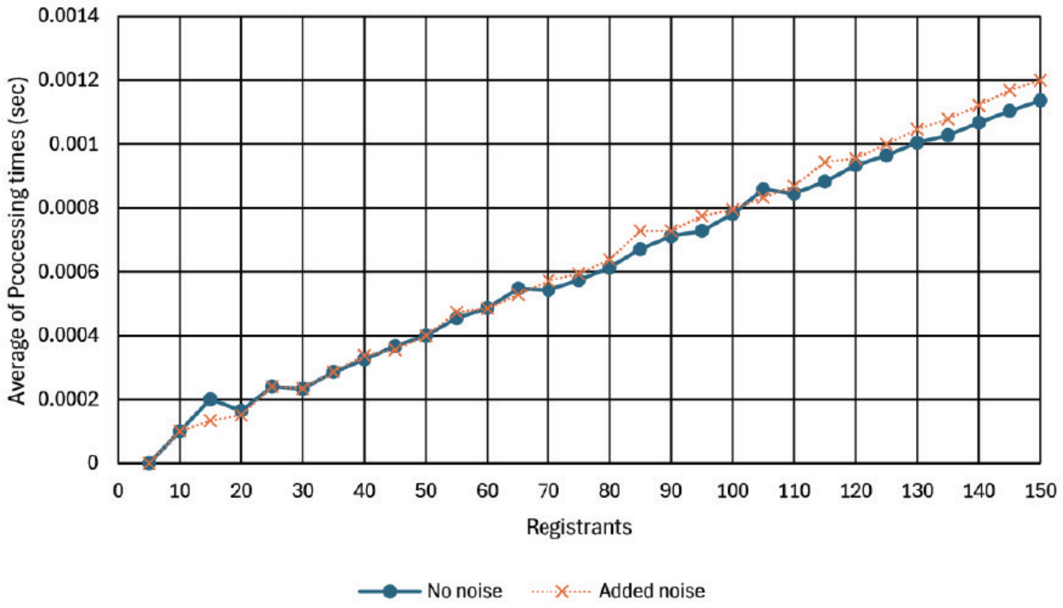


Figure 2.8: Processing time per number of registrants.

epsilon-noise strength was not the same between the two comparator cases, the cosine similarity was approximately 0.1, indicating that the person could not be identified. This indicated that the

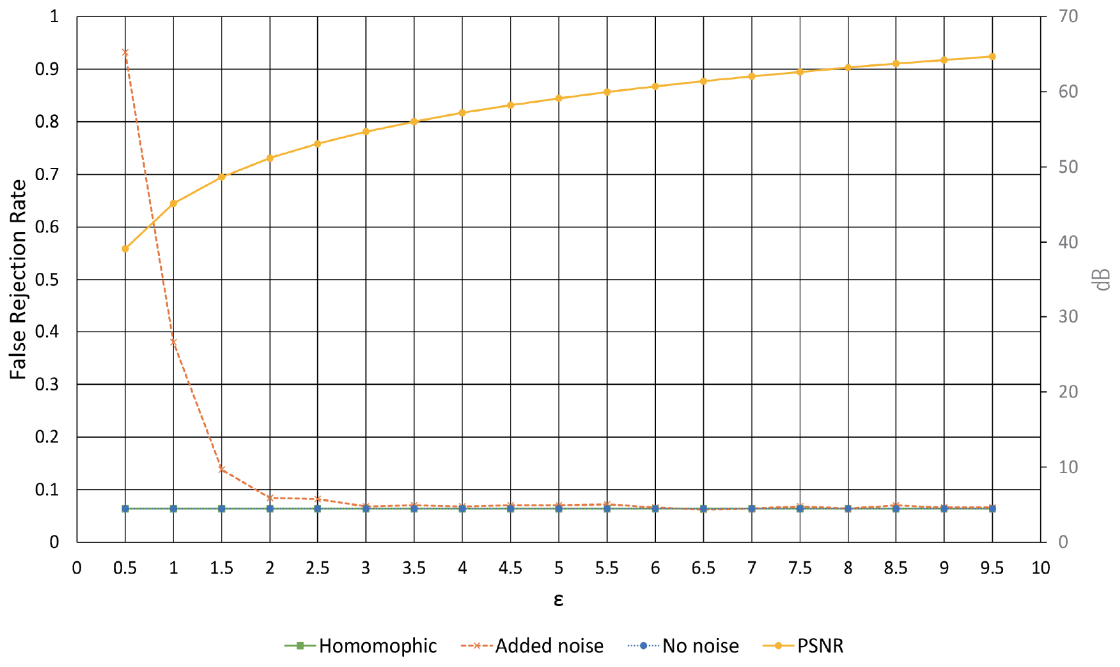


Figure 2.9: False rejection rate per epsilon.

person could not be identified. The classification of the person and the class using K-means began to deviate and the person could no longer be identified. The results of the verification using the common noise intensity and other cases are shown in the following subsection I. Verification in a larger dataset is a subject for future research, and we are currently investigating whether a distributed policy can achieve higher acceleration by grouping in a larger dataset.

2.4.5 Measured processing time results

In order to compare the overall calculation process, we calculated the average processing time to calculate the features for the face image of each registered user using each of the two methods. First, the average time taken to calculate the features from 7,500 face images and to register them in the database was 34.95 $msec$. Then, the average processing time for a person to add epsilon-noise to the facial features for 7,500 face images was 0.03 $msec$. For the quasi-isomorphic encryption process using the CKKS scheme, the average total time for key generation, key generation for the relinearization process, and CKKS context generation required for encryption and compositing was

87.51 $msec$ after 10 trials. The average time for the encryption process for a person with 7,500 facial features was 2.55 $msec$. Based on these results, the average computation processing time for one image until person identification when epsilon-noise is added, assuming a uniform data transmission time of 0.1 $msec$, is 34.95 + 0.03 + 1 $msec$, thus a total of 35.98 seconds. On the other hand, the average computation processing time for one person using the quasi-isomorphic encryption process with the CKKS scheme, assuming a uniform data transmission time of $1msec$, was 34.95 + 87.51 + 2.55 + 1 $msec$, thus a total of 126.01 $msec$. A summary diagram of the time required for the preparation process by each of the two methods is shown in Table 2.4. Considering the time required for the preparation process, it was found that the method using the quasi-isomorphic encryption process had a longer computation processing time than the method using the epsilon-noise addition method.

On a PC with a 12th generation Intel (R)Core(TM)i7 2.30GHz and an Nvidia RTX4070 Laptop, the average processing time per frame for face detection and feature computation was 0.031[sec], the cluster classification of 20 people took 2.75[sec], and processing all 7,500 frames took 318.37 [sec]. However, the combination of a 14th generation Intel(R) Core(TM) i9-14900K 3.20GHz and an Nvidia RTX4090 improved the overall performance of these processing times to 0.023[sec], 0.34[sec], and 222.35[sec], respectively. These results show that although the difference in PC performance is not significant for the frame-by-frame feature computation, the difference in PC performance clearly affects the processing time for cluster classification. In particular, as the number of participants increases, more powerful PCs tend to be required. However, 20 participants in a single tourist spot can be processed in real-time in less than 1 second on any of the PCs, and the user does not feel any load at present.

2.4.6 Verification of differences using the noise distribution function

The authors compared and evaluated which noise distribution most effectively reflects noise and impacts image quality. As a method, the PSNR and the False Rejection Rate (FRR) were calculated to assess the impact of adding noise to features for identification. The results showed that, while similar probabilistic outcomes are obtained with any noise distribution when the sample size increases significantly, within the range of individuals used in the proposed system, the Laplace distribution caused the greatest degradation in image quality. Therefore, the authors decided to adopt the Laplace distribution.

To determine which distribution is most beneficial, adjustments were made, as shown in equations (5)–(7), to maintain fairness, such as the width of the peaks for each distribution. Although all distributions would be equivalent if the sample size were infinite, in cases with limited samples, the Laplace distribution is considered somewhat easier to handle because of its longer tails.

The details of the evaluation methods are described in the next section.

$$\begin{aligned} \text{scale} &= \frac{\sqrt{\sigma^2_{\text{target}}/2}}{epsilon} \\ \text{noise} &\sim \text{Laplace}(0, \text{scale}) \\ \text{noisy\_features} &= \text{features} + \text{noise} \end{aligned} \tag{2.5}$$

$$\begin{aligned} \text{scale} &= \frac{\sqrt{\sigma^2_{\text{target}} \times 3}}{epsilon} \\ \text{noise} &\sim \text{Uniform}(-\text{scale}, \text{scale}) \\ \text{noisy\_features} &= \text{features} + \text{noise} \end{aligned} \tag{2.6}$$

$$\begin{aligned} \text{scale} &= \frac{\sqrt{\sigma^2_{\text{target}}}}{epsilon} \\ \text{noise} &\sim \mathcal{N}(0, \text{scale}^2) \\ \text{noisy\_features} &= \text{features} + \text{noise} \end{aligned} \tag{2.7}$$

2.4.7 Epsilon and Peak Signal-to-Noise Ratio (PSNR)

The authors evaluated the extent to which features remain visually identifiable when noise is added, using PSNR as the evaluation metric.

Initially, only a ResNet50 and U-Net-based encoder-decoder model was used. This model reconstructed images from input features, but it was heavily dependent on the training data. Consequently, when attempting to generate images from features of unseen data, the model was unable to reconstruct the images accurately, resulting in outputs that resemble collages of fragments from previously trained faces. This issue arose due to the model's tendency to overfit to the training data, leading to a lack of generalization capability. To address this problem, the authors improved the model by incorporating training based on the GAN into the decoder. GANs leverage competitive learning between the generator and the discriminator to acquire the ability to generate realistic images. The generator, in its effort to deceive the discriminator, becomes capable of producing plausible images even from features that are not present in the training data. This process enhances the GAN's ability to model complex data distributions, enabling it to reconstruct high-quality images from previously unseen data. Consequently, by integrating GAN-based training, the model was improved to accurately restore original images even from features of unseen data.

As discussed in the referenced work by Yang et al. (2020) [16], GAN-based reconstruction attacks pose significant threats due to their capability to generate realistic images from incomplete or obfuscated data. By leveraging the generator to produce plausible images and the discriminator to evaluate their authenticity, attackers can reconstruct high-quality images from leaked facial feature vectors. Such attacks, as indicated in the Introduction, can lead to severe privacy risks, including the unauthorized creation of 3D facial models, unlocking electronic devices, and producing deepfake content.

The verification method was evaluated using a module with specific functions. We implemented a GAN designed for image generation tasks, utilizing face embeddings extracted by InsightFace. The GAN comprises a generator and a discriminator, both defined using PyTorch's neural network module. The generator employs a U-Net-style decoder to reconstruct images from latent embeddings, while the discriminator uses convolutional layers to distinguish between real and generated images.

A key feature of our method is the addition of noise during feature convolution. To evaluate the robustness and performance of the GAN under various noise conditions, we incorporated predefined functions for Laplace, uniform, and normal noise. In addition, a PSNR calculation function was included to quantitatively assess the quality of the generated images.

To facilitate the loading and preprocessing of image data, we created a custom dataset class and used InsightFace to extract face embeddings. The dataset was divided into training and testing sets, with a data loader allowing efficient batching during training. The training function alternates between updating the generator and the discriminator: the discriminator classifies real images as real and generated images as fake, while the generator learns to create images that the discriminator classifies as real.

After training, the generator was evaluated on the test dataset, and the generated images were compared to the real images using PSNR. The results, along with the detected face bounding boxes, visually and quantitatively demonstrated the performance of the GAN. The noise distribution function was determined based on the PSNR and epsilon figures, and the Laplace distribution was selected, which showed the greatest degradation in image quality compared to the original image. The results of the evaluation based on this block diagram are shown in Fig. 2.10. The results are shown in Fig. 2.11 and Fig. 2.12.

2.4.8 Evaluation of Noise Resistance and Identification Performance

As part of this research, we compared the noise resistance and identification performance of our proposed "N-System," formally referred to as the Noise-Obfuscated Feature Protection System (NOFPS), and the adversarial "A-System," also known as the Adversarial Feature-to-Image Generation System (AFIGS). The N-System is designed to protect personal information by adding Laplace noise to the features, thereby obfuscating the original feature representations. In contrast, the A-System simulates an attacker attempting to regenerate the original image from the obfuscated features using a ResNet50 for feature extraction and a GAN-based autoencoder incorporating U-Net architecture for learning and generating fake images from the features.

Comparison of Noise Resistance and Identification Performance in the N-System and A-System

Since it is assumed that an attacker would find it difficult to directly use the complex feature extraction model employed in the N-System, a simplified model was constructed for the A-System for evaluation. Although the complex model of the N-System could not be directly used, we clarified the strengths and weaknesses of both systems by analyzing the impact of noise on the attack

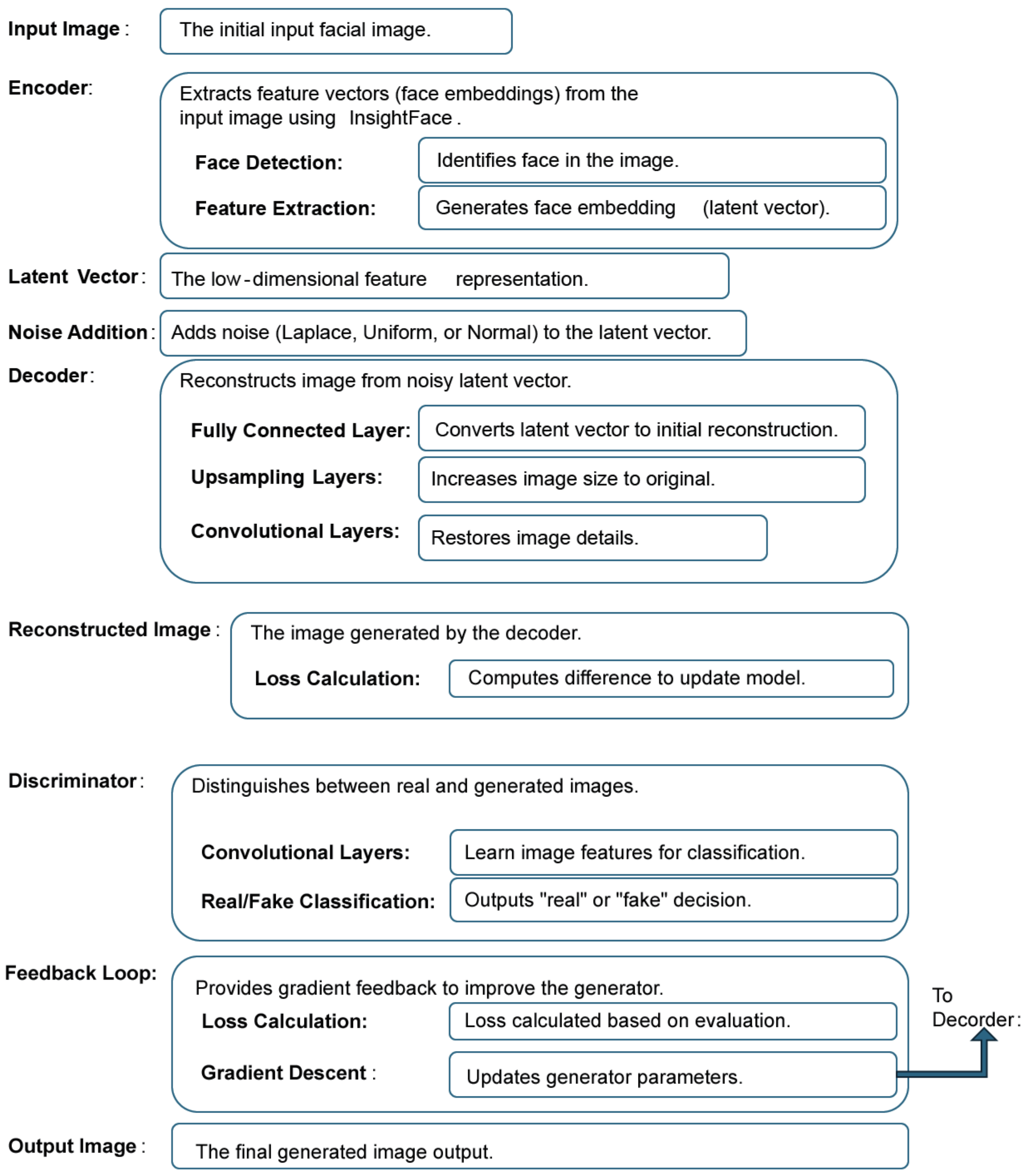

Input Image :
The initial input facial image.
Encoder:
Extracts feature vectors (face embeddings) from the input image using InsightFace .
Face Detection:
Identifies face in the image.
Feature Extraction:
Generates face embedding (latent vector).
Latent Vector:
The low-dimensional feature representation.
Noise Addition:
Adds noise (Laplace, Uniform, or Normal) to the latent vector.
Decoder:
Reconstructs image from noisy latent vector.
Fully Connected Layer:
Converts latent vector to initial reconstruction.
Upsampling Layers:
Increases image size to original.
Convolutional Layers:
Restores image details.
Reconstructed Image :
The image generated by the decoder.
Loss Calculation:
Computes difference to update model.
Discriminator:
Distinguishes between real and generated images.
Convolutional Layers:
Learn image features for classification.
Real/Fake Classification:
Outputs "real" or "fake" decision.
Feedback Loop:
Provides gradient feedback to improve the generator.
Loss Calculation:
Loss calculated based on evaluation.
Gradient Descent :
Updates generator parameters.
To Decorder:
Output Image :
The final generated image output.

Figure 2.10: module block.

resilience and image regeneration accuracy of the A-System.

The N-System, as a defensive mechanism, adds Laplace noise to the feature representations, aiming to protect the features as personal information. Hence, it is crucial to maintain a high True

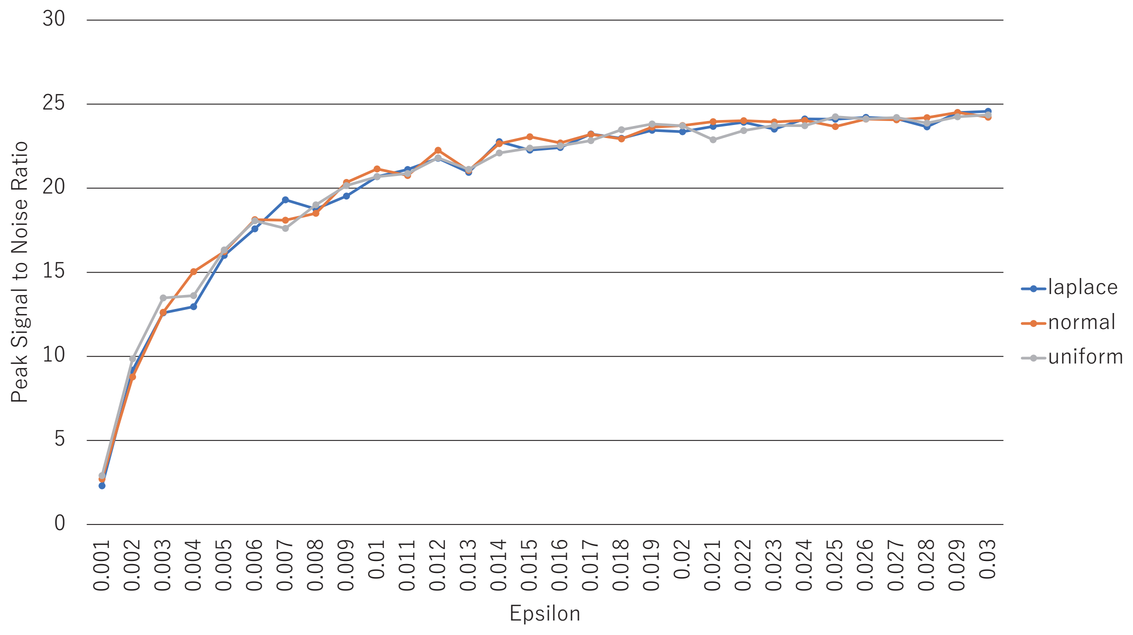


Figure 2.11: epsilon - PSNR.

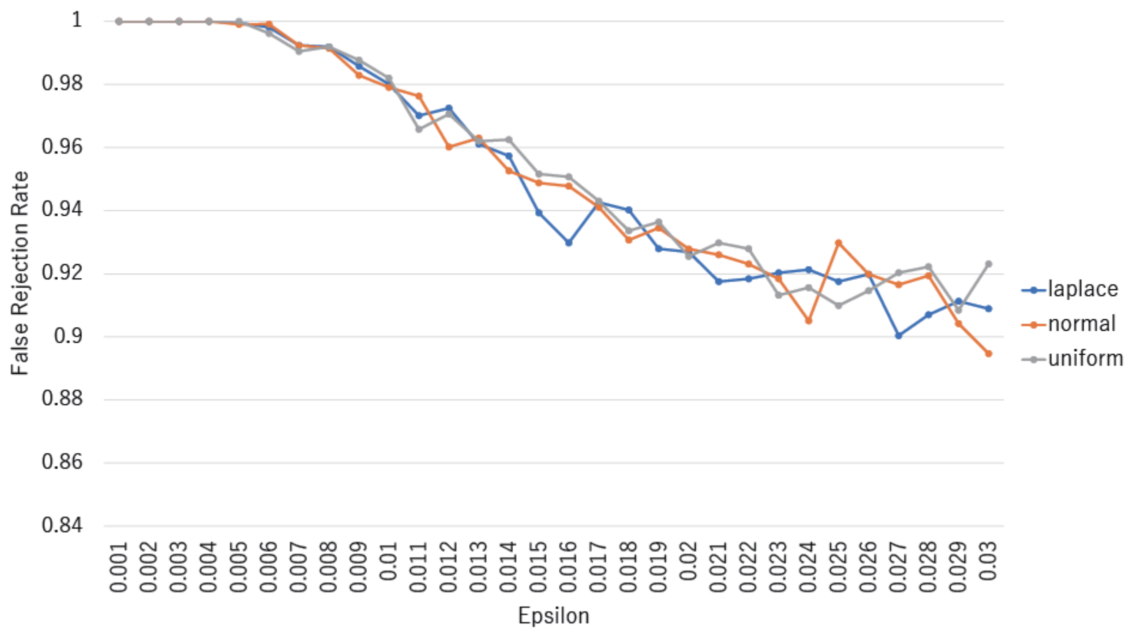


Figure 2.12: epsilon - FRR.

Acceptance Rate (TAR) while keeping the False Acceptance Rate (FAR) low, which is achieved by increasing the complexity of the features. On the other hand, the A-System, as an adversarial entity, aims to regenerate the original image from the noisy features and compromise the privacy of the feature data. The accuracy of its regeneration is evaluated using the PSNR.

Figure 2.13 shows the evaluation results using approximately 500,000 images extracted from video footage of 504 individuals.

To verify the effectiveness of the proposed NSystem in improving image clarity, a one-sided t-test was conducted comparing the PSNR values of N-System and A-System. The results of the t-test show a statistically significant difference with a t-statistic of 35.57 and a p-value of 5.92×10^{-180}.

The null hypothesis, which states that there is no significant difference between the PSNR values of N-System and A-System, was rejected at the 0.05 significance level. This indicates that the proposed N-System achieves significantly higher image clarity compared to A-System.

The TAR values displayed on the figure suggest that the complexity of the feature representations significantly impacts the accuracy of the identification. In the N-System, further complicating the features based on the simpler model of the A-System greatly improved noise resistance. For example, even when the noise intensity ranged from 0.1 to 0.6, the TAR of the N-System consistently remained above 90%, indicating high stability. Furthermore, even when the noise intensity increased to 0.8, the TAR remained above 80%, maintaining practical accuracy. This suggests that the N-System significantly reduces the risk of external attacks, particularly the risk of feature restoration attempts posed by the A-System. In contrast, in the A system, the TAR decreased sharply once the noise intensity exceeded 0.2, dropping below 50% at 0.4 and approaching random levels at 0.6. This indicates that the A-System, which relies on simpler feature or image comparisons, is ineffective against complex features and noise, resulting in a significant decline in identification accuracy.

Regarding FAR and False Rejection Rate (FRR), it was confirmed that the complex features of the N-System almost never cause incorrect identification. In the noise intensity range of 0.1 to 0.6, the FAR was close to 0%, and the FRR remained very low. Even at a noise intensity of 0.8, the FAR remained below 1%, demonstrating the N-System's strong defense. In contrast, the FAR in the A-System gradually increased as the noise intensity increased, exceeding 30% at a noise intensity of 0.4, and worsening further beyond 0.6. This shows that it becomes increasingly difficult for the A-System to accurately regenerate the original image.

Evaluation of Image Regeneration Accuracy Using PSNR and the Effect of Laplace Noise Intensity

In terms of PSNR-based quality evaluation, the N-System maintained more than 40 dB even at a noise intensity of 0.6, confirming that the feature matrix comparison functions effectively. In contrast, the PSNR of the A-System rapidly decreased as noise intensity increased, falling below 30 dB at 0.4 and decreasing further at 0.6. In particular, when the noise intensity exceeded 0.8, the PSNR dropped below 20 dB, showing almost no resemblance to the original image.

Regarding the effect of Laplace noise intensity, the N-System demonstrated stable identification performance within the noise intensity range of 0.1 to 0.8, maintaining high TAR and PSNR values. This indicates that the N-System is highly resistant to noise and is highly effective in protecting personal information. In contrast, the performance of the A-System rapidly deteriorated once the noise intensity exceeded 0.4, and the regeneration of the original image became almost impossible beyond 0.8. These results indicate that adding Laplace noise serves as an effective defense against external attacks, neutralizing the A-System's attempts to regenerate the original image. In particular, if the epsilon value is below 0.4, both security and identification accuracy are ensured. However, verification is likely required for each facial identification model. In this paper, we used the Buffalow-l model of Insightface.

Security Analysis Based on Epsilon Intensity and Identification Performance

To assess the security and identification performance of the N-System, we conducted a one-way analysis of variance (ANOVA) on the True Acceptance Rate (TAR) values across different noise intensities (epsilon). The epsilon values were divided into five groups to analyze the impact of varying noise levels on both identification accuracy and privacy protection.

The key security objective of the N-System is to prevent adversaries from reconstructing original facial images while maintaining high identification accuracy. The system achieves this by adding Laplace noise to feature representations, effectively reducing the risk of external attacks.

Table 2.5 shows the results of the ANOVA test, confirming that there is a statistically significant difference in TAR values across all epsilon groups. The F-statistics and p-values indicate that as epsilon increases, the N-System maintains a high TAR while offering stronger protection against image reconstruction attacks.

Table 2.5: ANOVA Results for Different Epsilon Ranges and Security Implications

Epsilon Group	**F-statistic**	**p-value**	**Security Implication**
Group 1 ($0.0 \leq \epsilon < 0.2$)	605482.78	0.0	High risk of image reconstruction
Group 2 ($0.2 \leq \epsilon < 0.4$)	37521.87	0.0	Moderate risk, improved protection
Group 3 ($0.4 \leq \epsilon < 0.6$)	5652.16	2.40×10^{-235}	Optimal balance between security and accuracy
Group 4 ($0.6 \leq \epsilon < 0.8$)	11710.14	3.22×10^{-283}	Strong security with acceptable accuracy
Group 5 ($\epsilon \geq 0.8$)	60915.22	0.0	Maximum security, reduced accuracy

The results highlight the trade-off between identification accuracy and privacy protection. The N-System demonstrates robust security across all noise intensity levels. Specifically, within the range of $0.4 \leq \epsilon < 0.6$, the system achieves an optimal balance, maintaining a high TAR while minimizing the risk of adversarial attacks.

As the noise intensity increases, the risk of feature reconstruction by adversaries, such as the A-System, decreases significantly. This is particularly evident in Group 4 and Group 5, where higher noise levels make it almost impossible for adversaries to regenerate original facial images. In contrast, lower epsilon values (Group 1) pose a higher risk of privacy compromise due to insufficient noise.

Overall, these findings confirm that the N-System effectively neutralizes potential image reconstruction attacks by adding Laplace noise to feature representations, ensuring both high identification performance and strong privacy protection.

2.4.9 Evaluation of differential noise intensity

We compared the TAR, FAR, and FRR between features extracted from facial images with noise based on Laplace distributions. For the feature set that simulates the registered information, noise was added exactly 1 time the original features, and for the feature set simulating footage captured by public cameras, noise was added on a varying scale from 0.2 to 2 times the original features.

We initially hypothesized that comparing features with 1x noise on both sides would result in the highest recognition accuracy for the same individual. However, the figure shows different results, as seen in Fig. 2.14. The red line in the center represents the comparison of features with 1x noise on both sides.

Impact of Epsilon on TAR, FAR, and FRR

Several interesting trends are observed in Fig. 2.15.

Decline in TAR: Looking at the trend of TAR, it generally decreases as the epsilon increases. However, the red line (1x comparison) does not perform as expected. Instead, comparisons between different noise levels (such as 0.2 to 0.6 times) sometimes show a higher TAR than the 1x comparison. It appears that at 0.2, the noise is neither too much nor too little, creating a ”sweet spot” where the TAR remains optimal.

Stability of FAR and FRR: Regarding the trends in FAR and FRR, the red line (1x comparison) seems to be more stable than the other noise levels. This suggests that when comparing features with different noise levels, there are likely to be more false acceptances (increased FAR) or rejections (increased FRR).

Intersection around epsilon 0.334 to 0.445: Several lines intersect around epsilon 0.334 to 0.445. At this intersection, the TAR varies sharply between different noise intensities, with the 1x comparison (gray line) converging with other noise intensity comparisons. This suggests that with moderate noise intensity, the comparison of features with different noise levels may result in a higher match rate.

Degradation at Higher Noise Levels and Unexpected Results

Degradation at higher epsilon : Beyond epsilon 0.667, TAR declines sharply, while both FAR and FRR increase noticeably. At this point, performance deteriorates in all conditions, including the 1x comparison (gray line). This indicates that the influence of Laplace noise becomes too strong, making it difficult to accurately identify features.

Unexpected results: Although the 1x comparison was expected to perform the best, comparisons with other noise intensities show a higher TAR in certain ranges. Possible reasons include:

Noise overfitting: In the 1x comparison, the added noise may exaggerate small differences between features, resulting in overfitting. Noise variability enhancing accuracy: In comparisons involving different noise levels, the variability introduced by the noise might reduce differences between features, leading to improved recognition accuracy. Nonlinear noise tolerance: The system's tolerance to noise might not follow a linear pattern. Certain combinations of noise levels may exhibit more robust performance.

Conclusion and Implications for the N-System

The figures shown in this subsection demonstrate that comparing features with 1x noise does not always yield the best results. In the moderate range of epsilon (0.2 to 0.6), comparisons between different noise intensities show better results in some cases, indicating a complex relationship between noise intensity and recognition accuracy. At higher epsilon values, overall performance declines. Therefore, careful consideration of noise intensity is necessary when deploying the N-System in practical scenarios.

Upon considering why the comparison between 1.0 and 1.0 is stable, as shown in Fig. 2.15, when the noise intensity was further reduced to 0.2, the performance dropped to the lowest level, confirming that 1.0 is the most stable. However, selecting the optimal noise intensity is challenging as certain combinations may lead to a decline in recognition accuracy. Additionally, when comparing different noise levels, the variation introduced by noise can sometimes smooth out differences between features, resulting in improved recognition accuracy. Furthermore, it is possible that the noise tolerance of the system is non-linear, which means that its resistance to added noise may not follow a simple linear pattern.

In conclusion, it was confirmed that the N-System can strongly protect personal information by adding Laplace noise to the feature representations while maintaining high identification performance. The N-System, evidenced by its stable TAR and FAR even at high noise intensities, is a highly effective defensive system that significantly reduces the risk of information leakage.

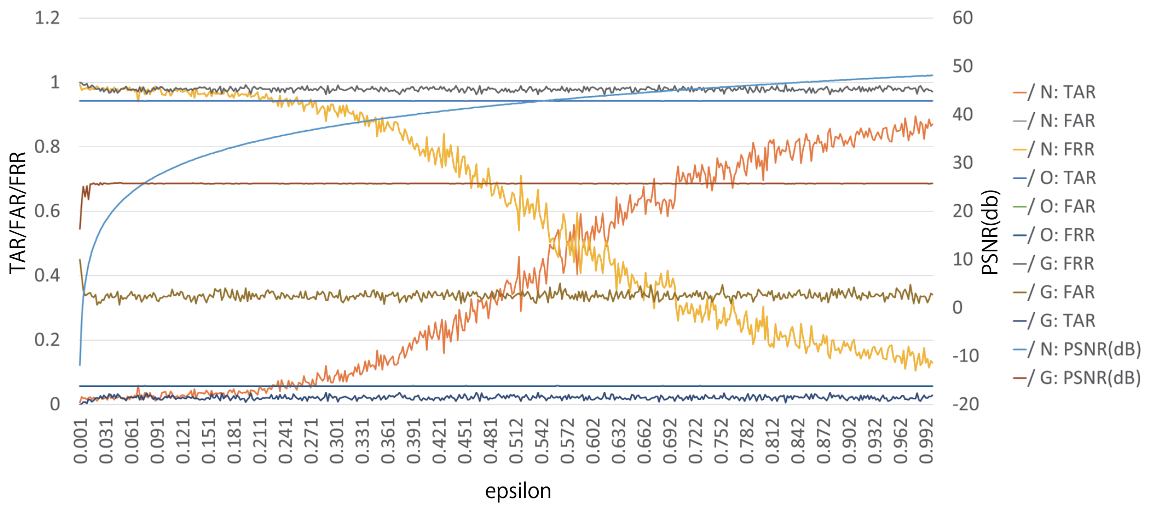


Figure 2.13: epsilon - TAR FAR FRR.

However, the ability of the A-System to regenerate the original image from the feature representations deteriorates sharply with increasing noise intensity, making it vulnerable to the defense of the N-System. This confirms that the N-System serves as a powerful defensive measure, balancing personal information protection with identification accuracy while safeguarding against attacks from systems like the A-System.

2.5 Conclusion

In recent years, the use of facial recognition in authentication systems has been increasing, along with concerns about privacy violations due to misuse of facial images and feature data. Facial recognition can be performed with encrypted feature data using quasi-homomorphic encryption, but it is computationally expensive. In this paper, we propose a system that allows personal identification in public spaces while protecting privacy.

We are developing a next-generation vlog system for modeling evaluation that can take pictures of passersby using cameras installed at sightseeing spots, blur the face images of all passersby,

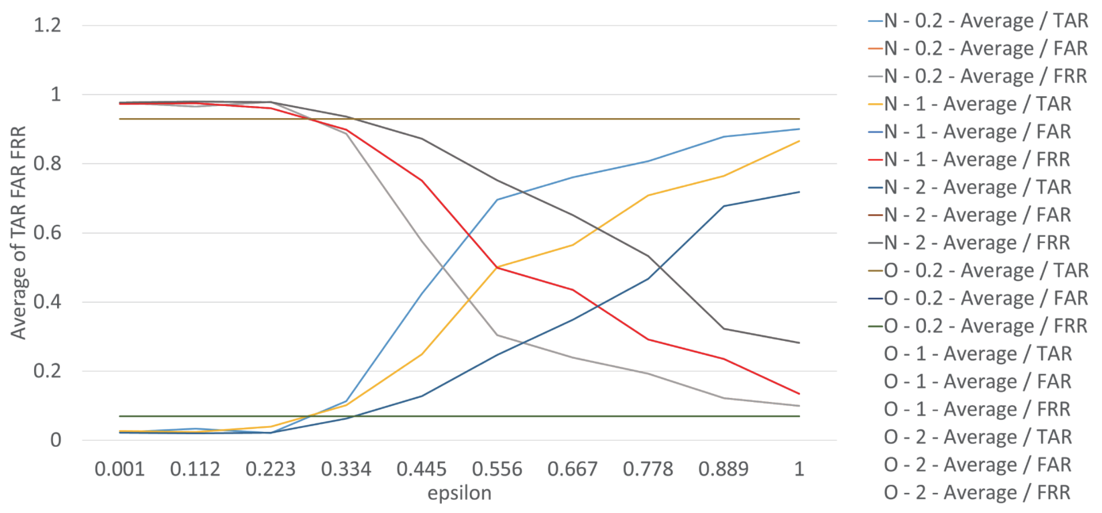


Figure 2.14: epsilon - differential noise TAR FAR FRR.

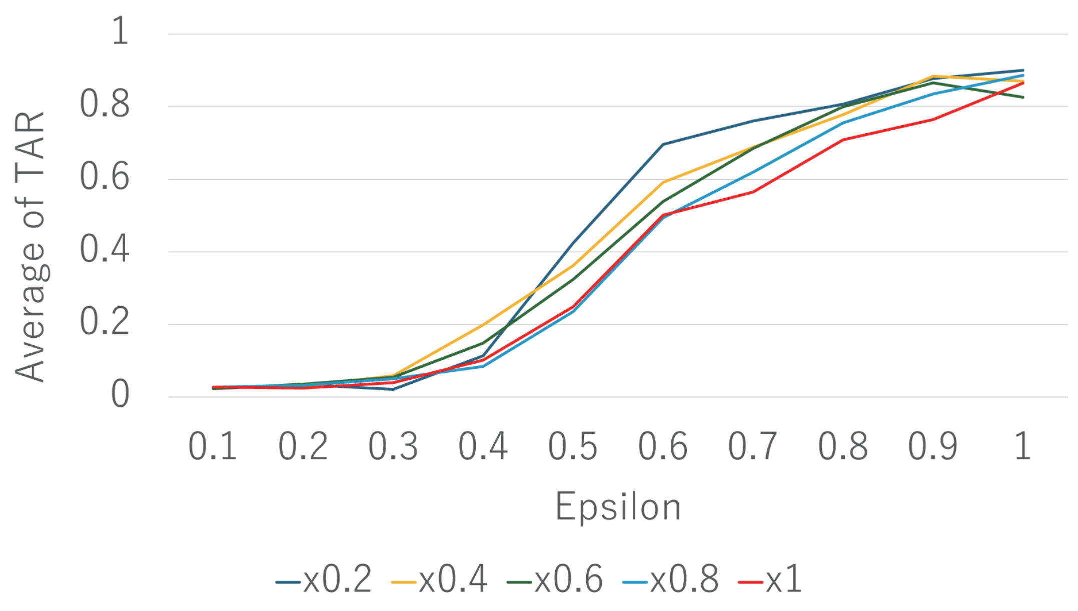


Figure 2.15: epsilon - differential noise.

remove the blurring for specific persons whose face images are registered, and upload the images to a vlog in real time in a hands-free manner. We conducted a similar experiment by increasing the amount of test data in the dataset and using the Euclidean distance between feature values to measure similarity.

The results showed that the system could identify individuals when noise was within the range of 0.2 to 0.6. Specifically, when the epsilon value was below 0.4, both security and identification accuracy were maintained. We also compared this method with homomorphic encryption, which does not require decoding and encoding overhead, and found that homomorphic encryption was impractical due to its extensive computational demands for similarity comparison. In recent years, the use of facial recognition in authentication systems has been increasing, along with concerns about privacy violations due to misuse of facial images and feature data. Facial recognition can be performed with encrypted feature data using quasi-homomorphic encryption, but it is computationally expensive. In this paper, we propose a system that allows personal identification in public spaces while protecting privacy.

We are developing a next-generation vlog system for modeling evaluation that can take pictures of passersby using cameras installed at sightseeing spots, blur the face images of all passersby, remove the blurring for specific persons whose face images are registered, and upload the images to a vlog in real time in a hands-free manner. We conducted a similar experiment by increasing the amount of test data in the dataset and using the Euclidean distance between feature values to measure similarity.

The results showed that the system could identify individuals when noise was within the range of 0.2 to 0.6. Specifically, when the epsilon value was below 0.4, both security and identification accuracy were maintained. We also compared this method with homomorphic encryption, which does not require decoding and encoding overhead, and found that homomorphic encryption was impractical due to its extensive computational demands for similarity comparison.

Chapter 3

ECA Rules for Distributed Mask Video Processing to Ensure Privacy Protection

In future societies, communication through video is expected to become increasingly prevalent. For instance, it is becoming common to broadcast video logs (vlogs) in real time to a large and unspecified audience as a form of communication.

Platforms like YouTube enable content creators to deliver live streams or provide viewers with pre-recorded on-demand videos. Similarly, viewers have widely adopted practices such as watching live broadcasts and consuming asynchronous stored videos.

The traditional one-way broadcasting model, where the same content is delivered to all viewers as seen in conventional television, is evolving. The new model emphasizes two-way video communication, where individual privacy is considered during video collection and personalized content is efficiently provided during distribution based on individual needs.

There are two primary challenges in video distribution. The first is addressing privacy concerns tailored to individuals. The second is resolving scalability issues based on the audience size associated with individual broadcasters.

This chapter focuses on optimizing video distribution according to viewers' perspective choices during streaming. A distributed masked video processing system was implemented using Event-Condition-Action (ECA) rules to achieve this optimization.

3.1 Introduction

With the recent popularization of omnidirectional cameras, multi-viewpoint live videos are often broadcast through the Internet. In multi-viewpoint Internet live broadcasting services, viewers can arbitrarily change their viewpoints. For example, major live broadcasting services such as YouTube Live and Facebook provide 360° videos in which each user can select their desired viewpoint. In recent Internet live broadcasting services, viewers or broadcasters have been able to add video or audio effects to the broadcast videos. To reduce the computational load including them for adding such effects, a number of distributed Internet live broadcasting systems have been developed [17], [18]. These systems are designed for single-viewpoint live videos, and the screen images (images to be watched by viewers) are the same for all viewers. Therefore, screen images can be shared among processing servers, and the computational load can be reduced by exploiting distributed computing systems. However, in multi-viewpoint Internet live broadcasting services, the screen images differ according to the viewpoint selected by the user. Thus, screen images cannot be shared among processing servers. Here, one of the main research challenges for multi-viewpoint Internet live broadcasting systems is how to reduce the computational load required to add effects under different screen images. In this paper, focusing on this challenge, we propose and develop a distributed multi-viewpoint Internet live broadcasting system. In our proposed system, video effects that can be shared among viewers are added by some distributed processing servers (i.e., on the server side). Video effects that cannot be shared among viewers are added by video players (i.e., on the player side). In such systems, it is difficult to determine whether it is better to add effects on the server side or player side. To determine this so as to effectively reduce the computational load, we use grouped rules. Moreover, we develop a distributed multi-viewpoint Internet live broadcasting system adopting our proposed rules system. The remainder of this paper is organized as follows. In Section 3.2, we introduce some related work. We describe the design and the architecture of our proposed system in Section 3.3. Evaluation results are presented in Section 3.4 and discussed in Section 3.5. Finally, we conclude this paper in Section 3.6.

3.2 Related Work

An Internet live broadcasting system that allows the viewing of recently recorded videos (playback) was proposed in [19]. Several methods have been proposed for reducing the delay time for the distribution of videos in live Internet broadcasting. SmoothCache 2.0 [20], video data from other peers are cached and distributed among a P2P network. As a result, the communication load and delay times are reduced. Dai et al. also proposed a distributed video broadcasting system using P2P networks to reduce delay times [21]. In the HD method proposed in [22], communication traffic is reduced by simultaneously transmitting image data to a number of viewers using one-to-many broadcasting with one-to-one communication. Even in our proposed system, these delay reduction methods can be applied when delivering videos, but our current research considers the addition of video or audio effects. Gibbon et al. proposed a system that performs video processing by transferring the data captured by a camera to a computer with high processing capabilities [23]. Ting et al. proposed a system that directly stores images captured by computers with low processing power in external storage devices, such as cloud storage [24]. However, these systems target stored video data and cannot be applied to live Internet broadcasting. J. Bae et al. proposed a concept of blocks to classify processing flows into several patterns. A block is a minimal unit that specifies the behaviors represented in a process model [25]. A. Frömmgen et al. proposed a learning algorithm of complex nonlinear network nodes by genetic algorithm and ECA rules in [26]. As described in these papers, it is important to learn effective sequences to execute ECA rules. These are not focused on multi-viewpoint image processing. We propose a model focused on image processing with multi-viewpoint image processing.

Some systems for distributing video processing loads have been proposed. Most of them fix load distribution procedures in advance. However, starting Internet live broadcasting is easy in recent years, and it is difficult to grasp which machines start Internet live broadcastings. Therefore, conventional systems establish load distributions at server side. MediaPaaS encodes, re-encodes, and delivers video using a server machine provided by cloud computing services [18]. Different from MediaPaas, our proposed system establishes load distributions using PIAX [27], a P2P agent platform. The system has multiple servers to broadcast videos, and once a client (video recording terminal) connects to a server to broadcast its recorded video, one of the servers is randomly selected by the load distribution server. The loads caused by broadcasting videos are distributed among the servers. In [17], we confirmed that the video processing time for encoding and distributing videos can be reduced by distributing the processing load to some servers.

3.3 Distributed Internet Live Broadcasting System

In this section, we explain our previously developed cloud-based live broadcasting system using ECA (event, condition, action) rules. After that, we explain our proposed multi-viewpoint Internet live broadcasting system.

3.3.1 Different World Broadcasting System

Summary

In our previous research [17], we constructed a differentworld broadcasting system using virtual machines (VMs) provided by a cloud service. These machines work as the different world broadcasting servers that add video effects. In general, a number of VMs can easily be used in a cloud service. The use of multiple VMs as different world broadcasting servers enable a high-speed addition of effects by distributing the load among different world broadcasting servers. Therefore, we implemented a distributed live Internet broadcasting system using the cloud service and evaluated its performance. In our developed system, video effect additions are executed on the VMs provided by the cloud service. Processing loads on different world broadcasting servers can be distributed by considering the load distribution when selecting a server. In conventional systems, load distribution is established by connecting processing servers via a load balancing mechanism such as a load balancer. In this method, when the load distribution mechanism needs to switch to another server while the video is being transmitted, the connection is interrupted. For this reason, it is difficult to switch servers while keeping smooth video plays. Therefore, in the different world broadcasting system, the load balancing mechanism selects a different world broadcasting server based on the requests.

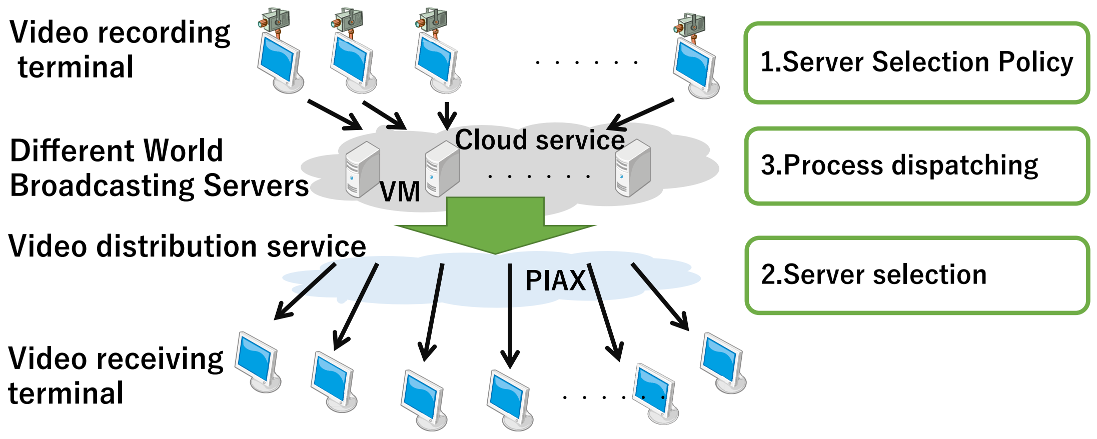


Figure 3.1: System architecture of the different world broadcasting system.

System Architecture

The system architecture of the different world broadcasting system is shown in Fig.3.1 There are three types of machine. The first is the client, which has cameras and records live videos. The second is the different world broadcasting servers, which execute processes for videos such as encoding, decoding, or video effect additions. The third type is the viewer, which plays the live videos. Each client selects a different world broadcasting server that executes the desired video effect, and transmits the video effect library and the recorded video to the different world broadcasting server. The different world broadcasting server is a VM of the cloud service that executes video processing on the video transmitted from the clients according to their requests. The video processed by the different world broadcasting server is delivered to the viewers via the video distributions service. In the system, viewers receive the processed video after selecting the server or channel of the video distributions service.

Load Distribution Mechanism

Figure 3.2 shows the load distribution mechanism of our developed system. The client software and the client side PIAX system are installed in the client. The different world broadcasting server software and the server side PIAX system is installed on the different world broadcasting servers. PIAX [27] is an open-source, Java-based platform middleware that enables efficient server resource

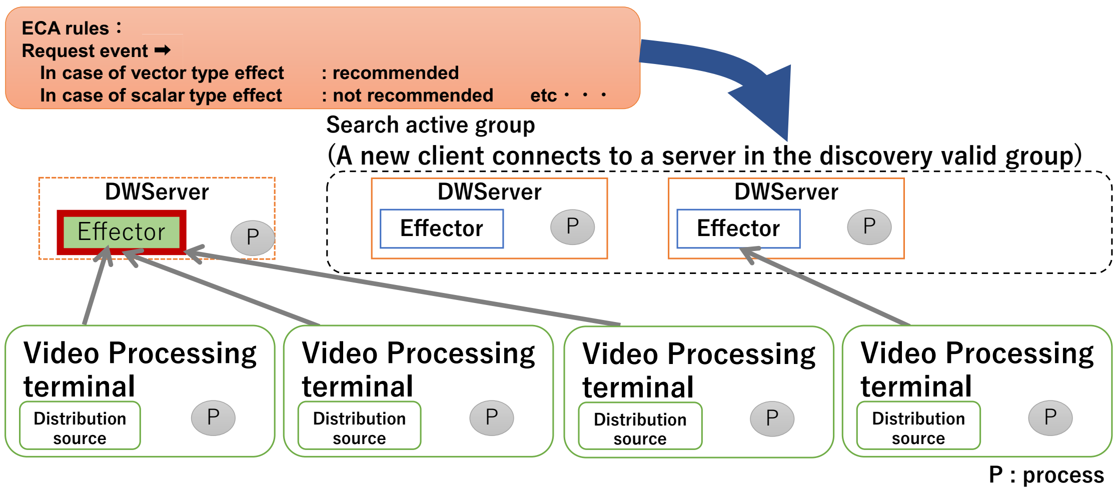

ECA rules：
Request event ➡
In case of vector type effect : recommended
In case of scalar type effect : not recommended etc・・・
Search active group
(A new client connects to a server in the discovery valid group)
DWServer
Effector
P
DWServer
Effector
P
DWServer
Effector
P
Video Processing terminal
Distribution source
P
Video Processing terminal
Distribution source
P
Video Processing terminal
Distribution source
P
Video Processing terminal
Distribution source
P
P : process

Figure 3.2: Load distribution mechanism using PIAX.

searches using the resource search function of the overlay network. The PIAX systems used by the client and the different world broadcasting servers connect with each other via the overlay network. The client side PIAX system searches the overlay network according to the client software requests. The system selects a different world broadcasting server from the list, and then the client side PIAX system returns the IP address of the selected server and listens to the stated port number of the server software. The client software then establishes a connection with the different world broadcasting server and starts transmitting the video. New connections from the client are controlled based on the load state of the different world broadcasting server.

3.3.2 Information considered for model partitioning and allocation

The notations for the information considered for DL model partitioning and allocation are

3.3.3 Extension of Different World Broadcasting System

Figure 3.3 shows an overview of our designed multi-viewpoint Internet live broadcasting system. As shown in the figure, our system converts the coordinates of videos from polar to rectangular when different world broadcasting servers execute processing. After that, the video images are delivered to viewers. In this section, we first explain image conversion of multi-viewpoint videos

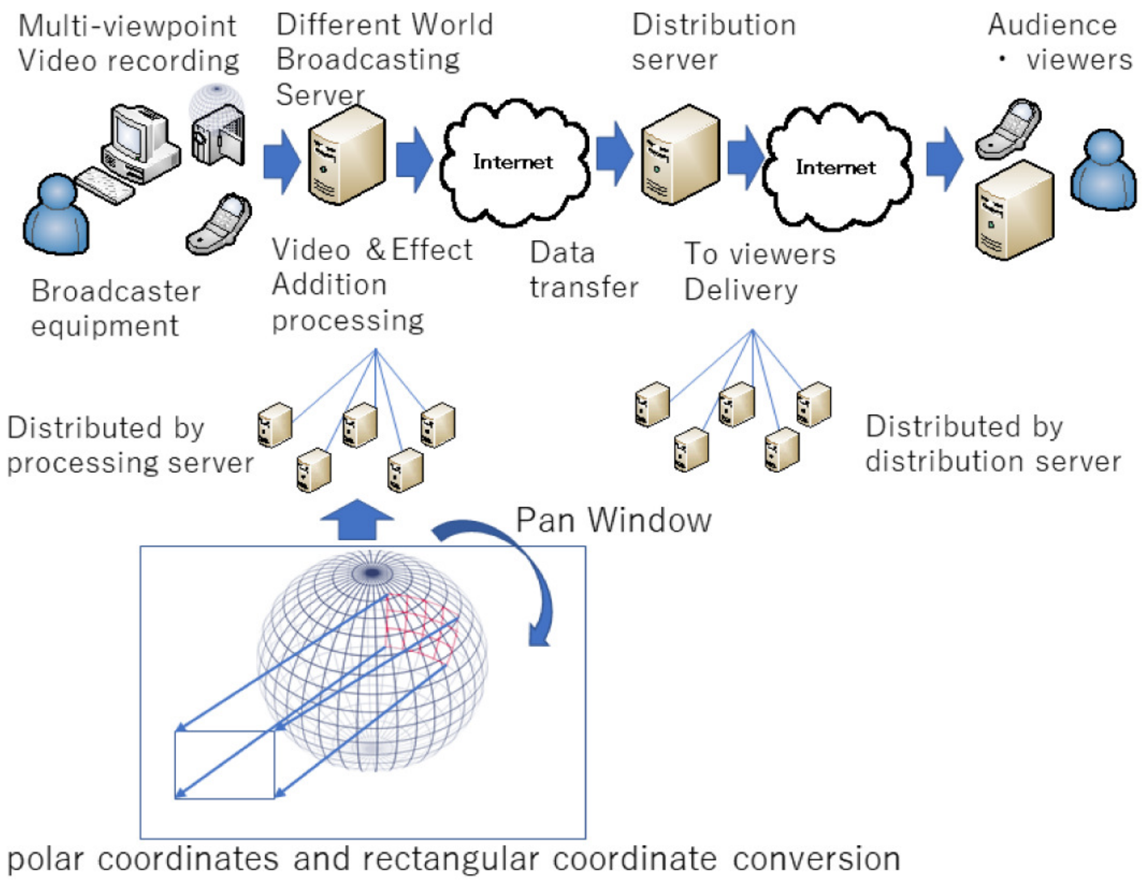
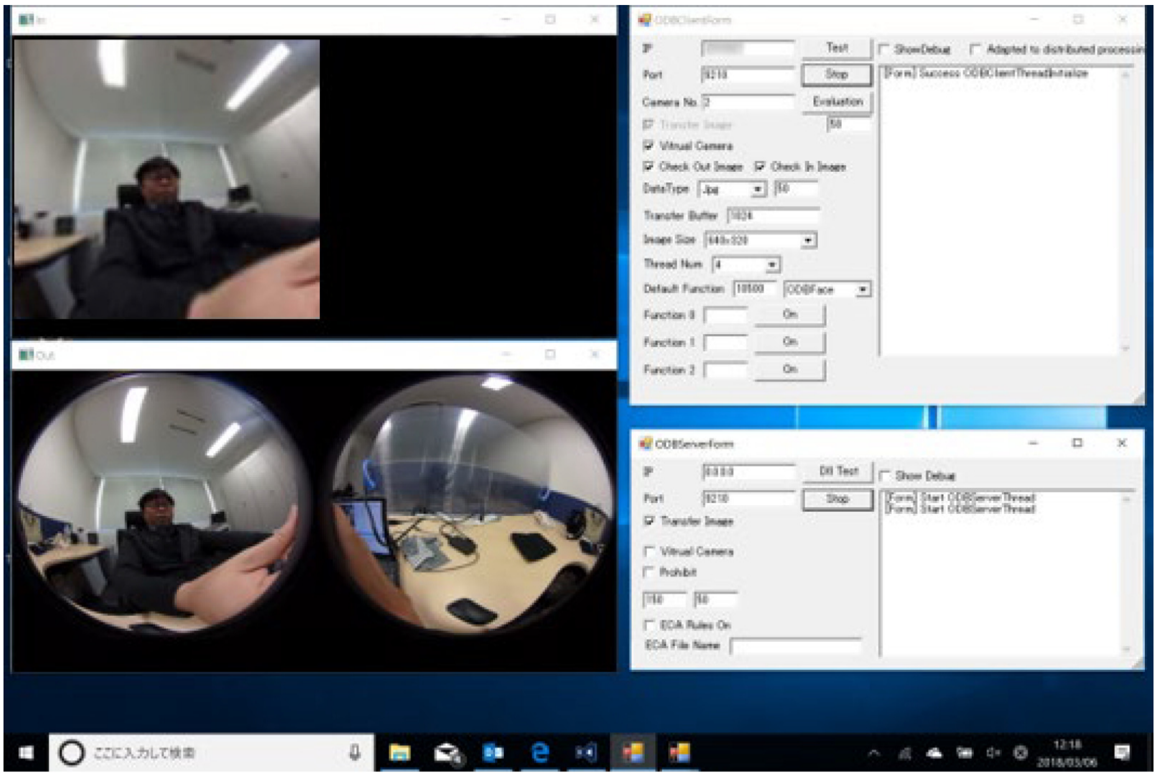

Multi-viewpoint Video recording
Different World Broadcasting Server
Distribution server
Audience · viewers
Internet
Internet
Broadcaster equipment
Video & Effect Addition processing
Data transfer
To viewers Delivery
Distributed by processing server
Distributed by distribution server
Pan Window
polar coordinates and rectangular coordinate conversion

Figure 3.3: An overview of our designed distributed multi-viewpoint Internet live broadcasting system.

and our design of ECA rules for multi-viewpoint videos. Then show some examples of ECA rules. (1)

Image Conversion of Multi-Viewpoint Videos

With omnidirectional cameras, it is not realistic to take dozens of omnidirectional images from a certain viewpoint and synthesize them on a computer to create a panoramic image. Instead, we create multi-viewpoint videos from panoramic images. The lower left part of Fig. 3.4 shows two panoramic images (front and back) for a multi-viewpoint video. These panoramic images were obtained from cameras using fisheye lenses. It is necessary to convert these images into a planar



ODBClientForm

IP [redacted]
Port 9210
Camera No. 2
☑ Transfer Image 50
☑ Virtual Camera
☑ Check Out Image ☑ Check In Image
DataType Jpg 50
Transfer Buffer 1024
Image Size 640x320
Thread Num 4
Default Function 10500 ODBFace
Function 0
Function 1
Function 2
☐ ShowDebug ☐ Adapted to distributed processin

```
[Form] Success ODBClientThreadInitialize
```

ODBServerForm

IP 0.0.0.0
Port 9210
☑ Transfer Image
☐ Virtual Camera
☐ Prohibit
150 50
☐ ECA Rules On
ECA File Name
☐ Show Debug

```
[Form] Start ODBServerThread
[Form] Start ODBServerThread
```

12:18
2018/03/06

Figure 3.4: Server software and client software.

image. There are many methods that obtain wide images from car-mounted fisheye lenses and correct the distortion [28]. Figure 3.5 shows how to obtain a wide image from a panoramic image in our proposed system. As shown in this figure, the wide images are obtained by assuming an imaginary hemispherical border for the panoramic images. The converted wide image is shown in the upper left part of Fig. 3.4. The conversion transforms virtual hemispherical polar coordinates into rectangular coordinates using the equation (4.1). In our proposed system, the distributed processing of polar/rectangular coordinates is performed using a different world broadcasting server.

$$\begin{aligned} |OP|^2 &= |QP|^2 + |QO|^2 = (\mu^2 + \nu^2) + w^2, \\ \frac{|OP'|}{|OP|} &= \frac{\sqrt{\mu^2 + \nu^2 + w^2}}{R}, \\ P(x', y', w') &= \frac{\sqrt{\mu^2 + \nu^2 + w^2}}{R}(x, y, z). \end{aligned} \tag{3.1}$$

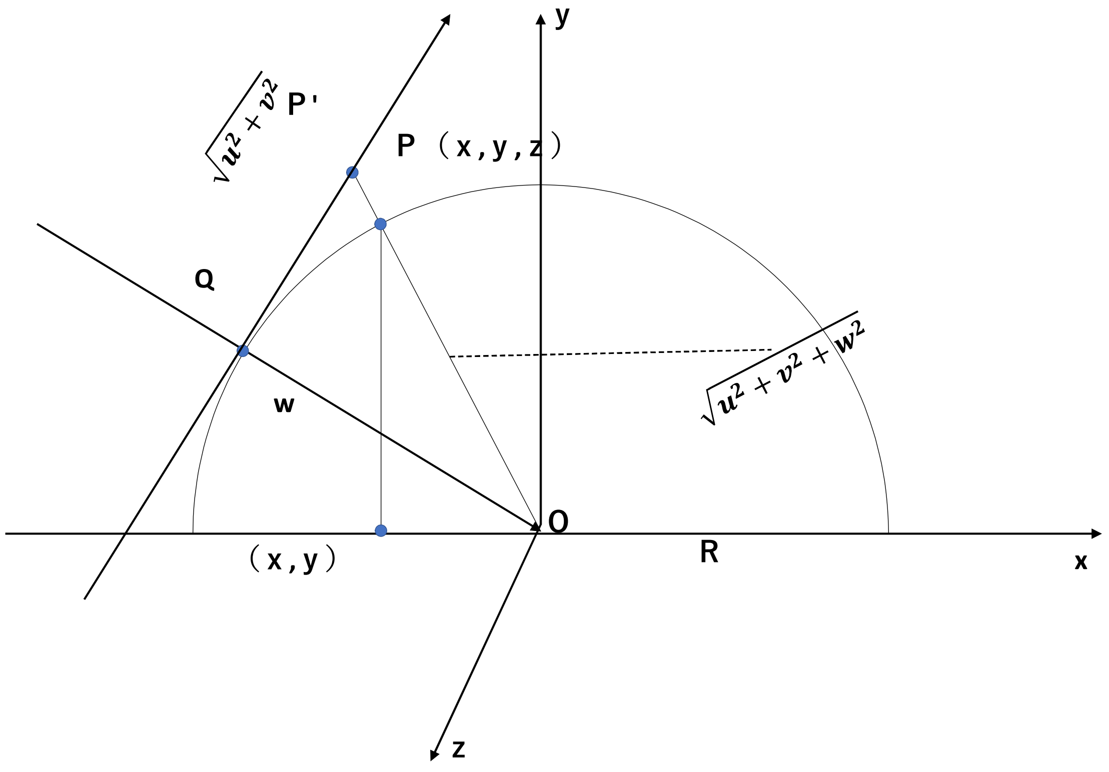


Figure 3.5: An image of the coordinate conversion.

Example of ECA Rules Set

In multi-viewpoint Internet live broadcasting services, the screen images differ according to the viewpoint selected by the user. Thus, the processes for adding effects are usually executed on the users' computers. On the other hand, general processes for Internet live broadcasting such as video encoding, video distribution are executed on the broadcaster's computer or the distribution servers. This means that processes for distributed multi-viewpoint Internet live broadcasting systems have some types. We design three types for ECA rules. One is the effect type that is related to video effects. The viewers' computers are suitable for the execution of this type because they do not need to transmit video data to others. Other one is the communication type and the rules in this type is executed on the computers performing communications. The last one is the processing type. The

DWB servers are suitable for the execution of this type because they execute these processes.

3.4 Design of ECA Rules for Multi-Viewpoint Videos

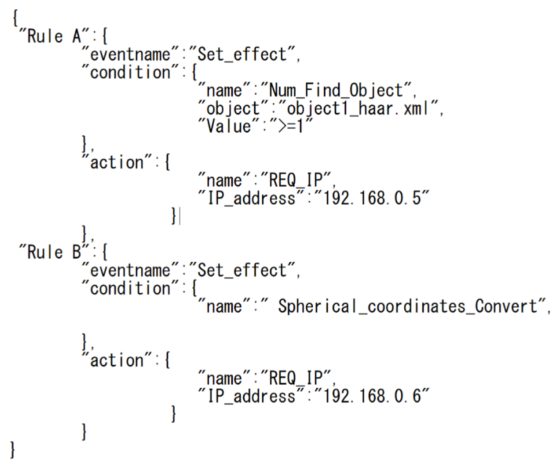
Video effects have various procedures. For example, the face detection process is generally performed before the mosaic effect is applied to the detected face. The “Timer” or “Message” functions of the ECA rules in the proposed system can define such procedures. If the procedure is defined in order-dependent ECA rules, the system needs to execute the rules according to the sequence. Otherwise, if the ECA rules do not depend on the processing request, the system can execute the rules concurrently. This reduces the processing time compared with order-dependent ECA rules. In the current system, it is impossible to process ECA rules in parallel. The parallel processing of cloud computing services is left as a future task. Lists of events, conditions, and actions are described in our previous research [17]. We list some of them in Tables 3.1–3.3,. Figure 3.6 shows an example of two ECA rules. In this example, the servers with IP addresses 192.168.0.5 and 6 are assigned as initial machines for the video processing requests from video recording terminals for the condition named “Num_Find_Object” and “Spherical_coordinates_Convert.” In cases where the processes of ECA rules have a sequence, the system should execute the processes in the order of the sequence. For example, Fig. 3.7 shows an example of the sequences of ECA rules. Some example sequences follow:

1. Is it a fisheye lens image? -¿ Perform full spherical coordinate transformation è Human detection.

2. Are humans in the image   Who?   Match with a specific person   Blur is applied.

3. Are humans in the image   Is it a known person registered in the DB?   If it is an unregistered person, blur.

The ECA rules are classified into hierarchies of detection, conversion, inquiry, and pixel processing.



```
{
 "Rule A":{
        "eventname":"Set_effect",
        "condition":{
                      "name":"Num_Find_Object",
                      "object":"object1_haar.xml",
                      "Value":">=1"
        },
        "action":{
                      "name":"REQ_IP",
                      "IP_address":"192.168.0.5"
                  }|
        },
 "Rule B":{
        "eventname":"Set_effect",
        "condition":{
                      "name":" Spherical_coordinates_Convert",

        },
        "action":{
                      "name":"REQ_IP",
                      "IP_address":"192.168.0.6"
                  }
        }
}
```

Figure 3.6: Examples of ECA rules.

Table 3.1: **Events in Communication**

Event Name	Description
Receive_Data	Occurs when data transmission finishes.
Finish_Transmission	Occurs when data transmission fihishes.
Computer_Request	Occurs when recomment server request.
Change_Server	Occurs when DWS server is busy.

3.4.1 Implementation of Proposed System

We developed a distributed live Internet broadcasting system using Microsoft Azure as a cloud service. The different world broadcasting servers run on the VMs provided by

Azure. Each VM is logically connected through a virtual network, which is one of the services provided by Microsoft Azure. Figure 3.4 shows a screenshot of the server software and client software. When starting the process of adding video effects, it provides an interface of the different world broadcasting server software. Using the client software, we can visually check the result of applying the selected effects. The client software holds the IP address of different world broadcasting servers from which video processing can be requested. If the “Apply distributed processing”

Table 3.2: **Variables for Conditions in Communication.**

Variable Name	Description
Data[]	Received data.
Transmission_Result	Result of transmission.
Turnaroundavg	Turn arount time average.
Taroundavgdiff	Turn arount time average previous differential.

Table 3.3: **Actions in Communication.**

Action Name	Description
Dispatch	Launch Dispatcher data.

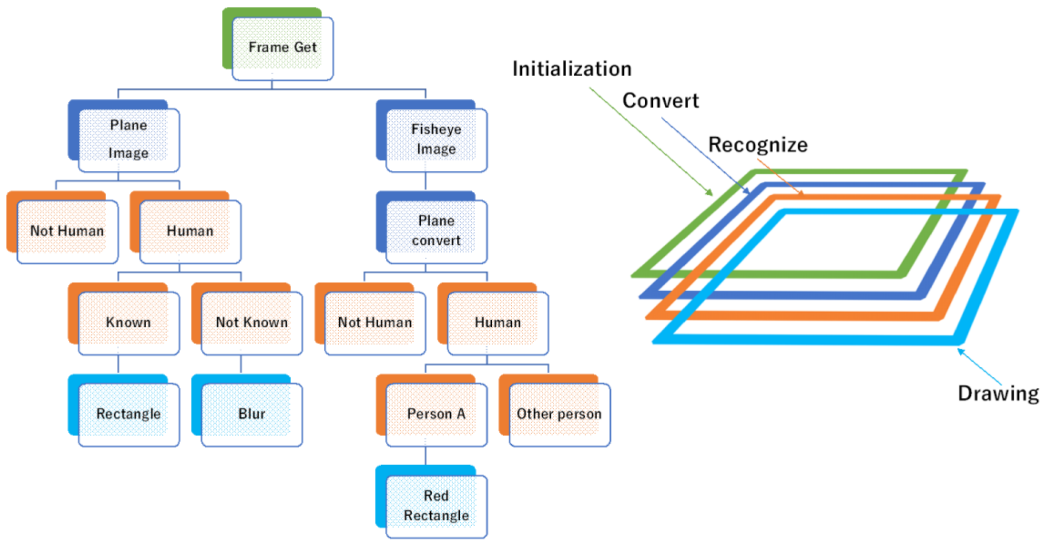


Figure 3.7: Examples of hierarchical ECA rules.

checkbox in the client software dialog box is selected, the client software requests the different world broadcasting server to execute the video processing specified by the pull-down menu of the initial IP address.

3.4.2 EXPERIMENTAL EVALUATION

We evaluated the performances of our implemented system.

Table 3.4: **Specifications of Microsoft Azure VMs.**

OS	Microsoft Windows Server 2016
Microsoft Azure Plan	Standalone Server Microsoft Corporation Virtual Machine x64-based PC.
CPU	Intel E5-2697 v3 Equivalent 2.4GHz.
Main Memory	3.584GB.

Table 3.5: **Specifications of Client PCs.**

	Client PC1	Client PC2	Client PC3
OS	Microsoft Windows10 Pro Version 1709,1511	Microsoft Windows10 Pro Version 1709,1511	Microsoft Windows10 Pro Version 1709,1511
CPU	Intel i7-7660U Equivalent 2.5GHz	Intel i5-6300U Equivalent 2.4GHz	Intel i3-4020Y Equivalent 1.5GHz
Main Memory	8.00 GB	8.00 GB	4.00 GB

Experimental System

In this evaluation, a different world broadcasting server was running on a VM provided by the Microsoft Azure service. Table 3.4 lists the specifications of the VM and OS. We used five different VMs for different world broadcasting servers. Open CV, parallelized by Intel's Parallel Computing Library TBB [29], was used as a library for executing video processing on different world broadcasting servers. The clients were PCs installed at Osaka University. Table 3.5 lists the specifications of the client PCs. We attached a full omnidirectional camera to only one PC. These PCs communicated with different world broadcasting servers via different home optical networks to avoid network congestion.

3.4.3 Evaluation Environment

We used a Theta S (RICHO Co., Ltd.) omnidirectional camera for evaluating our proposed system. Each video frame was encoded in JPEG format, transmitted, and received as a USB virtual camera. Image conversion and rule processing were realized by different world broadcasting. In the evaluation, we measured the time from the start of generating the original image data to the time that the processed image data were obtained. To confirm the efficiency of the proposed system, the video processing time, including the processing time of the ECA rule and the turnaround time, was measured as evaluation items. This includes the following four items: a) Preprocessing time during which the client receives data (same as the time from the end of reception of previous frame data to the start of the next frame data transmission). b) Communication time, while different world broadcasting servers receive frame data. c) Processing time on different world broadcasting servers. d) Communication time during which the client receives frame data from a different world broadcasting server. The video processing time is defined as the time from the start of the video processing, excluding the video data reception time, to the end of the processing. To select an available different world broadcasting server, we used the PIAX overlay network described in subsection 3.1. When a different world broadcasting server overloads, the server sends a notification to the PIAX process on the server side and waits until the load has decreased. The turnaround time of the evaluation was measured in two cases. The first case is a concentrated case in which three clients request video processing from one of three different world broadcasting servers. The second case is a completely distributed scenario in which each of the client requests is sent to a different server. As the video image processing for the evaluation experiments, face detections are executed on the processing servers after the coordinate conversions.

Figure 10: Turnaround time under one server Figure 11: Turnaround time under three servers

3.4.4 Influence of the Number of Servers

Processes were assigned among the different world broadcasting servers based on ECA rules. Figures 3.8 and 3.9 show the evaluation results of the turnaround time under the evaluation environment described in Section 4.1. The horizontal axis is the recorded frame number, and the vertical axis is the turnaround time. In Fig. 3.8, which shows the case where the load is concentrated on a single different world broadcasting server, the turnaround times gradually increase. Figure 3.9 shows the case where the image processing requests are distributed to three different world broadcasting servers. In this case, the turnaround time is less than 1500 ms, and the processing delay is around 7500 ms. In the real environment of Microsoft Azure, the different world broadcasting server from which PC 2 requests image effect processing is a VM in the East Japan region. Therefore, there

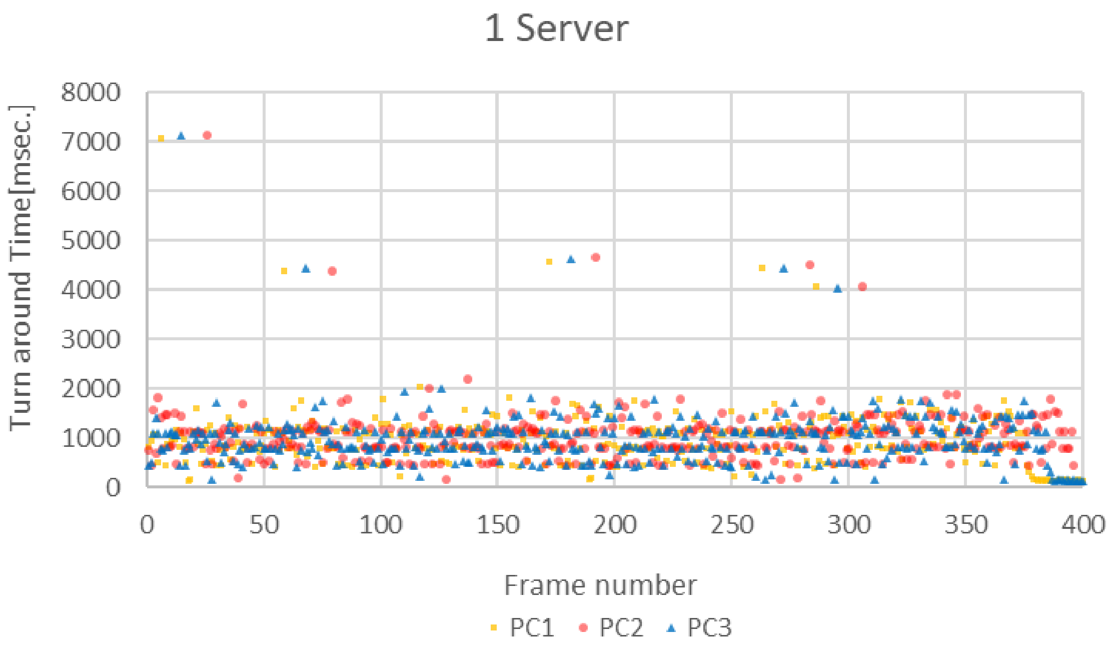


Figure 3.8: Turnaround times under one cloud server.

were variations in the communication route, and the turnaround time changes largely. We also measured the turnaround time required to change the processing server to the recommended different world broadcasting server by PIAX. The average time required to process a query for determining the recommended different world broadcasting server was 16 ms. As a result, we confirmed that the processing requests are allocated to the different world broadcasting servers based on the ECA rule, and the load is distributed. Moreover, we confirmed that the turnaround time might fluctuate, even for VMs with similar hardware performance, under the effects of communication delays.

Influence of Computational Load

We measured the turnaround times, changing the loads of DWB servers. To change the loads, we gradually increased the load caused by human face detection every one frame and measured the turnaround time. The results are shown in Figures 3.10 and 3.11. The turnaround times were measured to determine whether the load is concentrated on one virtual server or not. The turnaround times were approximately 1000 ms in this experiment. We have measured the turnaround time for

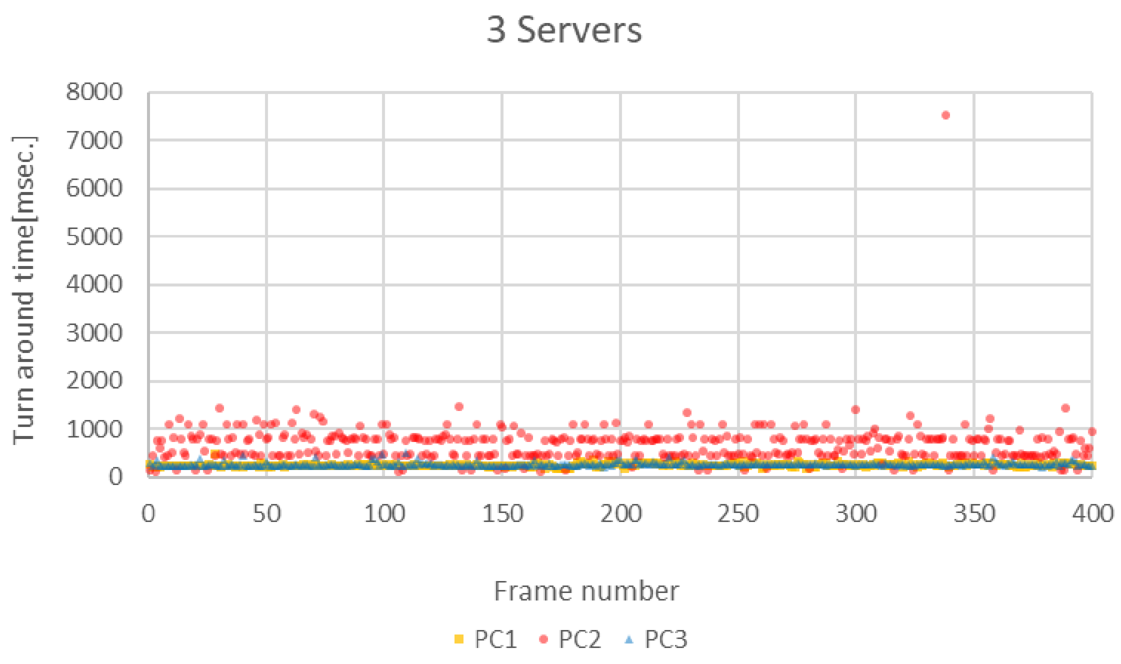


Figure 3.9: Turnaround times under three cloud servers.

single-viewpoint videos in previous research [30]. The turnaround times were approximately 16 ms. Comparing with this result, the turnaround times for multi-viewpoint videos are longer because the data amount is larger.

3.5 Discussion

3.5.1 Fluctuation of Turnaround Times

In our evaluation experiments, even when the calculation load was distributed among the three servers, video processing was sometimes concentrated on only one server. We used two networks for evaluation. (NTT's FLET'S Hikari and K-Opticom's eo light). When requests are concentrated on one different world broadcasting server, the turnaround time is relatively long. When requests from clients are concentrated on a single server, the processing load is distributed to the different world broadcasting server. Moreover, the video processing involved detecting faces in the video using the specified effect described in the ECA rule. Results using the test rules are shown in Figures

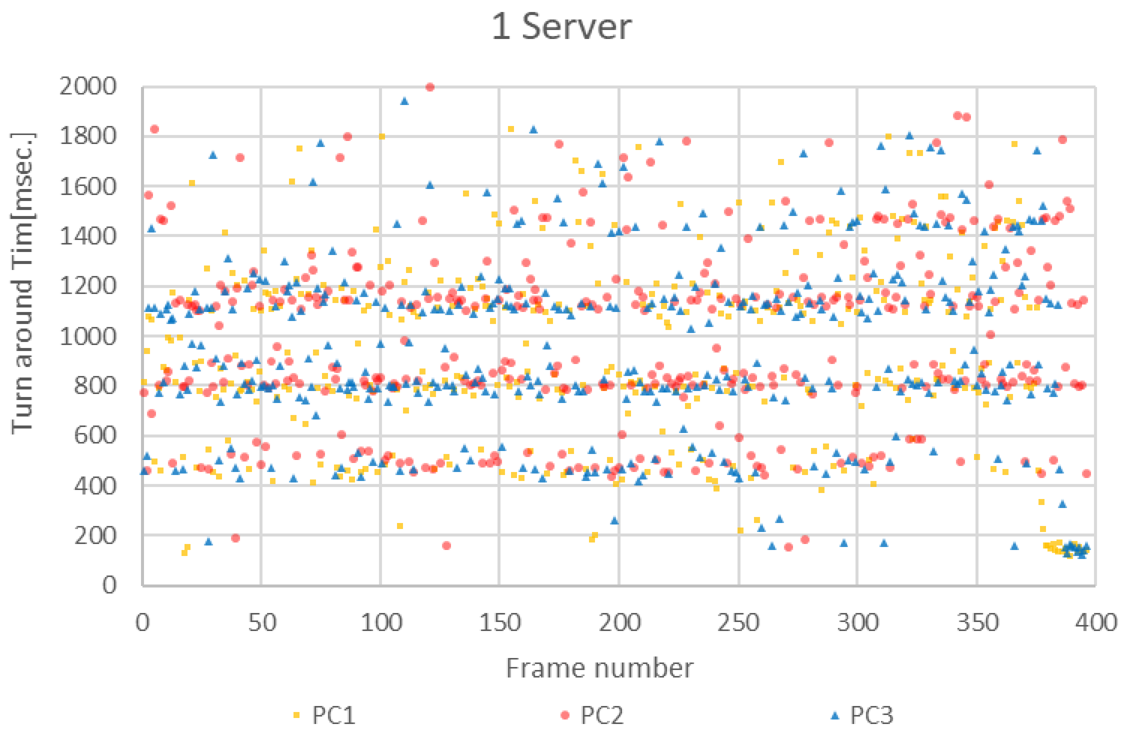


Figure 3.10: Turnaround time under one server.

3.10 and 3.11 , which confirm the fluctuations in turnaround time among cloud computing service VMs. This is caused by actual server performance fluctuations due to differences in the cloud environment of the network distance. Such issues should be considered when the user configures the system.

3.5.2 Effectiveness of ECA Rules

In previous research, we implemented a distributed Internet live broadcasting system using a cloud service and evaluated its performance. In the installed system, the processing of additional effects is performed using the VM provided by the cloud service. By determining which processing should be allocated to the VM using the ECA rule, it is possible to flexibly change the computer that executes the processing. As a result of our previous evaluations, we confirmed that the turnaround time of the effect adding process could be reduced. As an ECA rule for load balancing to be given

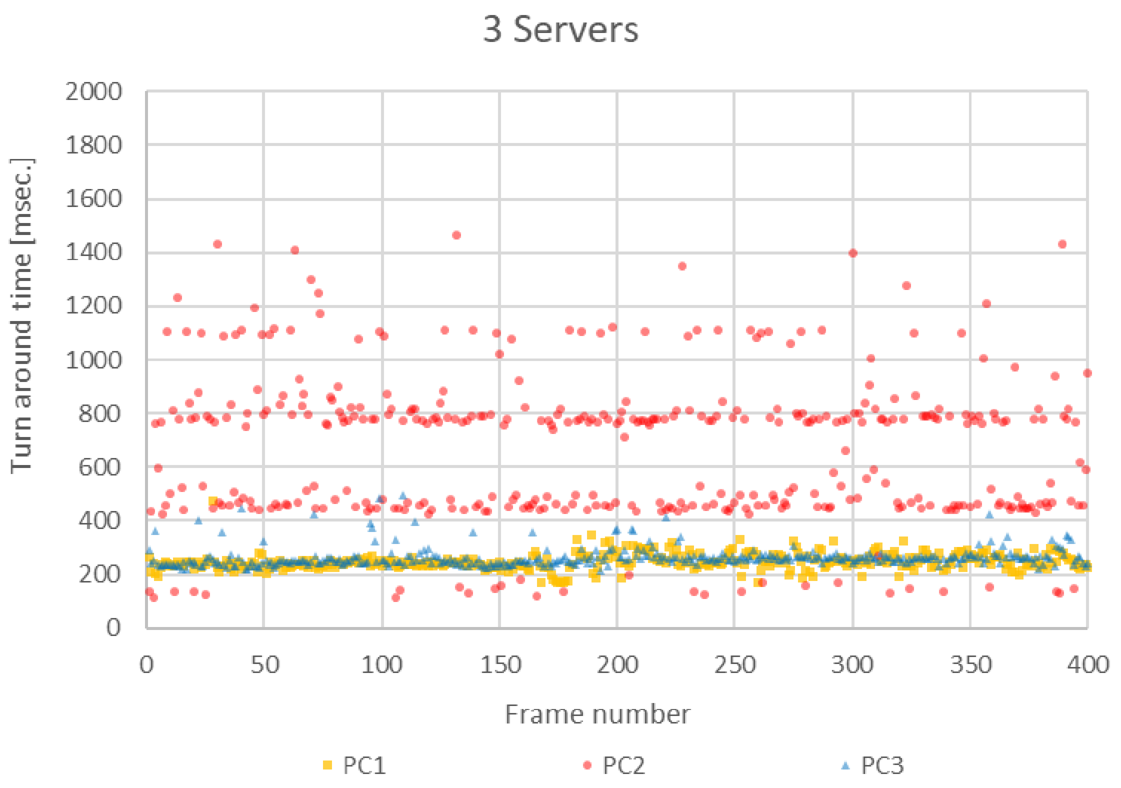


Figure 3.11: Turnaround time under one server.

to client software, the effect selection made by the client software is set as an event, and a list of corresponding enquiries is set in advance as an IP address. As a result, the server selection is performed automatically and smoothly in the process of adding special video effects.

In cases where the processes of an ECA rule have sequences, the system should execute the processes in the order of the sequence. Otherwise, the system can execute them in parallel. In this paper, we have proposed grouped three-stage rules. After the rules have been prepared, the location for their processing is selected to be either:

(1) a local client,

(2) edge computing,

or (3) cloud computing.

An image of the grouped rules is shown in Fig. 3.12 , and the example rules are shown in Fig. 3.13. In this rule system, the different world broadcasting server that adds the video effects changes

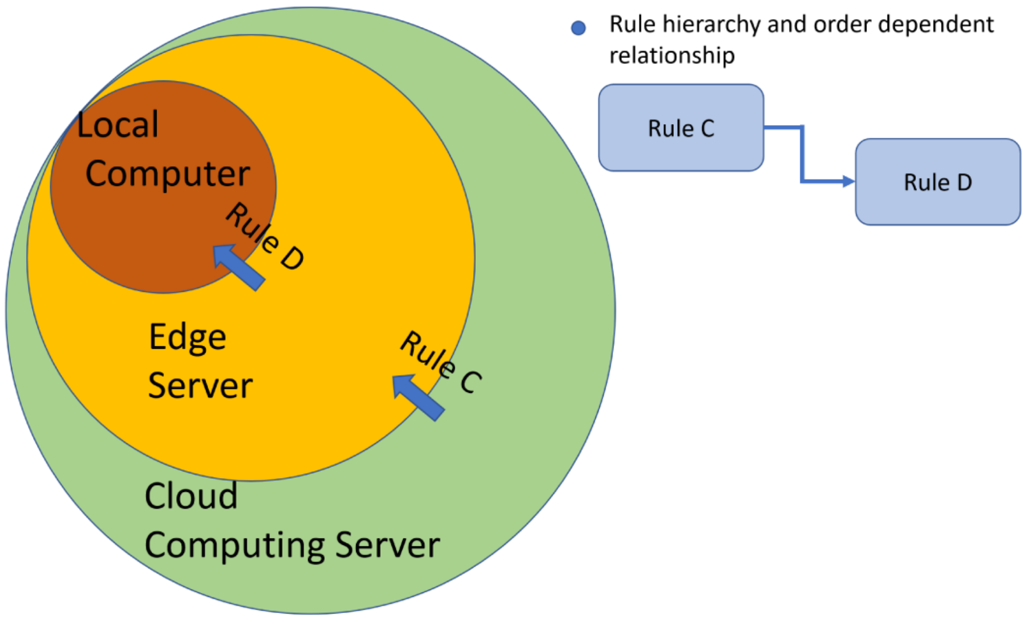
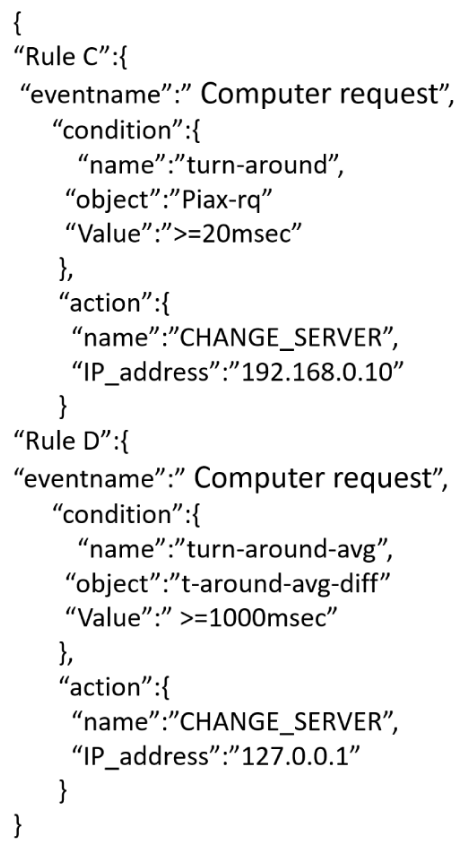


Figure 3.12: Image of the hierarchical rules.

as the performance of the current server varies.

3.6 Conclusion

In this research, we have proposed and developed a multi-viewpoint distributed live Internet (different world) broadcasting system. One of the main research challenges for multi-viewpoint Internet live broadcasting systems is how to reduce the computational load required to add effects under different screen images. Our proposed system adopts ECA rules for executing video processes. In this research, we focused on which computer executes the rules, we classified the rules into three types. Each type has a suitable computer for its execution. By classifying ECA rules to these types, our proposed system establishes appropriate execution of rules for video processes. In future work, we plan to exploit edge computing environments in which computers on the edge of the Internet can execute video processes. This could reduce the processing time because edge computers have



```
{
“Rule C”:{
 “eventname”:” Computer request”,
      “condition”:{
         “name”:”turn-around”,
       “object”:”Piax-rq”
       “Value”:”>=20msec”
      },
      “action”:{
        “name”:”CHANGE_SERVER”,
        “IP_address”:”192.168.0.10”
      }
“Rule D”:{
“eventname”:” Computer request”,
      “condition”:{
         “name”:”turn-around-avg”,
       “object”:”t-around-avg-diff”
       “Value”:” >=1000msec”
      },
      “action”:{
        “name”:”CHANGE_SERVER”,
        “IP_address”:”127.0.0.1”
      }
}
```

Figure 3.13: Example of ECA rules.

short turnaround times.

Chapter 4

Edge Computing-Based Data Caching for Novel Distribution Methods

As part of this research, we investigated a system designed to optimize video processing and distribution efficiency as a wide-area distributed system for delivering collected video content.

There are two primary challenges in video distribution. The first is addressing privacy concerns tailored to individuals. The second is resolving scalability issues based on the audience size associated with individual broadcasters. To address the latter, we proposed two distribution methods tailored to different scales: one for medium-sized audiences and the other for large-scale audiences.

This chapter focuses on the distribution method designed for medium-sized audiences. We proposed a novel approach that combines edge computing with data caching and retransmission to enhance distribution efficiency.

4.1 Introduction

Recently, video-on-demand (VoD) services such as YouTube or Netflix are widely used. In most VoD services, the clients request the video data to the video distribution server. The video distribution server sends the video data pieces sequentially to the clients so that they can play the video while receiving the pieces. When the clients cannot finish receiving each piece before starting playing it, the video playback is interrupted. A longer interruption time more annoys the viewers. Here, the interruption time means the total time that the playback is interrupted while a client is playing the video. Therefore, various interruption time reduction schemes for VoD services have been proposed. In large-scale VoD systems, many clients request the video data and thus the transmissions of data pieces frequently overlap with other transmissions. Here, the transmission means a series of distributing the pieces to a client. A transmission starts when a client requests playing a video to the video distribution server and finishes when the client finishes receiving all the pieces of the video data. If the times required for transmissions increase and the transmissions continue to overlap with others, the interruption times also continue to lengthen. Therefore, existing schemes for interruption time reduction aim to reduce the transmission time to avoid the overlapping of transmissions. Major techniques for this are pre-caching ([31–34]), redistributions of data pieces ([35–37]), etc. and are adopted in various CDN (Contents Delivery Networks) for scalable VoD systems ([38]). Unfortunately, these traditional approaches cause the communication and processing loads on the clients. Such extra loads decline the users' operability of the clients and further consume the batteries if they are mobile devices. Edge computing is one of the approaches that relief the computational loads on the clients since the edge servers, i.e., the servers on the edge of the network and geometrically close to the clients, are often managed by CDN companies such as Akamai or Cloudflare. In most of the VoD systems utilizing edge computing, the edge servers receive some pieces from the video distribution server and caches them for other transmissions. However, the number of the pieces that the edge servers need to send decreases by adopting the above both techniques (pre-caching and redistributions) to the edge servers. Here, the research challenges are: which pieces the edge servers should pre-cache and how to redistribute the pieces among the edge servers to effectively reduce the communication load and the processing load. In this paper, we propose a scalable VoD system on edge computing environments. Our proposed system adopts a distributed edge caching scheme. In our proposed system, the edge servers store some pieces before starting the VoD service. When a new client requests a video data, the video distribution server selects an edge server for sending the pieces of the requested video data to the client. The edge server sends its stored pieces to the client. After finishing sending the stored pieces, the video distribution server sends the subsequent pieces to the client. Thus, our proposed system reduces the probability that the transmissions overlap with other transmissions and increases the maximum number of the clients of that interruption time converges. The novelty of the proposed system is the adopting both the pre-caching technique and the redistribution technique to the VoD systems utilizing edge computing. The contributions of the paper are

(1) the increase of the number of the clients that the system can accommodate,

(2) the proposition of a scalable videoon-demand system,

(3) the confirmation of the effectiveness of the proposed system.

The paper is organized as follows. Some related work are introduced in Section 4.2. The proposed scheme is explained in Section 4.3, analyzed in subsection of Section4.3, and evaluated in Section 4.4. Finally, we conclude the paper in Section 4.5.

4.2 Related work

Many studies focus on fast data reception for VoD services.

Pre-caching Techniques for VoD Services Abuhadra et al. proposed a proactive caching technique for mobile devices [31]. The probability that the clients encounter interruptions decreases by sending more video data to the mobile devices while their network connections are available because they sometimes disconnect from the network. The proposed technique reduces in-network transmission delays by caching the video data. Feng et al. found the optimal cache placement for the system with wireless multicasting [32]. My research group proposed a broadcasting method for pre-cach-ing video data pieces predicting the video data that the client will play [33]. Coutinho et al. proposed a proactive caching technique for DASH video streaming [34]. In the proposed technique, the clients select a proxy caching server based on the network conditions. However, these pre-caching techniques for VoD services require the clients' storage capacity.

Redistribution (Peer-to-Peer Sharing) Techniques for Video-on-Demand Services Sheshjavani et al. proposed a peer-to-peer data sharing mechanism for VoD services [35]. In the proposed mechanism, the clients manage their buffer map. The buffer maps inscribe the received video data pieces and non-received pieces of each client. In the mechanism, the clients exchange the pieces based on the buffer map to receive the pieces that each client does not have. In the method proposed by Zhang et al., the clients send the pieces considering the bandwidth consumption [36]. Fratini et al. analyzed the efficiency of using replicated video servers [37]. However, similar to the pre-caching techniques, these redistribution techniques cause communication and processing loads on the clients. Such extra loads decline the users' operability of the clients and consume the batteries if they are mobile devices.

Video-on-Demand Services on Edge Computing Due to the recent prevalence of edge computing, some researchers focus on edge computing for VoD services. Mehrabi et al. proposed an edge computing assisted adaptive mobile video streaming [38]. The mechanism determines the video resolution and the video data rate based on the network conditions between the clients and the edge servers. The performance of an edge computing enhanced video streaming is investigated by Yang et al. [39].

4.3 Proposed System

In this section, we explain our proposed system.

4.3.1 Proposed System Architecture

Figure 4.1 shows the assumed system. The system consists of three layers, the top layer, the edge layer, and the client layer. One video distribution server is in the top layer. CDN (Con-tents Delivery Network) companies or VOD service companies provide the machines in the edge layer. The edge layer includes some edge servers. The clients are in the client layer. Similar to other researches for the VoD systems utilizing edge computing, the networks for these three layers are application layer networks and we assume that the influences of the underlaying session/transport layers are sufficiently small. The single video distribution server has full video data for all the videos and connects to the Internet. The edge servers also connect to the Internet and can communicate with

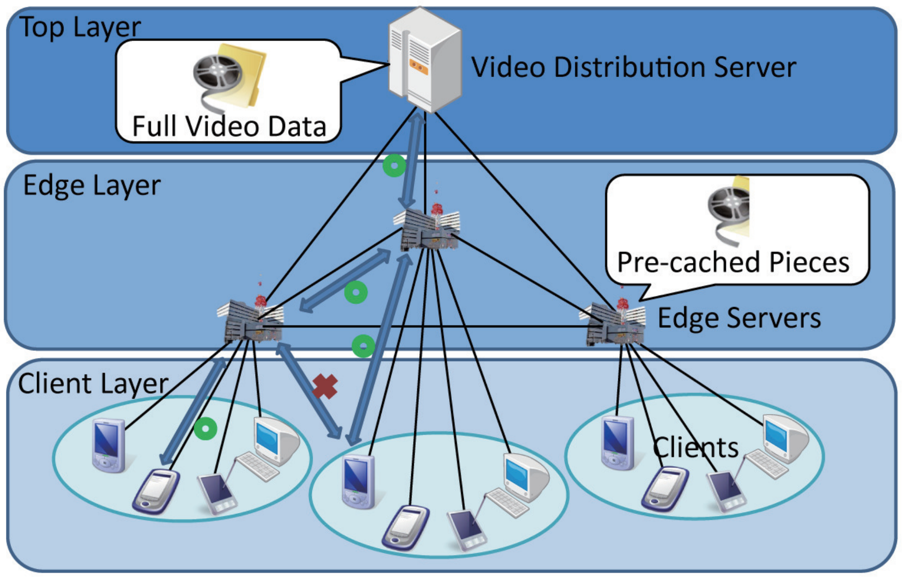


Figure 4.1: Assumed Environment.

the video distribution server. They can store a part of some video data (pieces). The clients connect to the geometrically closest edge server and can communicate with the edge server. The clients request the video data to the video distribution server. Each video data are divided into some pieces, as shown in Figure 4.2. The clients receive the requested video data pieces from the video distribution server and the edge servers. The assumed system model is general and practical. One of the applications is that the video distribution server provides the videos to the clients in a prefecture, and each edge server serves for each region in the prefecture. For example, there are eight local regions in Osaka, Japan. In this case, the number of the edge servers is eight, and the edge servers provides 100 videos. We show the assumed DDC in this section.

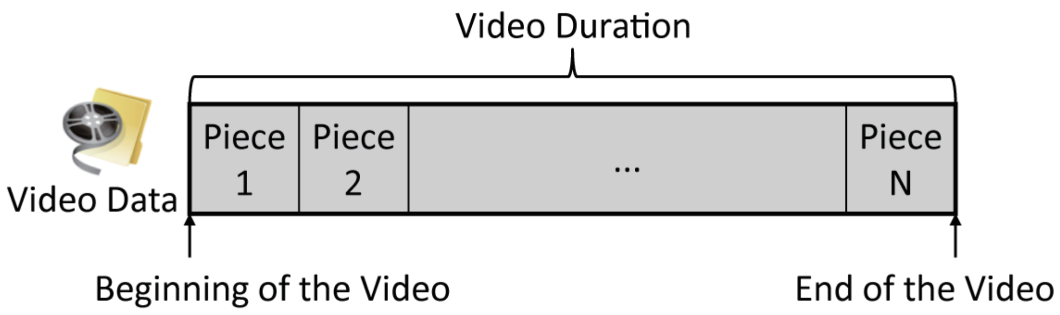


Figure 4.2: Video Data Division.

4.3.2 Target Issue

We explain our target issue in this subsection.

Issues in Conventional Systems

In the VoD systems, the video playback is interrupted when the clients cannot finish receiving each piece before starting playing it. A longer interruption time more annoys the viewers. Here, the interruption time means the total time that the playback is interrupted while a client is playing the video. In the VoD system in that a video distribution server distributes the video data to the clients, the communication load and the processing load for the distribution concentrate on the server. Therefore, in the cases that such VoD system hold many clients (scalable) and the clients frequently requests to play the videos to the video distribution server, the server is easy to overload. The overloading results in long video data transmission times and causes long interruption times. In most of the VoD systems utilizing edge computing, the edge servers receive pieces from the video distribution server and caches them for other transmissions.

Pre-Caching Pieces

The edge servers can receive pieces from the video distribution server before the requests for them come from the clients. That is, the edge servers can pre-cache the pieces. The pre-caching can reduce the communication load and the related processing load on the video distribution server and the edge servers. This is because they receive the pieces without the requests for the pieces and can receive them before starting the VoD service. Pre-caching more pieces reduce more loads, but require more storage capacity on the edge servers.

Redistributing Pieces

Moreover, the edge servers can redistribute pieces to other edge servers. The redistribution can distribute the communication load and the related processing load on the video distribution server to the edge servers because the edge servers send the pieces instead of the server. However, if an edge server frequently redistributes the pieces, the loads concentrate on the edge server, results in long interruption time. Therefore, the edge servers need to redistribute the pieces without causing the overloads on it.

Our Objective

Based on the discussion in this subsection, we aim to reduce the interruption time by adopting pre-caching and redistributing pieces on edge computing environments. For this, we propose a scheme which determines which pieces the edge servers should pre-cache and how to redistribute the pieces among the edge servers.

Proposed Scheme

Our proposed scheme runs on our proposed system architecture explained in Subsection 3.1. In the scheme, the edge servers pre-cache several preceding pieces of popular videos. When a client requests playing a video to the connected edge server, the edge server checks whether it has pre-cashed the requested video already. If the piece that the client is going to receive is pre-cached, the client receives the piece from the connected edge server. If the piece is pre-cached by another edge server, the connected edge server receives it from the edge server and after that sends it to the client. If the pieces is not pre-cached by any edge server, the connected edge server receives it from the video distribution server and after that sends it to the client.

How to Pre-Cache Pieces

To solve the first issue, the edge servers pre-cache preceding several pieces of popular videos, i.e., the pieces that are close to the beginning of the videos. This is because the time to start playing the preceding pieces is early, and the possibility that the clients encounter interruptions is high. The preceding pieces of the videos that are frequently requested by the connected clients are pre-cached. To avoid redundant precaching, the edge servers do not pre-cache the pieces that are pre-cached by other edge servers. The edge servers do not cache the pieces that are transmitted to the clients except for the pre-cached pieces to reduce the required storage capacity for caching. The number of the pieces to be pre-cached is a parameter for the scheme. For example, imagine the case that the VoD system provides 100 videos and owns eight edge servers. When the number of the pieces to be pre-cached is set by 10 per each video, each edge server pre-caches the preceding 10 pieces of the 100/8=25 videos.

How to Redistribute Pieces

Memory resources are responsible for storing the data required by computational resources. It encompasses memory and storage components. For the purposes of this chapter, memory space is divided into blocks of a certain size, with each block representing a single memory resource.

4.3.3 Flow of Procedures

In this subsection, we explain the flow of the procedures for the video distribution server, the edge servers, and the clients.

Video Distribution Server

The video distribution server has all the pieces of all the video data. When it receives a request for a piece, it sends the requested piece to the requesting edge server. Moreover, the video distribution server manages the statistics of the VoD system and measure the popularity of the video data to determine which video data each edge server should pre-cache. This can be performed by calculating popularity of the video data. The video distribution server can calculate this by getting the information about the video requests from all the edge servers.

Edge Servers

Figure 4.3 shows the procedure flow of the edge servers. In the figure, pj,i indicates the ith piece of the video j. When an edge server receives the request for the video from a client, it checks whether it pre-caches the first piece of the video or not. If the edge server pre-caches the piece, it sends the piece to the client. Otherwise, it finds the edge server that pre-caches the piece. If there is the edge server that pre-caches the piece, it requests the piece reception to the edge server and waits for the reception. When the reception completes, it sends the received piece to the client. If no edge servers pre-cache the piece, it request the piece reception to the video distribution server and sends it to the client when the piece reception completes. The edge server that receives the request of the video continue to this procedure until it sends the last piece. When it completes sending all the pieces of the video to the client, the request flow is finished.

4.3.4 Clients

Figure 4.4 shows the procedure flow of the clients. Similar to Fig.4.3, pj,i indicates the ith piece of the video j. When a client receives the first piece, it starts playing the piece. After playing the piece, the client try to continuously play the next piece and checks whether it has stored the next piece or not. If the next piece has already stored in its storage, it starts playing the next piece. Otherwise, it waits for the reception of the next piece. In this case, interruption occurs. The client continue to this procedure until finishing playing all the pieces. In this section, to find the necessary bandwidth among the video distribution server, the edge servers, and the clients, we analyze the communication situation under the proposed scheme. In our proposed scheme, each video data is divided into some pieces. The data amount for each piece is the same and is denoted by P. For example, if the number of the pieces of the video i is NPi, the data size of the video is P × NPi.

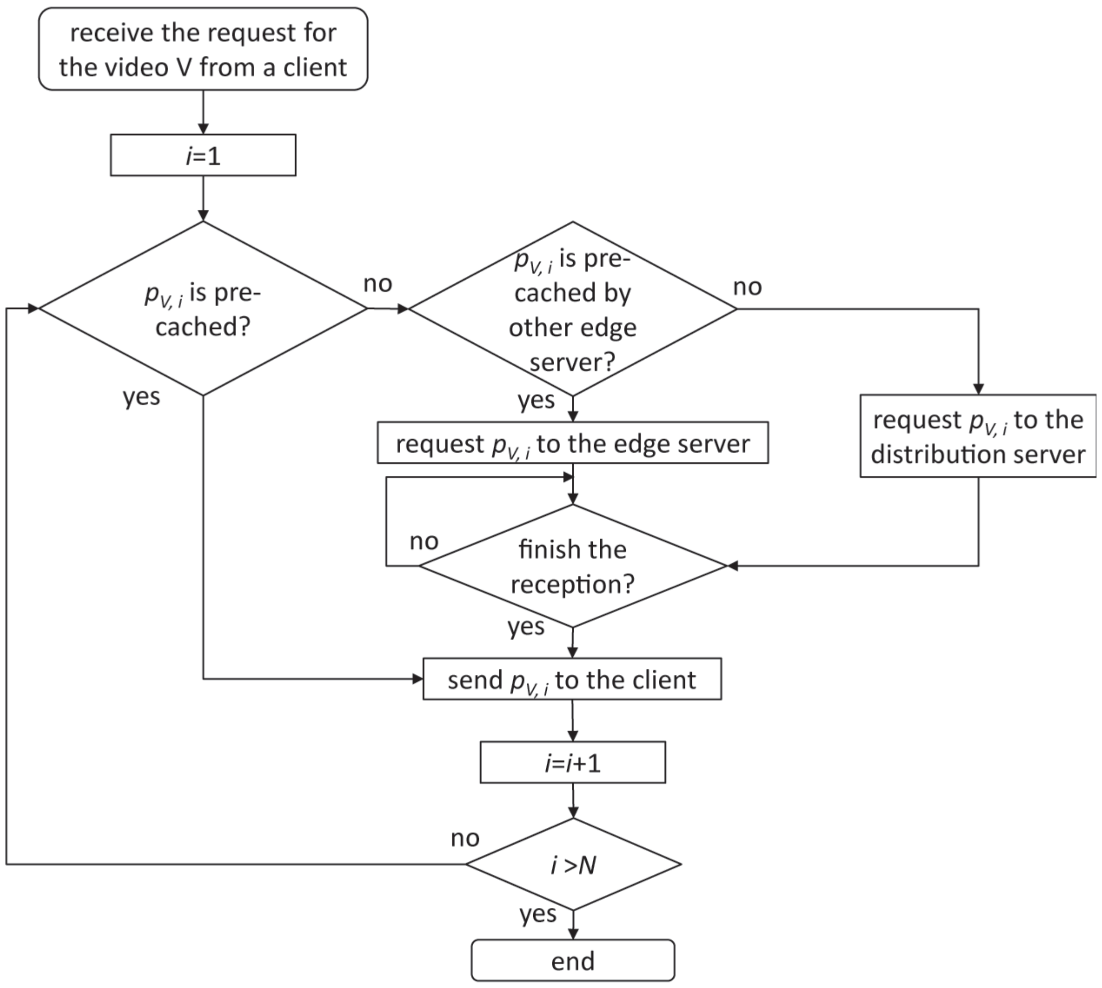

receive the request for the video V from a client
i=1
$p_{V,i}$ is pre-cached?
no
$p_{V,i}$ is pre-cached by other edge server?
no
yes
yes
request $p_{V,i}$ to the edge server
request $p_{V,i}$ to the distribution server
no
finish the reception?
yes
send $p_{V,i}$ to the client
i=i+1
no
i >N
yes
end

Figure 4.3: The procedure flow of the edge servers.

Necessary Bandwidth Between Clients and Edge Servers

Suppose the case when an edge server receives the new video request before finishing sending all the pieces to the current client. In this case, the interruption time diverges since the piece transmissions overlap. The average arrival interval of the clients for an edge server is given by ooo.ooo is the average global arrival interval of the request for the videos from the clients, and E is the number of the edge servers. Therefore,

$$\frac{PM}{B_{EC}} < \lambda_E \tag{4.1}$$

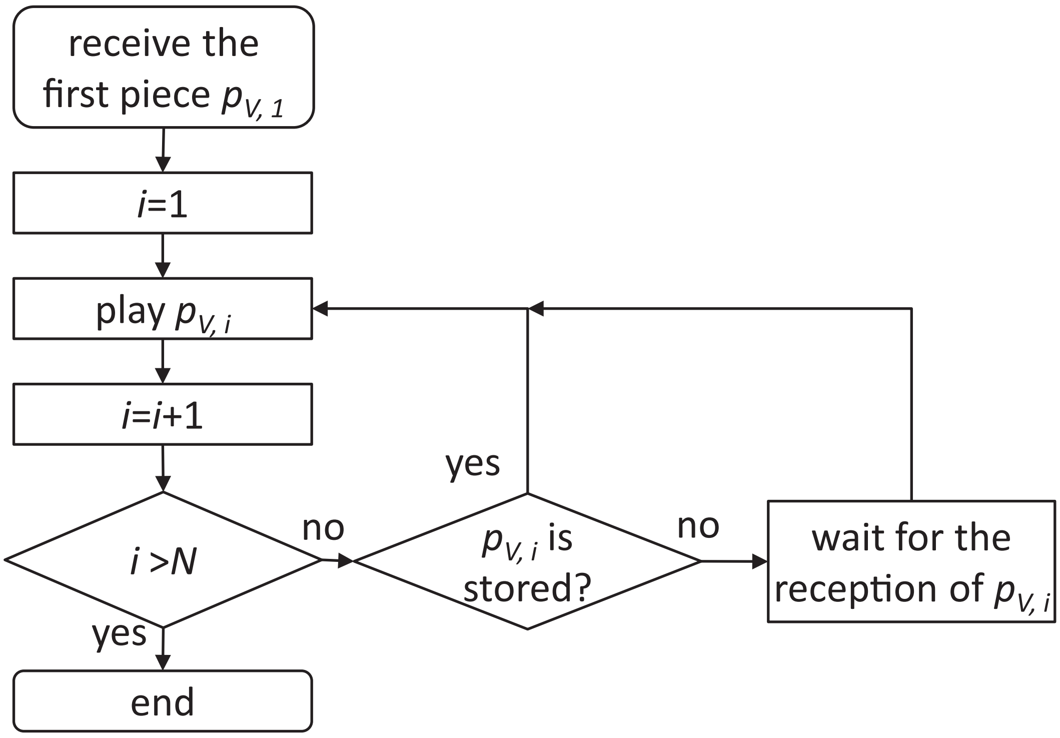

receive the first piece $p_{V,1}$
i=1
play $p_{V,i}$
i=i+1
$i > N$
yes
end
no
$p_{V,i}$ is stored?
yes
no
wait for the reception of $p_{V,i}$

Figure 4.4: The procedure flow of the clients.

Here, M is the maximum number of the pieces of all the videos. Bec is the bandwidth between the edge server and the clients. For the analysis, I assume that the bandwidth is the same for all the edge servers. From (3.1), the following inequality should be satisfied to converge the interruption time.

$$B_{EC} > \frac{PM}{\lambda_E} \tag{4.2}$$

Necessary Bandwidth Among Edge Servers

We assume that the video data are requested with a same probability. Each edge server has the same number of the videos. Let V denote the number of the videos that the system provides. Then, the probability that a video is requested when a client requests playing a video is 1/V. Therefore, the probability that the video data that an edge server has are requested is (V / E) / V=1/E. Contrary, the probability that the directly connected edge server does not have the requested video data is (E - 1)/E. Therefore, a redistribution occurs with the probability (E - 1)/E. The time that the edge server can consume for the redistribution is ooo if the edge server does not have the next requested video. The probability that the edge server does not have the next requested video is (E - 1)/E. If the edge server has the next requested video (the probability is 1/E), the time that the edge server can consume for the redistribution is 2ooo. Thus, the average time that the edge server can consume for the redistributions is given by

$$\sum_{i=1}^{\infty} i\lambda_E \left(\frac{E-1}{E}\right)^2 \left(\frac{1}{E}\right)^{i-1} \tag{4.3}$$

Let J the number of the pieces allocated to each edge server. The time needed to send J pieces among the edge servers should be shorter than this average arrival interval to avoid transmission overlapping. Therefore,

$$\frac{PJ}{B_{EE}} < \lambda(E-1)^2 \sum_{i=1}^{\infty} i \left(\frac{1}{E}\right)^i \tag{4.4}$$

Hence, the bandwidth among the edge servers BEE should satisfy the following inequality.

$$B_{EE} > \frac{PJ}{\lambda(E-1)^2 \sum_{i=1}^{\infty} i \left(\frac{1}{E}\right)^i} \tag{4.5}$$

Necessary Bandwidth for Video Video Distribution Servers

The edge servers receive the pieces that are not allocated to any other edge servers from the video distribution server. The time needed to send M – J pieces from the video distribution server to the edge server should be shorter than the average global arrival interval. Therefore,

$$\frac{P(M-J)}{B_{DE}} < \lambda \tag{4.6}$$

BDE is the bandwidth between the video distribution server and the edge servers. Hence, the following inequality should be satisfied to converge the interruption time.

$$B_{DE} > \frac{P(M-J)}{\lambda} \tag{4.7}$$

4.4 Experimental Evaluation

To check the performances of the proposed scheme, we measured the interruption time using our developed simulator.

4.4.1 Evaluation Setting

Based on the application example in Subsection 3.1, the number of the edge servers is eight, and the edge servers provide 100 videos. The bandwidth between each edge server and the clients is 100 [Mbps] considering a realistic situation. The bandwidth between the video distribution server and the edge servers is 600 [Mbps], and that among the edge servers is also 600 [Mbps], considering that these are in the backbone network. I set the same bandwidth to all the edge servers to make the experiments precisely understandable. The video distribution server can communicate with each edge server directly, and the edge servers can communicate with each other. The clients connect to the closest edge server. The video duration is 60 [min.], and the bitrate is 5 [Mbps] based on the videos provided by practical services. The data amount of a piece is the same as the video data for 5 [sec.] based on HLS (HTTP Live Streaming [39]) and is 3125 [Kbytes]. The number of the pieces in each video data is 720. The users request playing one of the videos according to a fixed arrival interval to make the results be easily understandable. In the case that the requests non-uniformly arrive, the average interruption time increases because the maximum value increases. The popularity of the video data is given fairly. We compare the proposed scheme with a conventional edge caching scheme, an often used caching technique for CDN. In the scheme, the pieces are cached at the edge servers, but not redistributed among them. The edge servers receive the pieces that need to be sent to the clients and are not cached from the video distribution server.

4.4.2 Influence of Arrival Interval

More frequent arrivals of the requests for playing the video data cause more transmission overlaps, and thus, the interruption time can continue to increase with a higher probability. Therefore, we investigate the influence of the arrival intervals of the clients' requests.

Interruption Time

Figure 4.5 shows the average interruption time under different arrival intervals. The horizontal axis is the arrival interval and the vertical axis is the average interruption time. In the legend, ‘Conventional Edge Caching’ indicates the average interruption time under the conventional edge cashing scheme explained in the previous subsection. ‘Proposed (J pieces)’ indicated the average interruption time under our proposed scheme. In the scheme, each edge server pre-caches J pieces. We can see that the average interruption times under each scheme are almost the same when the arrival interval is longer than a certain value. This is because the transmissions does not overlap with others and the interruption time does not continue to increase. Under the conventional scheme, the average interruption time when the arrival interval is longer than 30 [s] is a little bit longer than that under our proposed scheme. This is because the bandwidth between the edge servers and the clients are not sufficient and the transmissions sometimes overlap with others. On the other hand, we can see that the average interruption times under all schemes suddenly increases when the arrival interval is shorter than a certain value. This is because the transmissions always overlap with the next transmission and the interruption time continue to lengthen. In such cases, the VoD system abandons and cannot provide their services. For example, when the arrival interval is 25 [s], the conventional scheme cannot provide the service, but our proposed scheme can provide it and the average interruption time is approximately 325 [ms]. At the shortest, our proposed system can work even when the arrival interval is 22 [s]. The conventional method can provide service only when the arrival interval is longer than 30 [s] in this situation. Therefore, our proposed system can improve the shortest arrival interval under that the system can work 26% compared with the conventional system.

Edge Servers’ Data Transmission

One of the indexes for the communication load and the processing load on the edge servers is the data amount transmitted to others. Therefore, we investigate the amount changing the arrival interval. Figure 6 shows the total data amount transmitted to the clients by the edge servers. Since the average interruption time diverges when the arrival interval is excessively short, the lines stop at

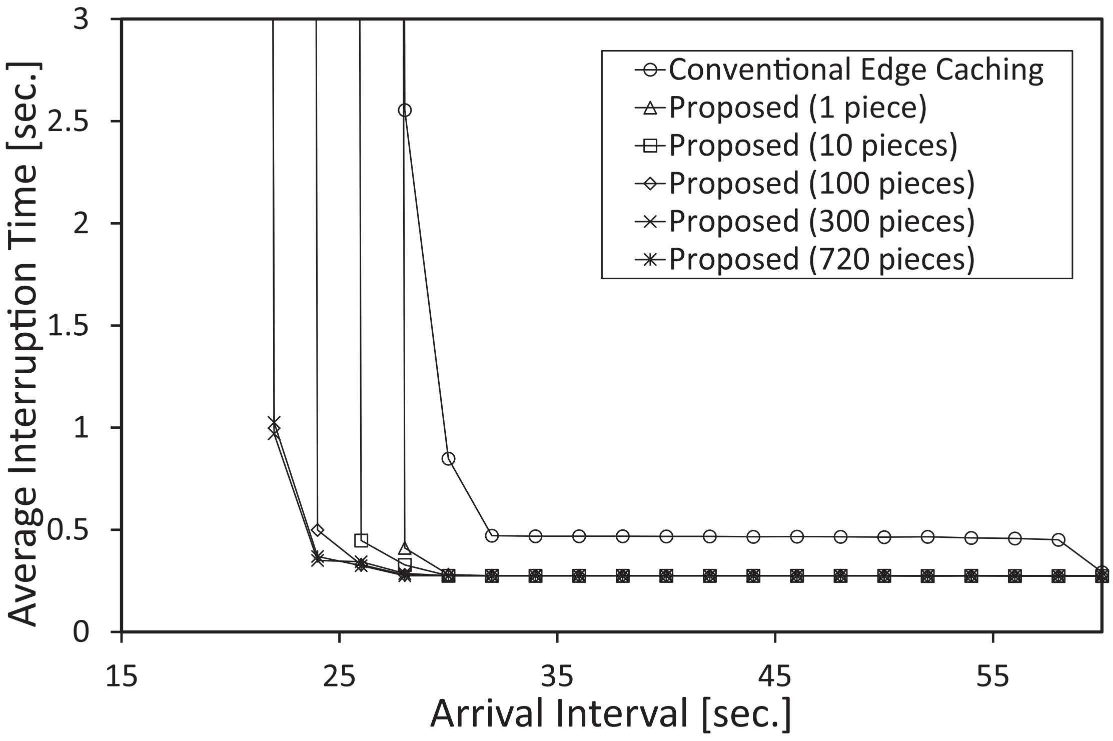


Figure 4.5: Average arrival interval and the interruption time.

the shortest arrival interval under that the interruption time converges. The data amount decreases as the arrival interval increases because the number of the clients that receive data per time decreases. The data amount increases as the edge servers pre-cache more data because they need to transmit them instead of the video distribution server. Figure 7 shows the total data amount transmitted to other edge servers. The result is similar to Fig. 6, the total data amount transmitted to the clients. But, the amount differs. The data amount transmitted to other edge servers is generally larger than that to the clients because the edge servers request the data transmissions to other edge servers in proportional to the number of the clients and each edge server respond to the requests from all the other edge servers. We only show the total data amount transmitted to the clients because we confirmed that the total data amount transmitted to the edge servers is similar to this.

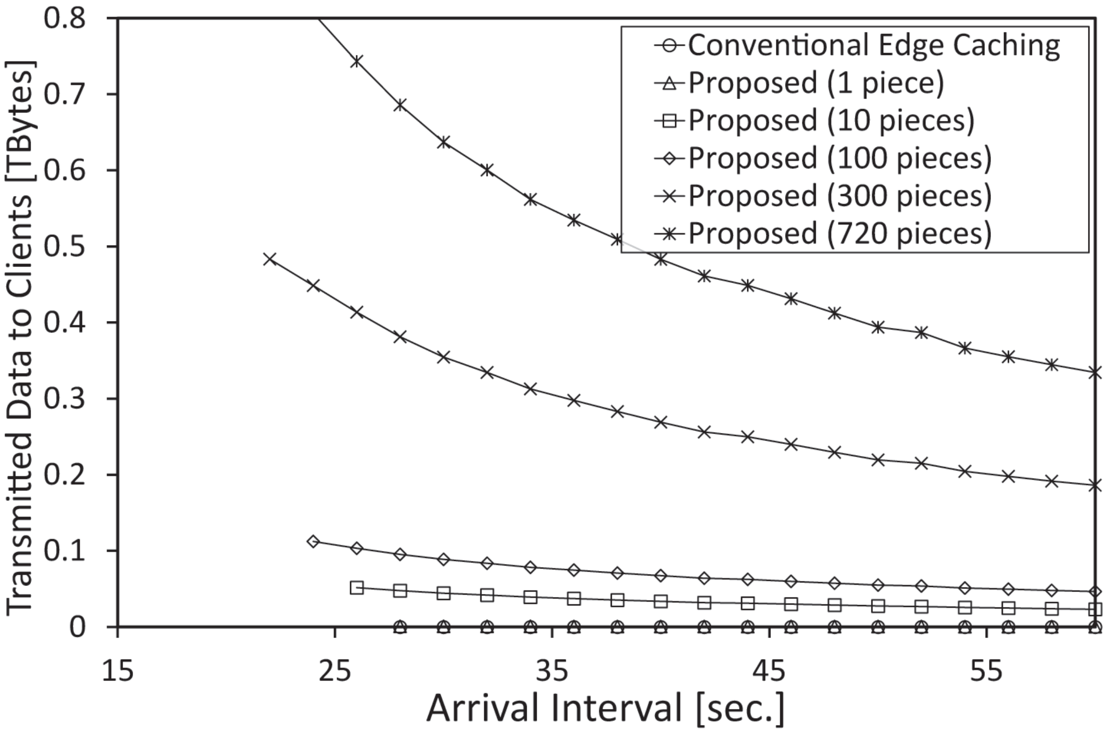


Figure 4.6: Average arrival interval and the transmitted data to the clients.

4.4.3 Influence of Edge-Client Bandwidth

A less communication bandwidth between the edge servers and the clients (edge-client bandwidth) cause more transmission overlaps, and thus, the interruption time can continue to increase with a higher probability. Therefore, we investigate the influence of the bandwidth between the edge servers and the clients. Figure 4.8 shows the average interruption time under different client-edge bandwidth. The arrival interval is 30 [s]. The horizontal axis is the client-edge bandwidth, i.e., the bandwidth between each edge server and the clients. The vertical axis is the average interruption time. We can see that the average interruption times under our proposed scheme are almost the same when the edge-client bandwidth is larger than a certain value. Similar to the discussion for the previous subsection, this is because the transmissions does not overlap with others when the edge-client bandwidth is large. For the same reason as the previous subsection, the average interruption time under the conventional scheme is a little bit longer than that under our proposed scheme. Since

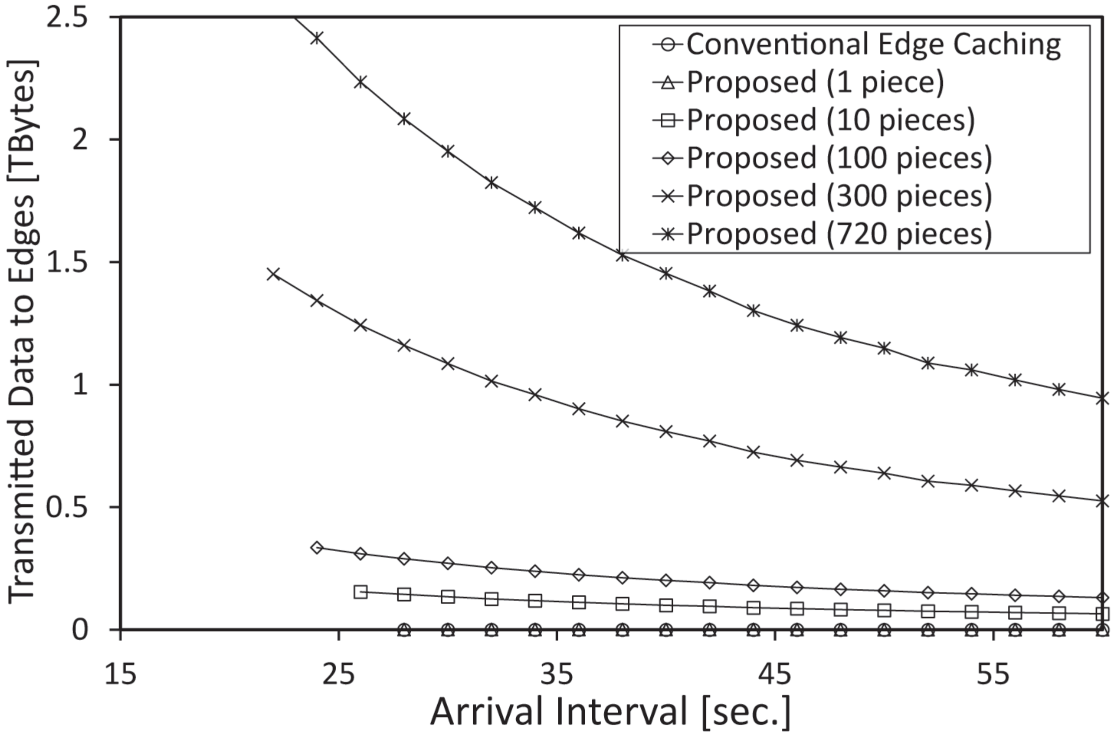


Figure 4.7: Average arrival interval and the transmitted data to the edge servers.

the behavior between the edge servers and the clients are the same between our proposed scheme and the conventional scheme, the edgeclient bandwidth under that the interruption time diverges is the same, approximately 75 [Mbps]. Figure 4.9 shows the total data amount transmitted to the clients. Since the average interruption time diverges when the edge-client bandwidth excessively small, the lines stop at the smallest bandwidth under that the interruption time converges. The total data amount does not change even when the edgeclient bandwidth changes because the data amount only depends on the number of the clients and the number of the precached pieces.

4.4.4 Influence of Edge-Edge Bandwidth

A less communication bandwidth among the edge servers (edge-edge bandwidth) cause more transmission overlaps. Thus, the interruption time can continue to increase with a higher probability. Figure 4.10 shows the average interruption time. The arrival interval is 30 [s]. The horizontal

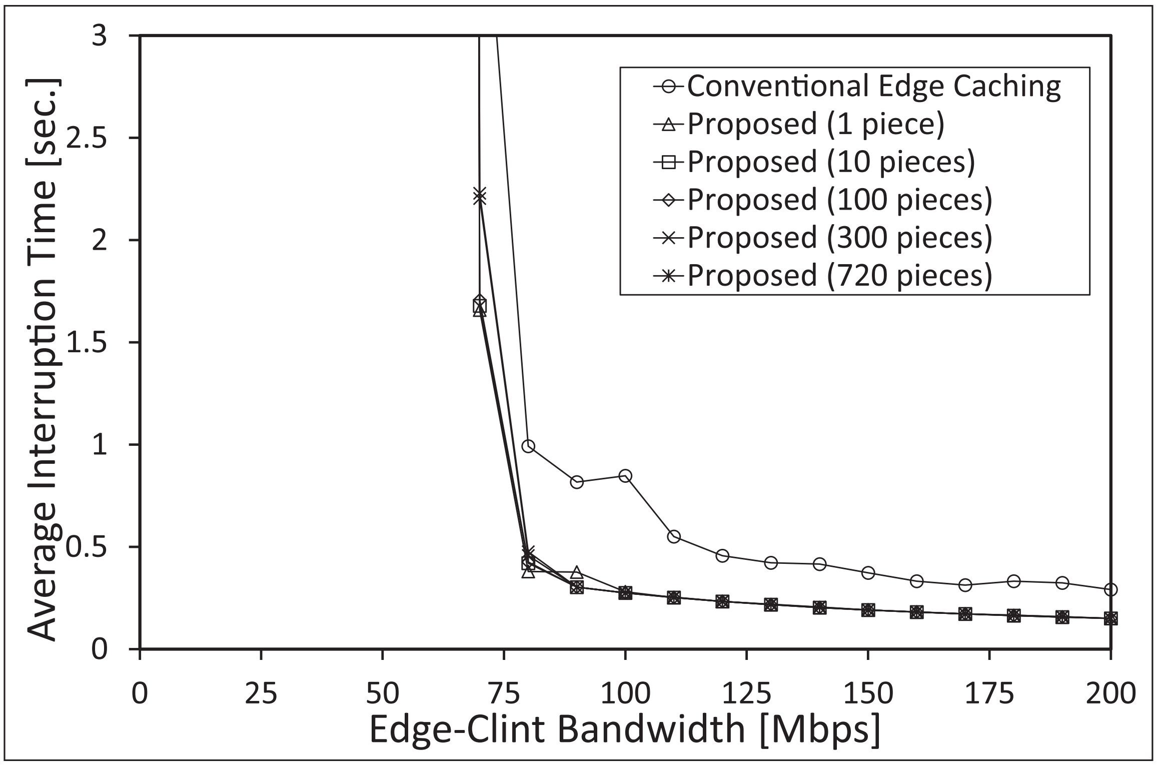


Figure 8

Figure 4.8: Edge-client bandwidth and the interruption time.

axis is the edge-edge bandwidth. The average interruption time under the conventional scheme is constant even when the edge-edge bandwidth changes because the edge servers do not redistribute the pieces in the scheme. The average interruption time under our proposed scheme decreases as the edge-edge bandwidth increases because the edge servers can faster redistribute the pieces to another edge server. Moreover, the average interruption time continue to lengthen when the edge-edge bandwidth is smaller than a certain value because the transmissions overlap. The total data amount transmitted to the clients and the edge servers do not depend on the edge-edge bandwidth and these have a similar tendency as shown in Figs. 6 and 7. Therefore, we show the total data amount transmitted to the clients for the situation of Fig. 4.10 in Table 4.1. As shown in the table, the data amount increases as the number of the pre-cached pieces increases because the edge servers own more data. At the maximum case, i.e., the edge servers pre-cache all the 720 pieces, the data

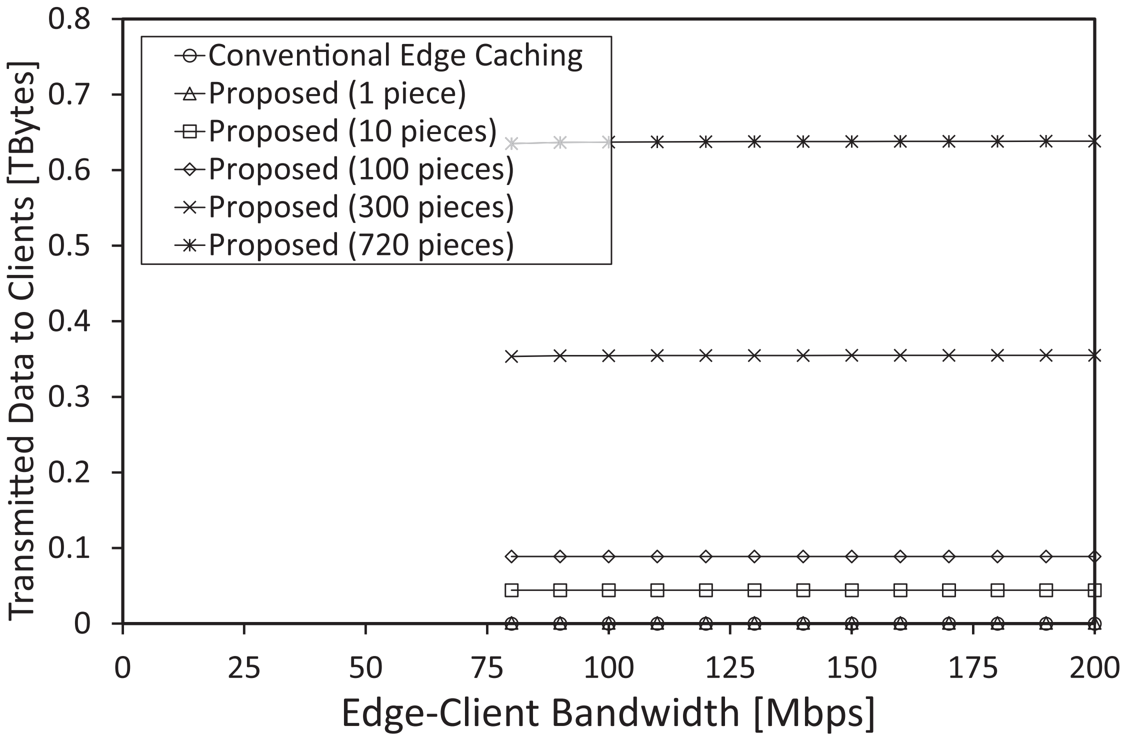


Figure 4.9: Edge-client bandwidth and the transmitted data to the clients.

Table 4.1: **Transmitted data to the clients under Figure 10.**

Pre-cached pieces	0	1	10	100	300	720
Transmitted Data Amount	0 [Gbytes]	888 [Gbytes]	44.4 [Gbytes]	88.8 [Gbytes]	354 [Gbytes]	636 [Gbytes]

amount is 636 [GBytes] for 100 video data.

4.4.5 Influence of Cloud-Edge Bandwidth

A less communication bandwidth between the video distribution server and the edge servers (cloud-edge bandwidth) cause more transmission overlaps. Therefore, we investigate the interruption time changing the cloud-edge bandwidth. Figure 4.11 shows the average interruption time. The arrival interval is 30 [s]. The horizontal axis is the cloud-edge bandwidth. The average interruption time

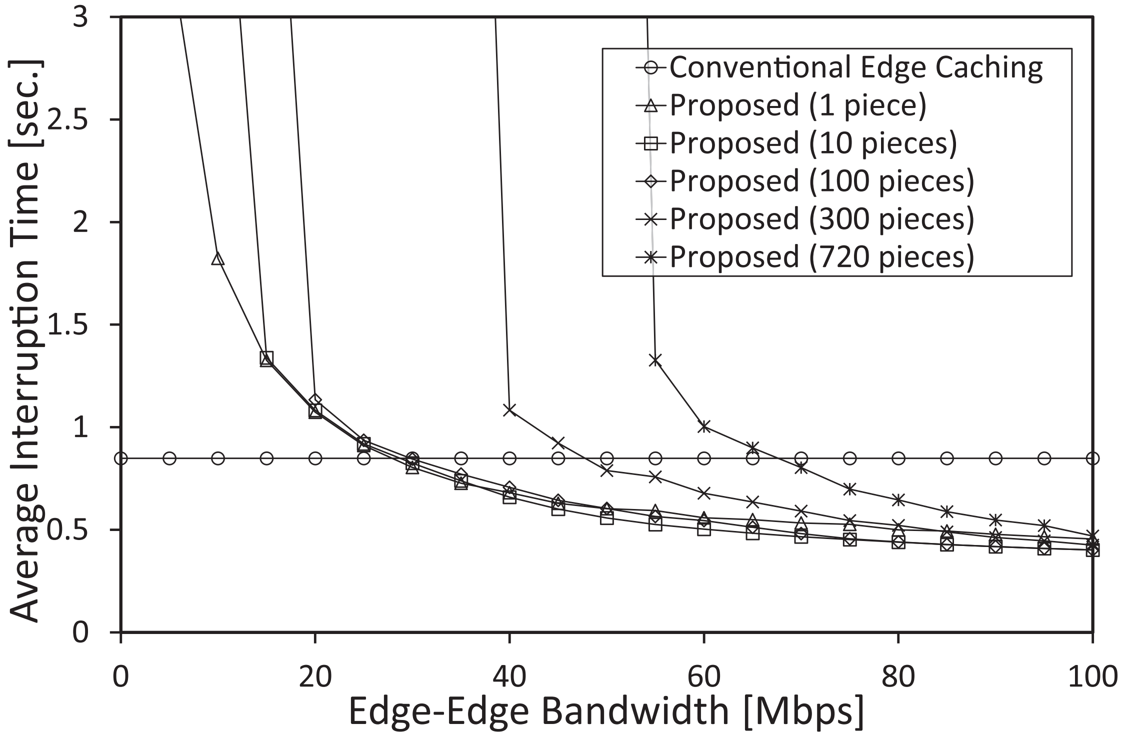


Figure 4.10: Edge-edge bandwidth and the interruption time.

under the conventional scheme is longer than that under our proposed scheme because the bandwidth between the edge servers and the clients are not sufficient and the transmissions sometimes overlap with others as shown in Fig. 4.5. Similar to previous results, the average interruption time suddenly increases when the cloudedge bandwidth is smaller than a certain value because the transmissions continue to overlap. The cloud-edge bandwidth under that the average interruption time converges is larger as the number of the pre-cached pieces increases because the edge servers receive less pieces from the video distribution server as the pre-cached pieces increase. Since the edge servers pre-cache all the pieces when they pre-cache 720 pieces, the average interruption time does not change even when the cloud-edge bandwidth changes in this case.

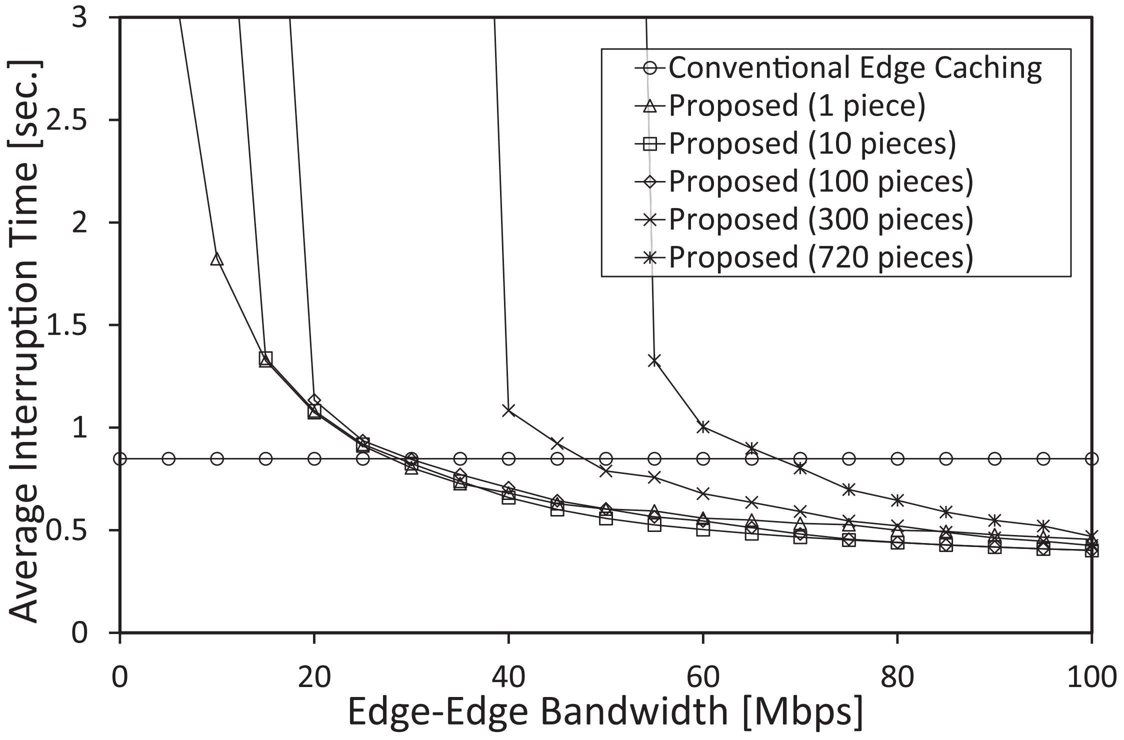


Figure 4.11: Cloud-edge bandwidth and the interruption time.

Influence of Edge Servers

The load on the edge servers can distribute as the system equips with a larger number of the edge servers. However, it takes much monetary cost as the system equips with a larger number of the edge servers. Therefore, we investigate the interruption time changing the number of the edge servers. Figure 4.12 shows the result. The arrival interval is 30 [s] and the horizontal axis is the number of the edge servers. We can see that the average interruption time is almost constant under each scheme when the number of the edge servers is large because the transmissions do not overlap if the system equips with a sufficient number of the edge servers. Otherwise, the transmissions overlap and the interruption time continue to lengthen. For example, in the evaluated situation, the interruption time diverges when the number of the edge servers is less than 6. Therefore, it is better for the system to find the appropriate number of edge servers under that the interruption time converges.

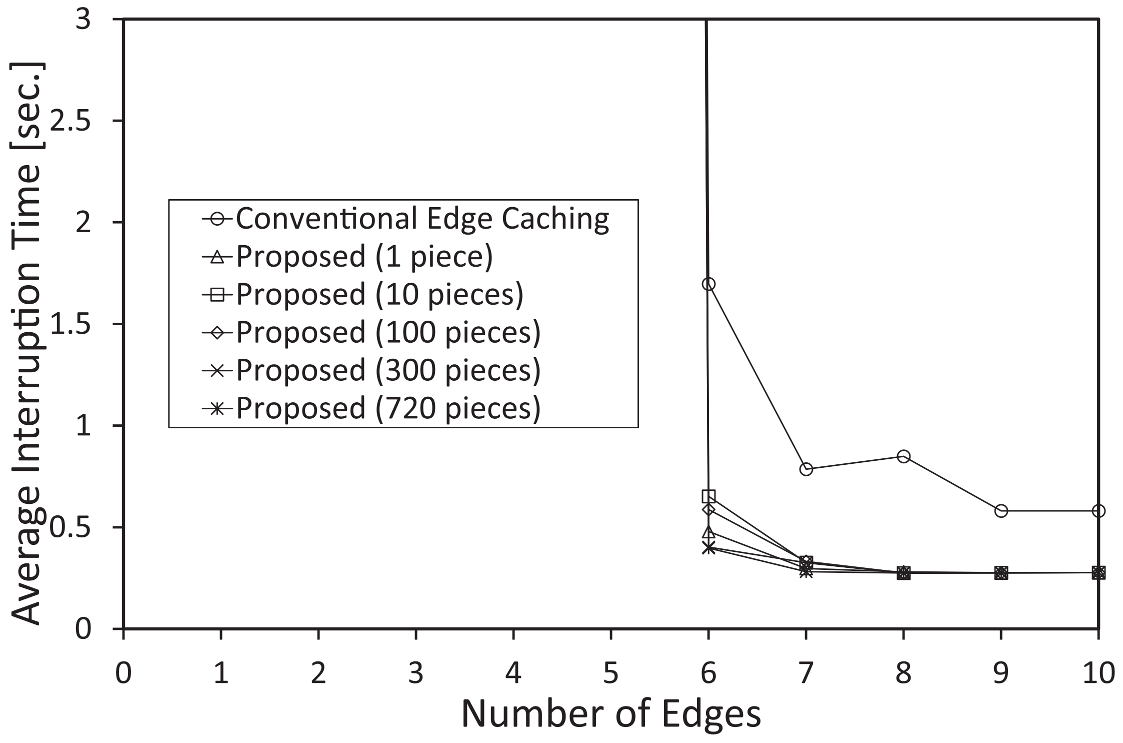


Figure 4.12: Number of the edges and the interruption time.

4.4.6 Summary of Evaluation Results

We confirmed that the average interruption time did not largely depend on the number of the pre-cached pieces when the communication network was not congested, i.e., a longer arrival interval, a larger bandwidth (edge-client/edgeedge/cloud-edge). Meanwhile, the arrival interval or the bandwidths under that the system could work, i.e., the average interruption time converges, could be improved by precaching more pieces on the edge servers. This was because the transmissions did not overlap with others when the communication network was not congested. On the other hand, a larger number of pre-cached pieces causes more computational loads on the edge servers.

4.5 Conclusion

In this paper, we proposed a scalable VoD system on edge computing environments. Our proposed system adopted the pre-caching and the redistribution techniques. Our proposed system reduces the probability that the transmissions overlap with other transmissions and increases the maximum number of the clients of that interruption time converges. Our simulation evaluation revealed that our proposed system can transmit the video data more than the conventional scheme. Our proposed system can improve the shortest arrival interval under that the system can work by 26% compared with the conventional system in the simulated situation. In the future, we will consider the popularity of the videos and adopt the dynamic caching technique to the edge servers. Moreover, we will consider the communication loads on the edge servers and the distribution server.

Chapter 5

Wireless Broadcast-Based VoD System for Efficient and Stable Playback

As part of this study, we investigated a system to optimize video processing and distribution efficiency as a wide-area distributed system for delivering collected video content.

Video distribution presents two major challenges. The first is addressing privacy concerns tailored to individuals. The second is solving scalability issues related to audience size associated with individual broadcasters. To address the latter, we proposed two distribution methods: one designed for medium-sized audiences and another for large-scale audiences.

This chapter focuses on the distribution method for large-scale simultaneous broadcasting to a vast audience. Large-scale video distribution often results in network congestion due to the transmission of massive amounts of video data, leading to playback interruptions. To resolve this issue, we proposed a data delivery method utilizing wireless broadcasting in the VHF band.

The evaluation of this wireless broadcasting-based video distribution system demonstrated that parallel transmission of video data significantly reduced waiting times before playback initiation. Furthermore, the system proved capable of providing stable viewing experiences for multiple users simultaneously.

Due to the increase in the speed of wireless communication in recent years, video-on-demand (VoD) systems has got great attention. In conventional VoD systems, when the number of clients

receiving video data increases, the problems of longer playback interruptions or longer buffering times occur. Accordingly, several research focus on the VoD systems that adopt broadcast-type distribution (B-VoD). In B-VoD systems, video files are divided into several parts and distribution servers broadcast them in the sequences of predetermined broadcast schedules to reduce interruption time. However, conventional broadcasting methods do not consider the issues that occur in actual B-VoD systems, such as initial buffering delay and the increases of data sizes caused by video file division. Interruption time can be further reduced by considering them. Hence, in this research, we propose a data segment scheduling method for B-VoD systems. We measure the performance of our proposed method on an actual B-VoD system utilizing radio broadcasting (VHF-band) to achieve a broadcast-type distribution. The results of the experiments confirmed that, under the condition of an uninterrupted playback, the waiting time until the start of a playback can be reduced by up to 85% under our proposed method by dividing the requested video data into 6 parts and broadcasting the parts in parallel.

5.1 Introduction

With the recent increase in wireless communication speeds, video-on-demand (VoD) systems are attracting increasing attention.

In the VoD system, users select the video they wish to view from a list of videos available on a website, etc., and start viewing by replaying the video from the beginning. Compared to video download systems, in which users download video files and then play them back, users do not need to wait for the video file download to complete, and many VoD services such as Amazon Prime Video and Netflix have been launched. In a VoD system, playback terminals play video while receiving video data from a distribution server. When the number of playback terminals receiving video data from a distribution server increases, and the processing and communication load on the distribution server increases, the video data is not received in time for playback, resulting in interruptions and long waiting times before playback begins. For this reason, VoD systems utilizing broadcast-type distribution have been studied. Broadcasting type delivery is a form of delivery that can deliver data to a large number of terminals at once. In these VoD systems, the delivery server can broadcast video data, so the load on the delivery server does not depend on the number of clients, and the possibility of playback interruption is reduced. In this study, we call such VoD systems broadcast VoD systems. In a broadcast VoD system, a video file is divided into several segments, and the broadcast server broadcasts the segments according to a predetermined broadcast schedule to reduce playback interruption time [40], [41], [35], [42], [43], [44]. However, conventional broadcasting schemes have not taken into account the following problems that occur in real-world broadcast VoD systems.

- Initial buffering delay
- Increase in data size due to splitting of video files

By taking these considerations into account, the broadcast server can efficiently deliver segments so that the next segment is received just before the client finishes playing each segment. This further reduces the playback interruption time. In addition, previous studies of broadcast VoD systems have measured performance using simulation programs. By measuring the performance in a real-world environment, we can confirm the usefulness of the broadcast VoD system in a real-world environment. Therefore, we propose a scheduling method for segments of a broadcast VoD system. We also measure the performance of the proposed method in a real environment. Although there have been studies evaluating broadcast-type VoD systems on intranets, the radio broadcast-type VoD system, which is a broadcast-type VoD system using wireless broadcasting, has not been realized because it requires expensive hardware and is very difficult to launch radio waves due to restrictions under the Radio Law. In this study, we realize a broadcast-type distribution system using radio broadcasting (VHF, Very High Frequency band). Experimental results show that the proposed method can be used to broadcast the requested video data in six parts without interruption. The latency time to start playback can be reduced by up to 84The results of this study confirmed the following. The contribution of this study is to develop a real-world segmentation

The main topics of this paper are a proposal for a VoD scheduling method, a demonstration experiment of a wireless broadcasting VoD system, and a comparison of conventional simulation evaluations and real-world evaluations. Section 2 introduces related research, Section 3 describes the wireless broadcast VoD system and the proposed scheme, Section 4 describes the actual system configuration, Section 5 presents the results of evaluation experiments, Section 6 provides a discussion, and finally Section 7 concludes this paper.

5.2 Related Research

To reduce the playback latency and interruption time in VoD systems, we have developed a method of duplicating and arranging video files [40] and a method of Methods using peer-to-peer technology [41], [35], [42] have been proposed. These methods consider the case where each playback terminal is communicated one-to-one using a communication network such as the Internet, and there is a problem that the load on the distribution server increases when the number of playback terminals increases. In references [45], [46], [47], a method to reduce the load on the delivery server by using multicast for one-to-many communication is proposed. These methods assume an environment where minimum bandwidth and data reachability to playback terminals are guaranteed to ensure multicasting. In the real environment, multicast bandwidth may not be guaranteed or multicast forwarding may be prohibited by routers for reasons of communication services. Several systems have been proposed that combine communications with broadcast-type distribution using wireless radio waves [48], [49], [50]. In these systems, video is delivered by broadcasting and related information is delivered by communication. However, they cannot provide on-demand video distribution. Our research group proposed a bandwidth allocation method that effectively reduces playback interruption time in an environment where video is delivered using both broadcast and communication channels [51]. However, it did not take into account the problems that occur in a real-world wireless broadcast VoD system as described in 5.1

5.3 Wireless broadcast VoD

This section first explain B-VoD systems and then introduce three methods for broadcasting video data using such a system. These methods are implemented by the actual system examined in this study and will be discuss as follows. This section first describes a wireless broadcasting VoD system, and then describes three methods for broadcasting video data in a wireless broadcasting VoD system. These methods are implemented in the actual system of this research.

5.3.1 Wireless broadcast VoD system

Figure 5.1 shows the assumed environment of a wireless broadcasting VoD system. The broadcasting server holds video files and is connected to the broadcasting equipment. The video file can be divided into several parts (segments), and the broadcasting server can broadcast segments via the broadcasting facility. Playback terminals within the broadcast range can receive the segments broadcast by the broadcasting facility. Playback terminals can play back each of the received segments. For example, a video file in the case of dividing a 60 [s] video into N segments Si (i = 1, ... ,N), the playback terminal can start playing the video when it has received S . 1If a 60 [s] video is equally divided into 6 segments, S1 contains the first 10 [s] of the video. In a wireless broadcast VoD system, the broadcasting server divides each segment into smaller data structures called chunks and broadcasts them. Chunks are smaller than segments. When broadcasting chunks, the broadcast server adds a segment number (i in Si) and a chunk number (a number assigned to each chunk in order from the top of the data in Si). This allows a playback terminal to receive chunks from the middle of a segment even if it starts receiving while a segment is being broadcast. Digital terrestrial broadcasting and satellite broadcasting are examples of broadcast-type distribution systems that can realize the wireless broadcasting necessary for the construction of such a wireless broadcasting-type VoD system. Broadcasting-type distribution systems other than wireless broadcasting include cable TV broadcasting and bandwidth-guaranteed IP protocol broadcast addresses and multicast addresses. In MPEG, a video file consists of a set of frames called a GoP (Group of Pictures), and a playback terminal can play back the video for each GoP. In HLS (HTTP Live Streaming), the video file is divided into several video data called TS files, and the playback terminal can play each TS file independently.

5.3.2 Conventional video data broadcasting method

This section describes a method for reducing the waiting time (playback waiting time) between the start of reception of broadcast data by a playback terminal and the start of video playback in a wireless broadcast-type VoD system.

Conventional methods are described. First, a simple broadcast method is described, followed

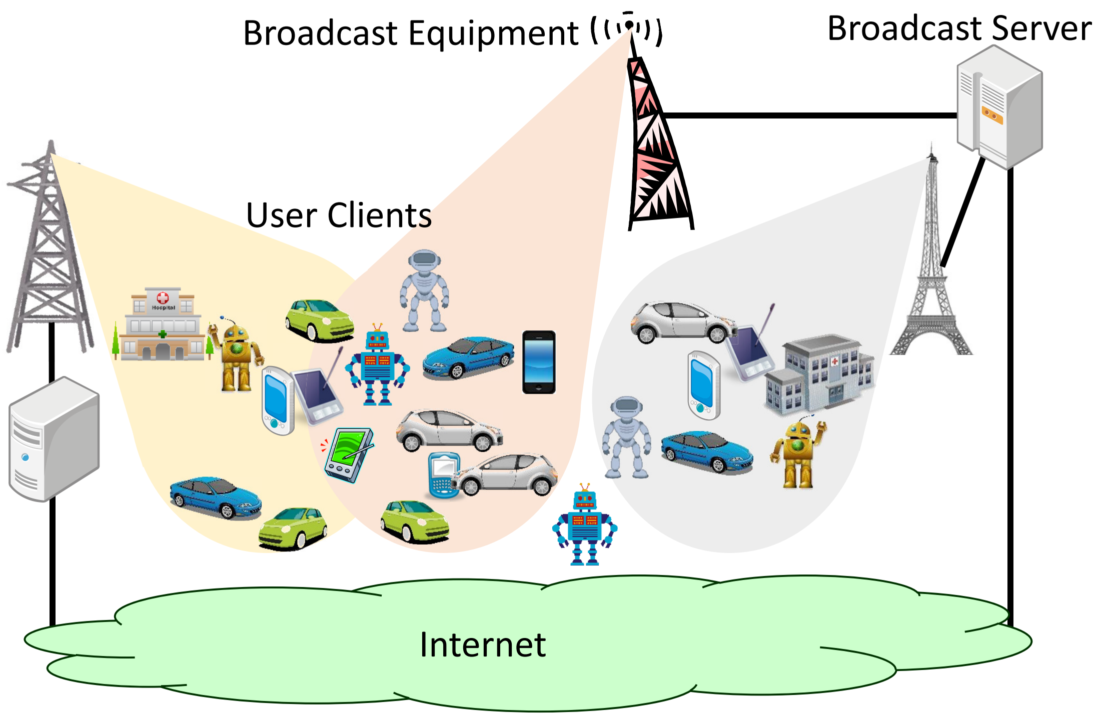


Figure 5.1: An assumed system environment for broadcast-type VoD systems.

by a method that reduces latency compared to the simple method.

(1) Simple method In the simple method, video files are repeatedly broadcast. Figure 5.2 shows an image of the simple method. The broadcasting facility repeatedly broadcasts the same video file using broadcast bandwidth B. The broadcast bandwidth is the available bandwidth that the broadcasting server can use to broadcast the data. The broadcast bandwidth is the bandwidth available to the broadcasting server for broadcasting data. The data size of the video file D is the data size of the video file broadcast by the broadcasting server. Number of segments N is equal to the number of segments in the video file. For example, if a 4.75 [Mbytes] video file is divided into 6 segments and broadcast with a bandwidth of 3.8 [Mbps], B = 3.8 [Mbps], D = 4.75 [Mbytes], and N = 6. The average playback latency of the simple method is described and calculated separately for the playback terminals that started receiving the video during the broadcast of S1 and for other playback terminals (Figure 5.3).

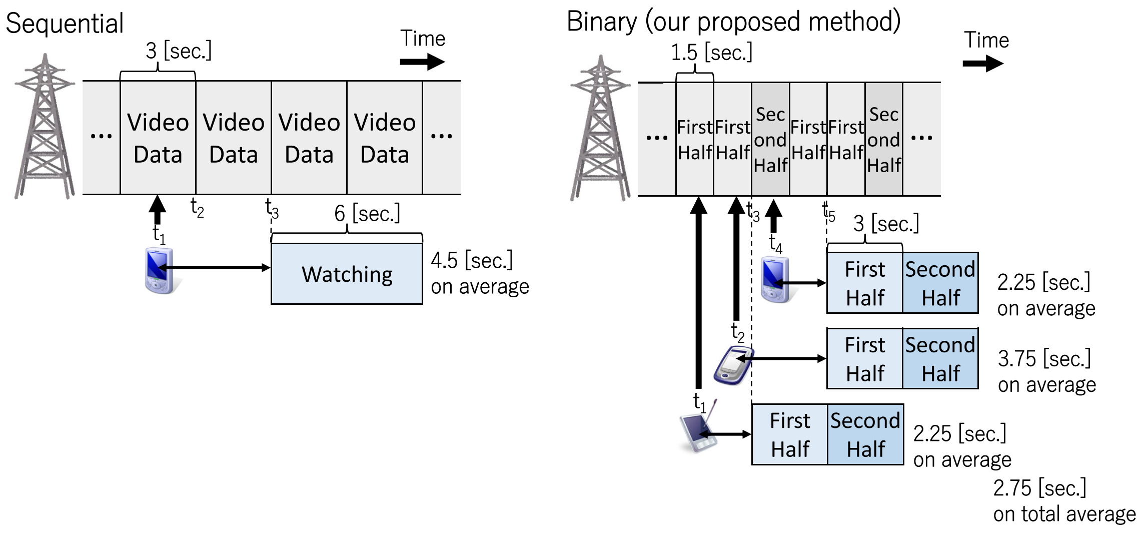


Figure 5.2: Images of the simple method and the binary method. SEC WRONG!

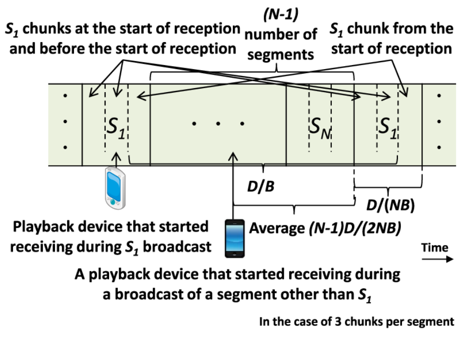

S1 chunks at the start of reception and before the start of reception
(N-1) number of segments
S1 chunk from the start of reception
S1
...
SN
S1
D/B
D/(NB)
Playback device that started receiving during S1 broadcast
Average (N-1)D/(2NB)
Time
A playback device that started receiving during a broadcast of a segment other than S1
In the case of 3 chunks per segment

図3

Figure 5.3: An image for explaining the waiting time under the simple method.

The playback device that started receiving during the broadcast of S1 (the device on the left in Figure 5.3) can receive the chunks that come after the beginning of S1. In order for this playback device to reconstruct S1, it is necessary to receive the chunks in S1 that come before the beginning of S1, as well as the beginning of S1 itself. In the simple method, since each segment is broadcast in order repeatedly broadcast in order, so it will take D/B until it has received the chunks in the front. This D/B is the average time it takes for a playback terminal that started receiving during the broadcast of S1 to complete receiving S1 (i.e. the average playback waiting time). In the simple method, it takes D/B for the broadcast of all segments and D/(NB) for the broadcast of S1, so the proportion of playback terminals like this during the broadcast of a video file is 1/N. segments take D/B, and broadcasting S1 takes D/(NB), so the percentage of playback terminals like this during a single broadcast of a video file is 1/N.

Next, the playback terminal that starts receiving during the broadcast of a segment other than S1(the terminal on the right in Figure 5.3) will wait an average of (N - 1)D/(2NB) until the broadcast of S1 starts. This can be calculated from (N - 1)D/(NB) + 0/2, as the maximum wait time is (N - 1)D/(NB) and the minimum is no wait time. This playback terminal starts receiving the broadcast of S1 when the broadcast of S1 starts. The time from when it starts receiving S1 until it finishes is D/(NB). Therefore, the average playback waiting time for playback terminals that start receiving during the broadcast of segments other than S1 is (N - 1)D/2NB+D/NB = (N + 1)D/(2NB). is (N - 1)D/(NB), so the proportion of such playback terminals during a single broadcast of a video file is (N - 1)/N. From the above, the average playback waiting time for the simple method is given by 1/N ×D/B + (N - 1)/N × (N + 1)D/(2NB), which is

$$\text{Average Waiting Time} = \left\{ \frac{(N-1)(N+1)}{2N} + 1 \right\} \cdot \frac{D}{NB} \tag{5.1}$$

.

In the first-half-second method, segments are classified into those included in the first half of the video file (first-half segments) and those included in the second half (second-half segments), and the first-half segments are broadcast frequently. By broadcasting the first-half segments frequently, the playback terminal has more opportunities to receive the first segment, and the average playback waiting time can be shortened. , and the average playback waiting time can be shortened. The image of the first-half-second method is shown in Figure 5.2. By determining the number of broadcasts of the first-half segment group so that the playback terminal can meet the conditions for receiving the second-half segment group completely during playback of the first-half segment group, the playback terminal can play the second-half segment group without interruption after the first-half segment group has finished playing. If the bit rate of the video file is R, then according to the discriminant formula A, after broadcasting the first half of the segments the number of times shown below, and then broadcasting the second half of the segments once, it is known that the above conditions are met and the shortest playback waiting time is given [43].

$$\begin{aligned} a &= \frac{B}{R}, \\ A &= \lfloor a \rfloor (a - \lfloor a \rfloor) - \lfloor a \rfloor, \\ \text{if } A < 0: &\quad \lfloor a \rfloor \text{ times}, \\ \text{if } A > 0: &\quad \lceil a \rceil \text{ times}. \end{aligned} \tag{5.2}$$

The example in Figure 5.2 is the case where A ¡ 0, and the first half of the segments are broadcast twice, and the second half is broadcast once. The average playback waiting time when waiting until playback does not stop is given by the following equation.

$$\text{Average Waiting Time} = \begin{cases} \frac{\alpha+3}{2\alpha+2} \times \frac{D}{B} & \text{if } A < 0, \\ \frac{3\alpha+6-2\lfloor\alpha\rfloor}{2\alpha+4} \times \frac{D}{B} & \text{if } A > 0. \end{cases} \tag{5.3}$$

5.3.3 Proposed Method

Proposed Initial Buffering based Parallel Broadcasting (IBPB) method divides a video file into several segments and broadcasts them in parallel, taking into account the initial buffering data size. (It is called the (IBPT) method. The details are explained below.

(1) Ideas to reduce playback waiting time for IBPB method The IBPB method divides a video file into N segments. In a wireless broadcast-type VoD system, the client receives and stores the beginning of the video data for initial buffering before starting playback, so the IBPB method divides the video file so that the data size of the first segment is the same as that of the initial buffering. In the IBPB method, each segment is broadcast periodically and In order to reduce the playback latency, the broadcast server can use the segment are broadcast in parallel. For this reason,

the broadcast bandwidth is also increased by N broadcasts. The segments are divided into channels, and each segment is broadcast periodically on each broadcast channel.

(2) How to split a video file using the IBPB method In the IBPB method, a video file is divided into segments Si (i = 1,..., N) and broadcast using N broadcast channels with N equal segments in the broadcast bandwidth. N) and broadcast using N broadcast channels with the broadcast bandwidth divided into N equal segments. Figure 5.4 shows an image of the IBPB method. By determining the data size of each segment to satisfy the condition that reception of Si+1 can be completed by the time the playback terminal finishes playing Si , the playback terminal can play from the beginning to the end without interruption [44]. The data size of Si+1 is denoted by Di+1 . Since the broadcast bandwidth of each broadcast channel is B/N, it takes NDi+1 /B to broadcast Si+1 . Since it takes (D1 + — + Di)/R to finish playing Si , Di+1 is given by

$$D_{i+1} = \frac{B}{NR}\sum_{j=1}^{i} D_j \quad \text{if} \sum_{j=1}^{i+1} D_j < D \tag{5.4}$$

From this, when D1 is determined, D2 is determined and D2 is determined 31The IBPB method calculates D1 satisfying the following equation

$$\sum_{j=1}^{N} D_j = D \tag{5.5}$$

2 broadcasts and then one broadcast of the second half segment group In other words, the total data size of all segments is the data size of the image file is equal to the data size of the image file. However, if D1 satisfying equation (5) is smaller than the data size of the initial buffering, D1 is set to the same data size as the data size of the initial buffering because the data needed for buffering is also included in the second and subsequent segments, and reception takes longer. This allows an upper limit to the number of segments.

When a video file is divided, the data size increases because an identifier is added to indicate which part of the video file before the division is included in the divided part. The IBPB method solves this problem by dividing a video file so that it satisfies Equation (5) using the actual data size after the division. This allows the data size to be taken into account when dividing the file,

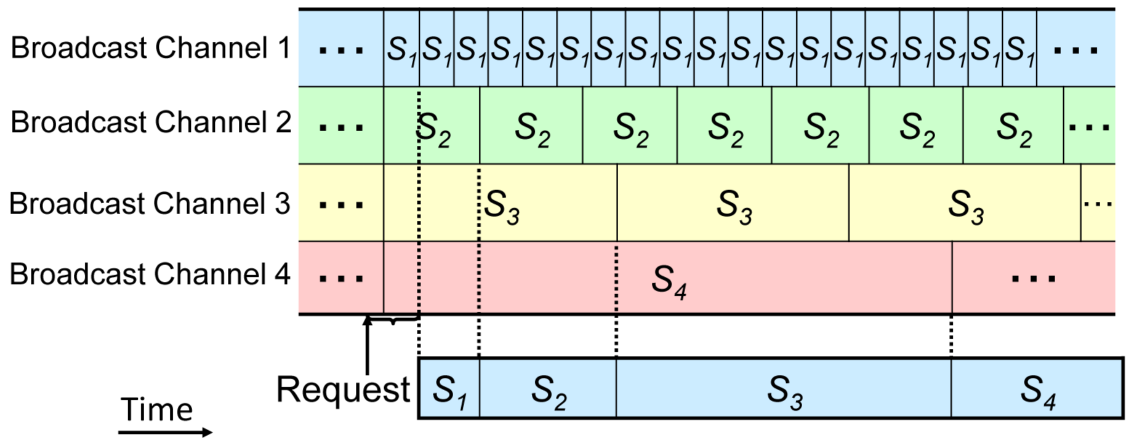


Figure 5.4: An image of the IBPB method.

so the IBPB method can broadcast segments after solving the problem. For example, when using ffmpeg (software that can perform various video-related processes [52]) to divide video files, the playback time of each segment is required. However, due to the GoP structure explained in Section 3.1, it is not always possible to obtain the given playback time. In the empirical experiments in this paper, the operation of dividing the video by giving the playback time was repeated, and D1 was determined to satisfy Equation (5)

(but not less than the size of the initial buffering data). The average playback waiting time is equal to the average time until S1 is completed, and is

$$\text{Average Waiting Time} = \frac{ND_1}{B} \tag{5.6}$$

.

5.4 Conventional video data broadcasting method

In this section, we will explain the wireless broadcasting-type VoD system that we designed and implemented.

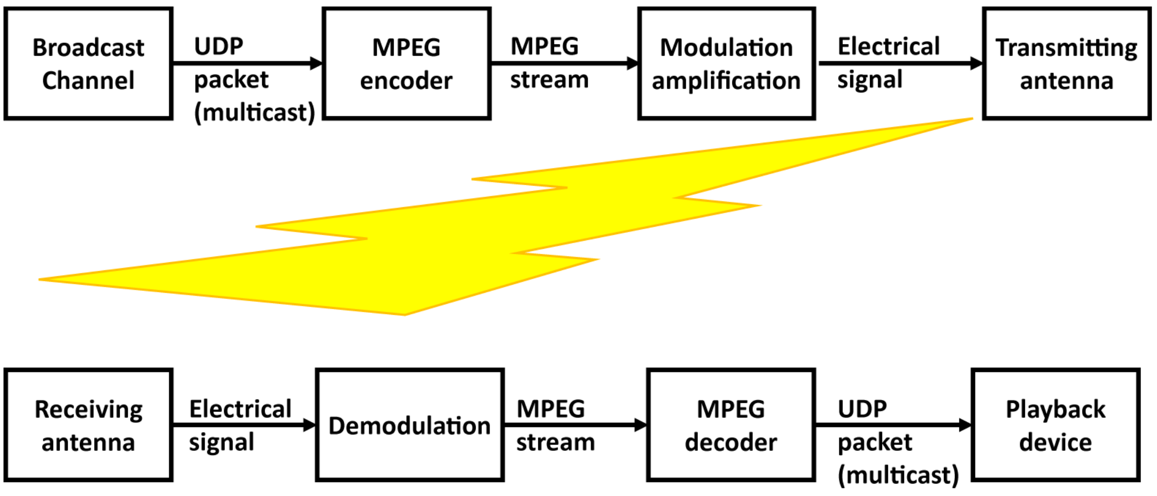


Figure 5.5: An architecture of our designed wireless-broadcast-type VoD system.

5.4.1 system design

Figure 5.5 shows a diagram of the designed wireless broadcast VoD system. The upper part shows the transmitter side and the lower part shows the receiver side. On the transmission side, the broadcast server divides the video data into units called chunks and sends them to the MPEG encoder. The data transmission speed to the MPEG encoder. The MPEG encoder uses IPDC (IP Data Cast) to transmit IP packets over the airwaves in order to broadcast the received UDP packets over the airwaves. The modulated and amplified electrical signal is emitted as radio waves via a transmission antenna. The receiver receives the emitted radio wave with a receiving antenna, demodulates it to obtain an MPEG stream, extracts UDP packets with an MPEG decoder, and sends them to the playback terminal. The playback terminal reconstructs chunks from the UDP packets and stores the segments when it receives all the chunks that make up each segment. The replay terminal plays each segment when it is time to play it. In Figure 5, only one receiver is shown, but multiple receivers can also operate.

5.4.2 System Implementation

This section describes the implemented wireless broadcast VoD system.

(1) delivery software The distribution software is software that runs on the broadcast server to

broadcast video files.

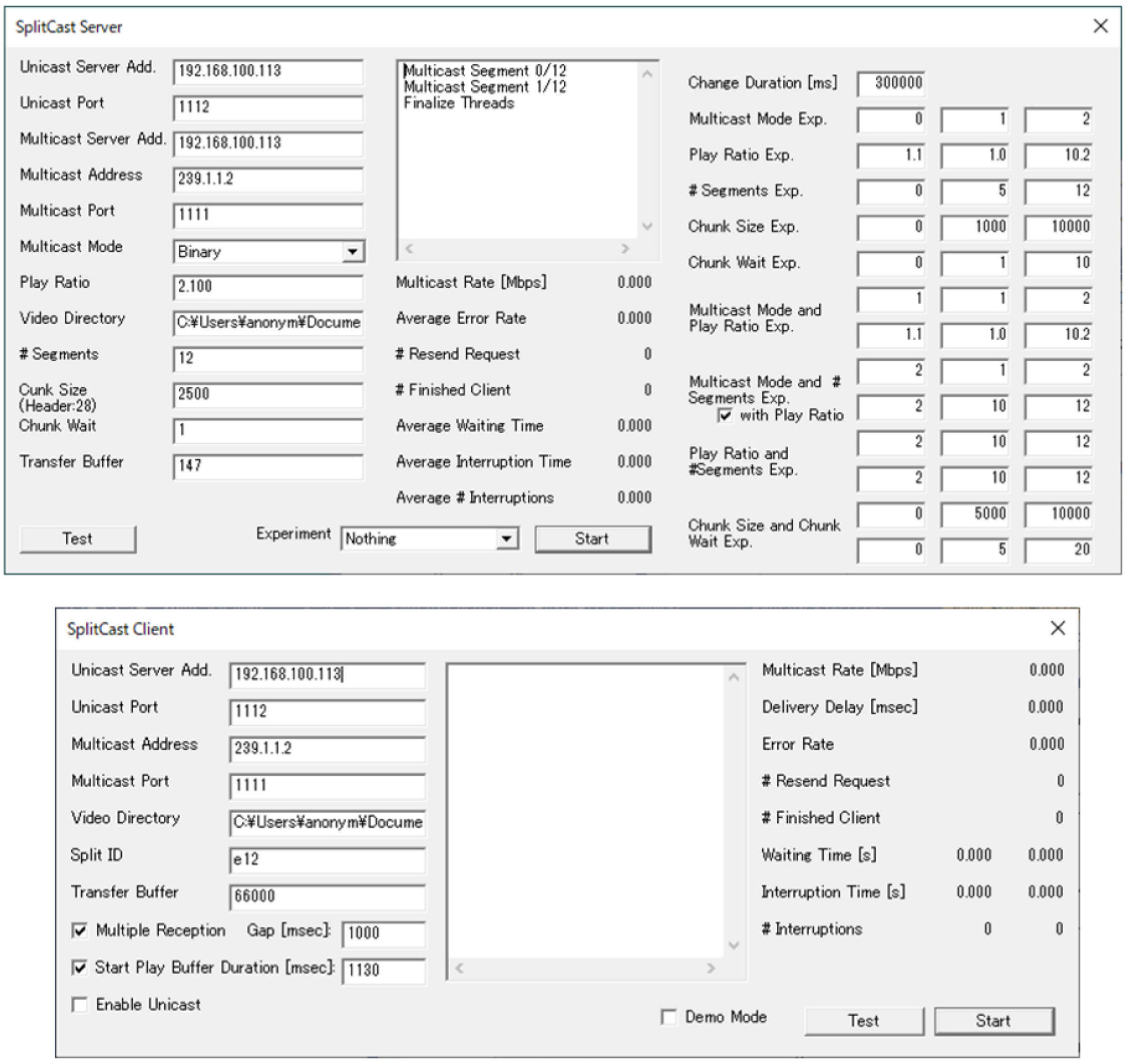
Figure 5.6 shows a screenshot of the delivery software implemented in (above). On the left side of the screenshot is the section for configuring distribution settings. The server address for sending UDP packets is can be adjusted by changing the data size and ansmission interval of the chunks. It is possible to specify the port number, port number, and multicast address. It is also possible to select a broadcasting method as described in section 3.2 and set parameters for each method. The location of video files and the number of segments can also be specified. The transmission interval of chunks, the data size of chunks, and the data size of UDP packets can also be specified to adjust the broadcast bandwidth. The upper center part is the log output for operation checks. On the right side of the screenshot is the area for specifying the range where each parameter is automatically changed for evaluation experiments.

(2) Receiving Software

Receiver software is software that runs on the playback terminal to receive and play video dataFigure 5.6 (bottom) shows a screenshot of the receiver software implemented in 10. The left side of the screenshot shows the reception settings: multicast address and port number for receiving UDP packets, and destination for storing video files. You can also switch modes for evaluation experiments. The center part is the log output for operation checks. The right side of the screenshot shows information on reception, such as broadcast bandwidth, latency, and error rate. HLS was used as the video file structure, and the using TS file corresponds to a segment. The received video files are played back ffplayer , which is widely used in the field of research and development together with ffmpeg files with many different video encodings. of the TS because it can play back video ffplayer requires buffering file for HLS playback, so the buffering time can be specified in the receiving software. Because ffplayer requires buffering of TS files for HLS playback, the buffering time can be specified in the receiving software.

(3) hardware (esp. computer)

Figure 5.7 shows a photograph of the of the wireless broadcast VoD system implemented hardware , and Table 5.1 shows the details of the hardware. The hardware was connected based on the esign described in section 3.2



SplitCast Server

Unicast Server Add.	192.168.100.113
Unicast Port	1112
Multicast Server Add.	192.168.100.113
Multicast Address	239.1.1.2
Multicast Port	1111
Multicast Mode	Binary
Play Ratio	2.100
Video Directory	C:¥Users¥anonym¥Docume
# Segments	12
Cunk Size (Header:28)	2500
Chunk Wait	1
Transfer Buffer	147

```
Multicast Segment 0/12
Multicast Segment 1/12
Finalize Threads
```

Multicast Rate [Mbps]	0.000
Average Error Rate	0.000
# Resend Request	0
# Finished Client	0
Average Waiting Time	0.000
Average Interruption Time	0.000
Average # Interruptions	0.000

Change Duration [ms]	300000		
Multicast Mode Exp.	0	1	2
Play Ratio Exp.	1.1	1.0	10.2
# Segments Exp.	0	5	12
Chunk Size Exp.	0	1000	10000
Chunk Wait Exp.	0	1	10
Multicast Mode and Play Ratio Exp.	1	1	2
	1.1	1.0	10.2
Multicast Mode and # Segments Exp. ☑ with Play Ratio	2	1	2
	2	10	12
Play Ratio and #Segments Exp.	2	10	12
	2	10	12
Chunk Size and Chunk Wait Exp.	0	5000	10000
	0	5	20

Experiment: Nothing

SplitCast Client

Unicast Server Add.	192.168.100.113
Unicast Port	1112
Multicast Address	239.1.1.2
Multicast Port	1111
Video Directory	C:¥Users¥anonym¥Docume
Split ID	e12
Transfer Buffer	66000
☑ Multiple Reception Gap [msec]:	1000
☑ Start Play Buffer Duration [msec]:	1130
☐ Enable Unicast	

Multicast Rate [Mbps]		0.000
Delivery Delay [msec]		0.000
Error Rate		0.000
# Resend Request		0
# Finished Client		0
Waiting Time [s]	0.000	0.000
Interruption Time [s]	0.000	0.000
# Interruptions	0	0

☐ Demo Mode

Figure 5.6: Screenshots of our developed distribution software (top) and reception software (bottom).

5.5 evaluation experiment

This chapter describes the evaluation experiments. First, the playback latency and interruption time for each method are shown. Then, the simulation results are compared with the actual results, focusing on the following points

- Feasible Broadcast Bandwidth

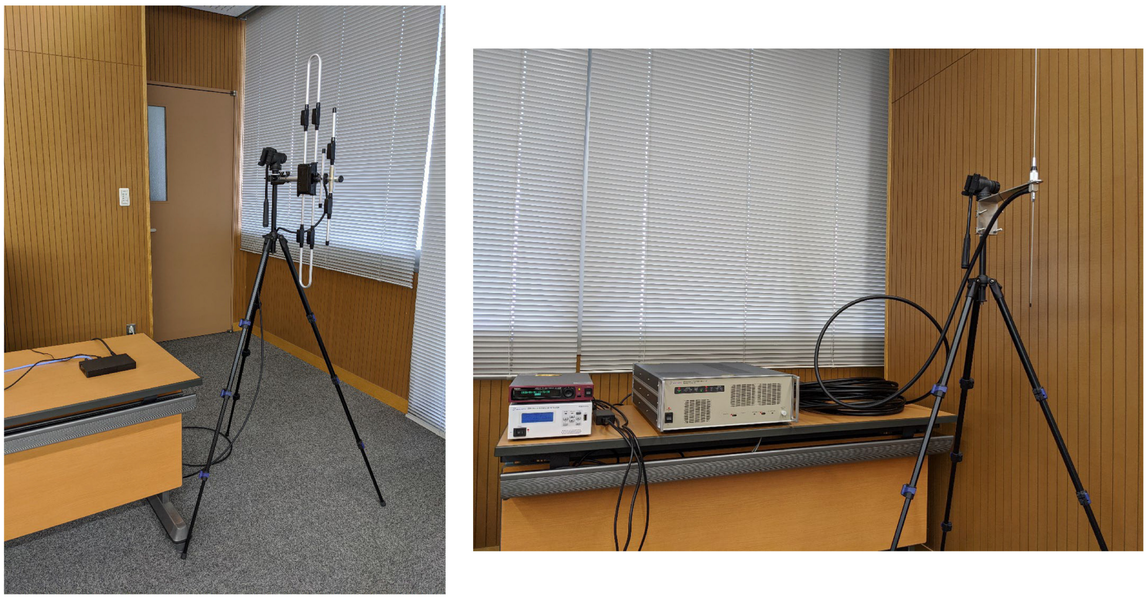


Figure 5.7: Hardware photos.

- Delayed delivery due to actual processing
- Buffering of video data

5.5.1 Evaluation Experiment Environment

In the evaluation experiment, we used the actual system described in section 4.2. To create the video files for evaluation, we used Buck BUNNY, a 4K stereo video published in [53], and since the broadcast bandwidth was approximately 3.8 [Mbps], we used 650 [Kbps] for 60 [s]. (480 × 270, 20 [fps]) video, transcoded by ffmpeg and used. The reasons for these values are explained in detail in section 6.3. In addition, the playback time and bit rate are evaluated when they are changed in section 5.3. The chunk transmission interval was set to 1 [ms] and the chunk size t o 7.2 [Kbytes] (for details, see (Section 5.5). Segmentation in the front-back half method and IBPB method was performed using ffmpeg, specifying the pre calculated playback time for each segment. The initial buffering data size was set to the equivalent of 1 [s] (details are described in Section 5.4 (3)). For the V-High band radio emissions, we obtained cooperation from the IPDC Forum, which holds the license for the V-High specific experimental test station. Experiments The first meeting of

Table 5.1: **Hardware list.**

Broadcast Server and Playback Terminals (2 units)	Manufacturer : VAIO Model: VAIO OS: Windows 10 Home CPU: Intel Core i7 Main Memory: 16 GB SSD: 256 GB NIC: Realtek PCIe GbE Family Controller
MPEG Encoder	Manufactured by ASTRO CX-5528
Modulator	Manufactured by Eiden Co., Model MSH3000A Frequency JP 12ch (219 MHz)
Amplifier	Manufactured by Eiden Co., Model 4034A Power Output: 10 mW $\rightarrow$ 1W
Transmission Antenna	Base Station Dipole Antenna
Reception Antenna	Manufactured by Maspro Denkoh Co., Model 162E3-P
Demodulator and MPEG Decoder	Manufactured by SKnet, IPDC Receiver

the ”International Conference on the Future of Education and Research” was held on the afternoon of August 5, 2020, in a conference room on the 7th floor of the Toyonaka Education and Research Building of the Cybermedia Center, Osaka University.

5.5.2 Playback latency and interruption time

In this section, we show the playback latency and interruption time for the implemented broadcast method. For the front-and back half method, the number of broadcasts in the first half segment group was varied, so the results for the simple method and the IBPB method are shown first, followed by the results for the front and-back half method. For the measurement, it was assumed that the playback terminal started receiving the video at the same time as the broadcast server started broadcasting at time 0 [s], and when the playback terminal started playing the video, the other playback terminals started receiving the video 1 [s] later.

(1) Simple and IBPB methods In the simple method and the IBPB method, the number of

segments, which is the number of divisions in a video file, affects the playback latency. Therefore, we measured the average playback latency by varying the number of segments. In the simple method, segments are simply repeated in order and broadcast. The results are shown in Figure 5.8, where Actual is the actual measurement result and Simulation is the simulation result. Simulation results where Simple are obtained from Equations (1) and (5), indicates the IBPB method. simple method and IBPB In the IBPB method, the data size of S1 [s], which is the data size of the initial indicates the is given as 1 buffering, so the graph is up to 6 segments in Eq. (4) is 5). (the maximum i that satisfies the if condition It can be seen that the average playback latency decreases as the number of segments increases. This is because the larger the number of segments, the smaller the data size of the first segment and the shorter the time required for reception. Since the playback terminal starts playing the video after receiving the first segment, the larger the number of segments, the shorter the average playback latency. However, the amount of reduction tends to become smaller as the number of segments increases. This is because the data size of a segment is inversely proportional to the number of segments. In this experimental environment, the broadcast bandwidth is larger than the video playback rate, so the simple method does not cause any interruption during video playback. In the IBPB given in equation (4) method, the data size of each segment was so that no nterruption would occur, and no interruption occurred. For example, when a video file is roadcast in 6 segments (short) The average playback latency of the simple method is 6.45 [s] and that of the IBPB method is 1.01 [s], indicating a maximum 84% reduction.

(2) front-back half-method The average playback latency and interruption time were measured by varying the number of broadcasts of the first half segment group in the front and back half method.

Figure 5.9 shows the results of the average playback latency. The horizontal axis represents the number of broadcasts of the first half of the segment group, and the vertical axis represents the average playback latency. The graph shows that the more frequently the first half of the segment group is broadcasted, the shorter the average playback latency tends to be. This is because the first half of the segment group is broadcasted more frequently, increasing the opportunities for playback terminals to receive S 1 . Figure 5.10 shows the results of the average interruption time. This graph shows that It can be seen that the interruption occurs when the number of broadcasts of the first half

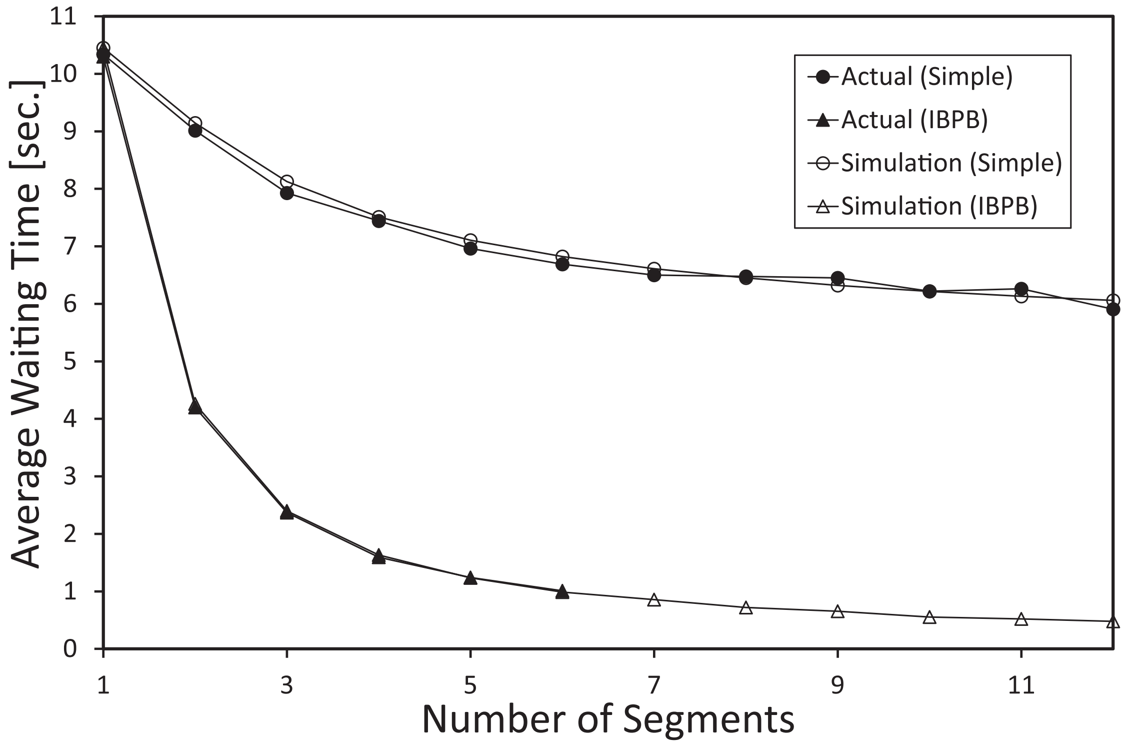


Figure 5.8: Average waiting time under the simple method and the IBPB method.

segment group is greater than 6, and the interruption time increases proportionally thereafter. This is because the more the number of broadcasts of the first half segment group, the longer the interval between broadcasts of the second half segment group becomes, and the smaller the probability that the playback terminal can receive the second half segment group while the first half segment group is being played. The average playback latency in this case is 3.41 [s], which is shorter than the simple method but longer than the IBPB method. This is shorter than the simple method, but longer than the IBPB method.

5.5.3 Playback Time and Bit Rate

Average playback latency is the video playback time and bit rate The results of this section are based on a simulation. Note that the evaluation results in this section are based on simulations, since they were conducted at a later date after the real-world experiments. However, as shown i n Figure 5.8, there is no significant difference between the actual and simulated performance of

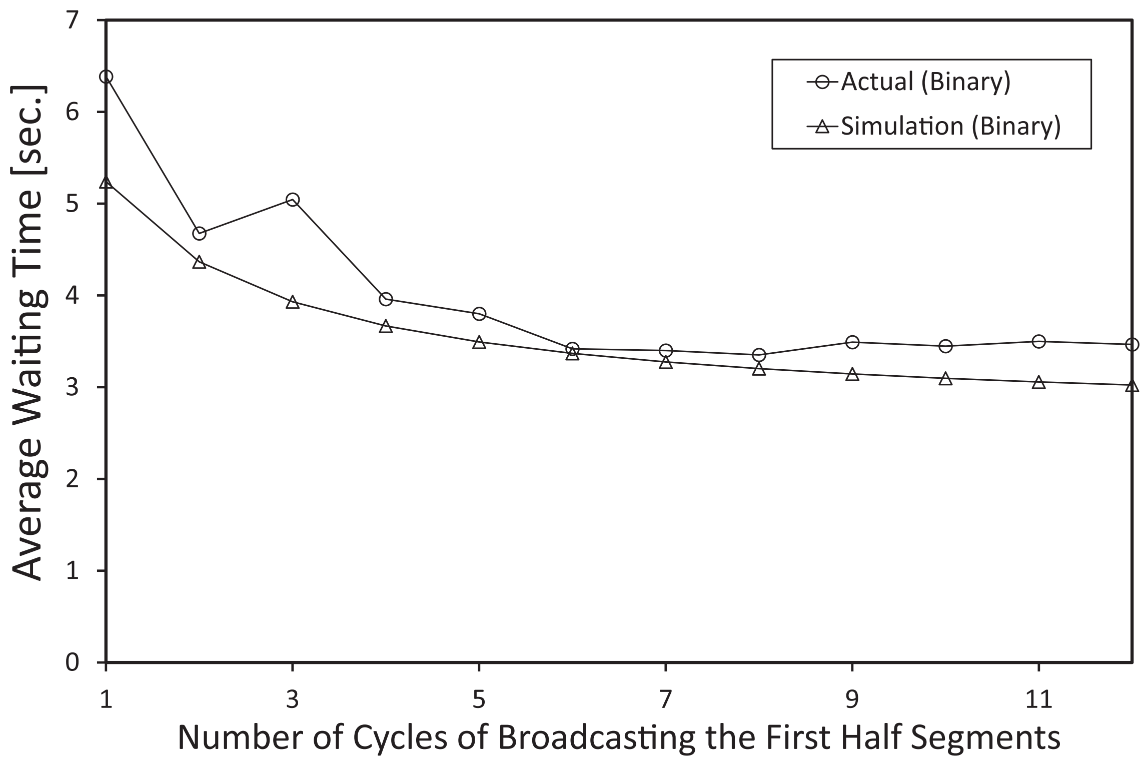


Figure 5.9: Average waiting time under the binary method.

the simple method and the IBPB method when the number of divisions is 6 or less. Figure 5.11 shows the average playback latency for different video playback times. The broadcast bandwidth and bit rate were set to 3.8 [Mbps] and 650 [Kbps], espectively. The broadcast bandwidth and bit rate were set to 3.8 [Mbps] and 650 [Kbps], respectively. Since the average playback latency for the simple and IBPB methods with different number of segments is shown in Figure 5.8, the cases with 2 (N = 2) and 6 (N = 6) segments were simulated as representative examples. The number of broadcasts in the first half of the segment group was given by Equation (2) to avoid interruptions in the front and back half method. The horizontal axis is the video playback time, and the vertical axis is the average playback latency until the video starts playing. The results show that the average playback latency is proportional to the playback time in all cases. For the simple and front-and-back methods, the average playback latency is proportional to the playback time, since equations (1) and (3) give the average playback latency, while for the IBPB method, the longer the playback time, the larger D1 satisfying equation (5) becomes, and thus the average playback latency increases. This

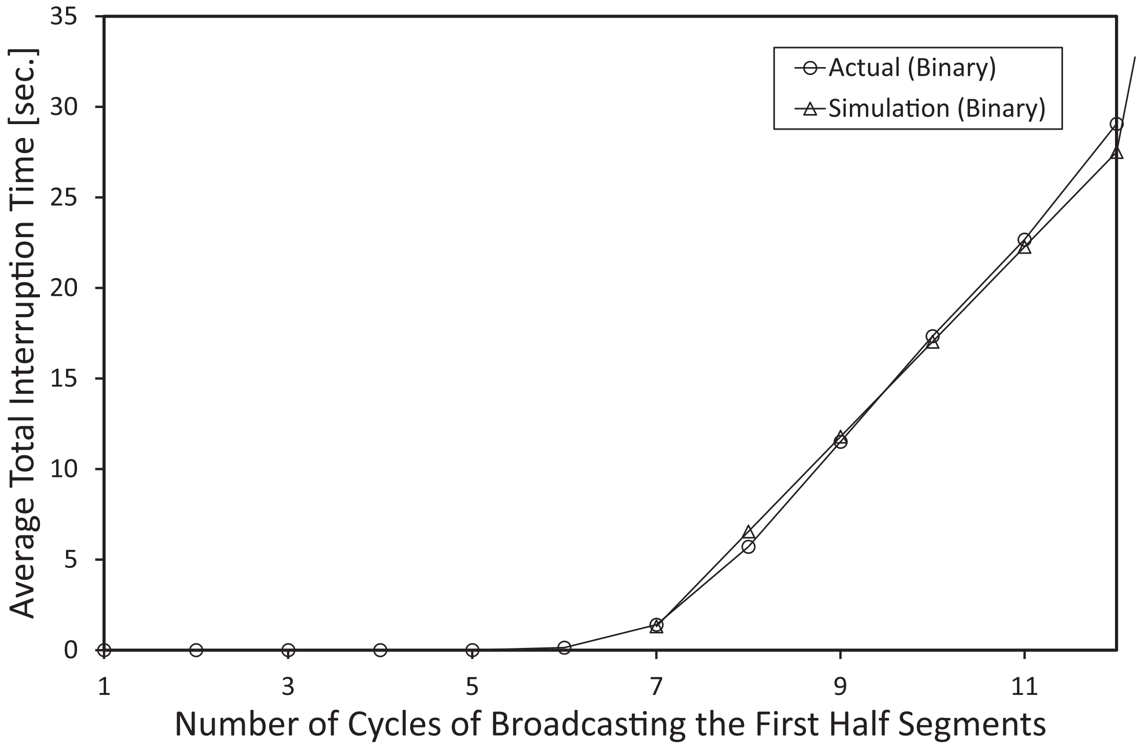


Figure 5.10: Average interruption time under the binary method.

result indicates that the ranking of average playback latency among the methods is independent of playback time. Next, Figure 5.12 shows the average wait time for different bit rates. The playback time is shown below. The broadcast bandwidth was set to 3.8 [Mbps] and the playback time to 60 [s]. As in the previous section, the cases with 2 (N = 2) and 6 (N = 6) segments are simulated as representative examples, and the number of broadcasts of the first half segment group is given by Equation (2) for the front-back half method. The horizontal axis is the bit rate and the vertical axis is the average playback latency. The larger the bit rate, the larger the data size of the video file, and the longer the average playback latency. The average playback latency of the 6 split IBPB method is shorter than these. In the front-and-back half method, the average playback latency changes significantly as the number of broadcasts of the first half of the segment group changes, and is therefore depicted as changing in stages. In the front-back half method, as the bit rate increases, segments are broadcast in the same order as in the simple method, so the average playback latency is the same as in the simple method. This result indicates that the order of shortest average playback

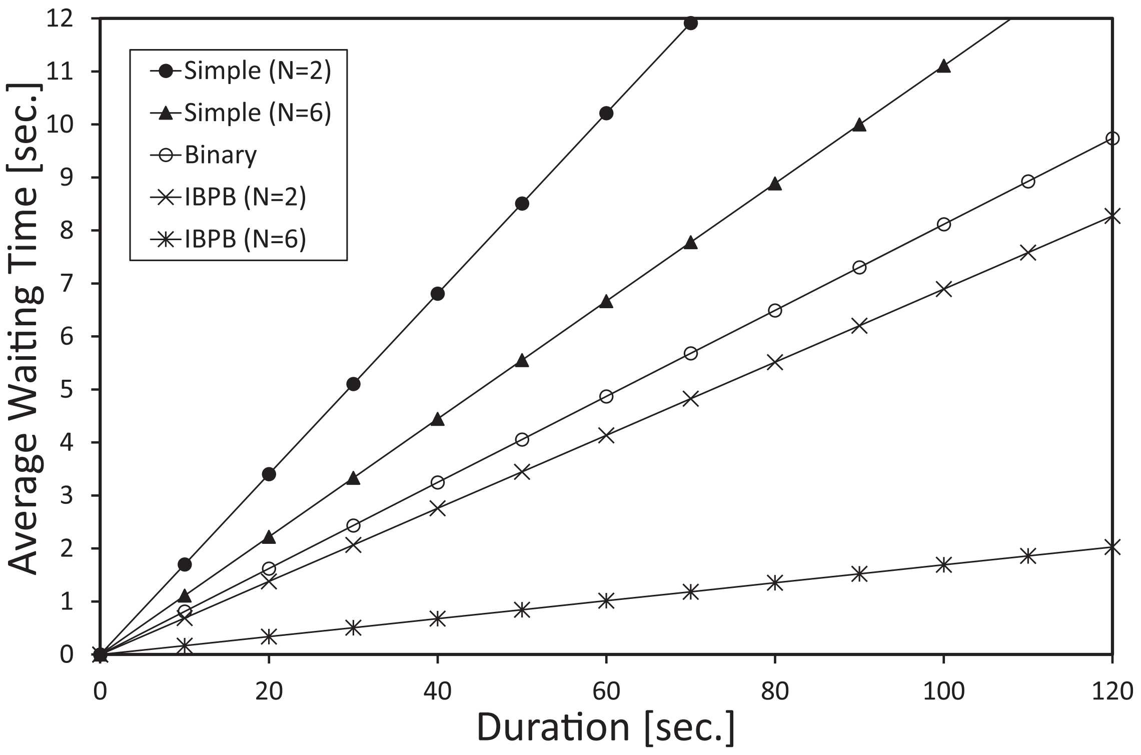


Figure 5.11: Average interruption time under the binary method.

latency for each method is independent of bit rate.

5.5.4 Difference from simulation

In this section, we explain the differences from the simulation results that were revealed in the real system evaluation.

(1) Achievable Broadcast Bandwidth In simulations, various broadcast bandwidths can be set, but in real systems, there is an upper limit to the available broadcast bandwidth due to hardware and software processing performance and other reasons. Therefore, we investigated the feasible broadcast bandwidth. We varied the broadcast bandwidth by changing the chunk data size and transmission interval. The investigation revealed that when the speed at which the broadcast server sends out UDP packets exceeds 3.8 [Mpbs], UDP packets are not sent out from the demodulator.

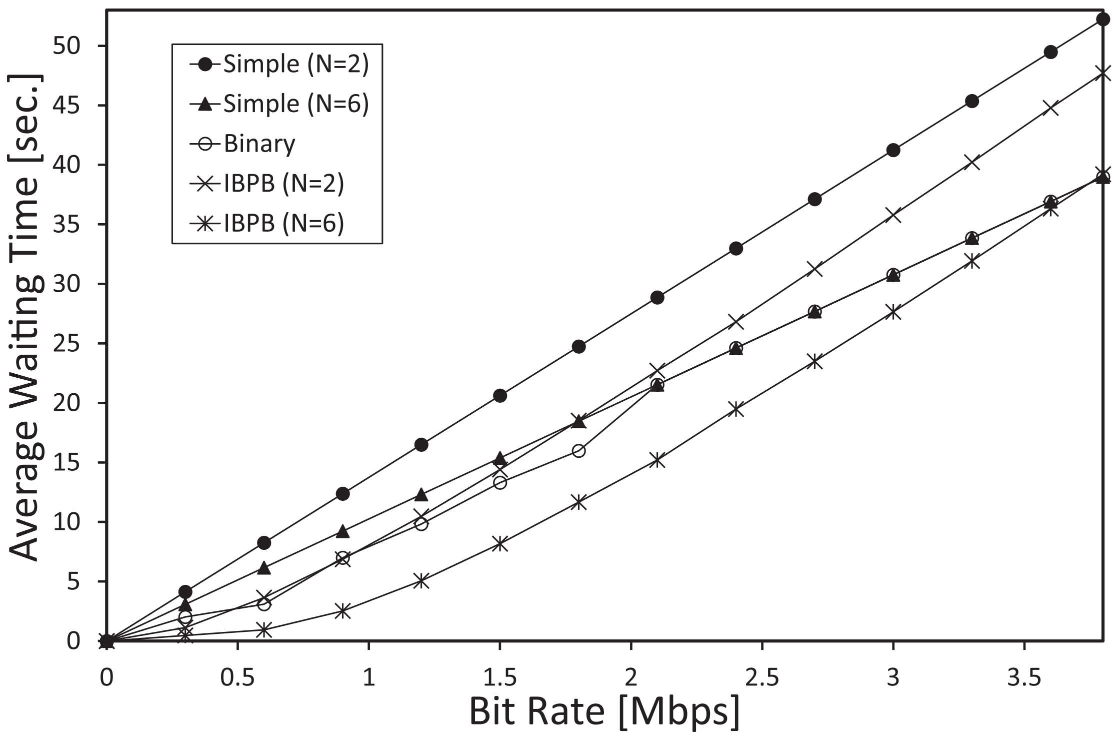


Figure 5.12: Bit rate and average waiting time.

The chunk size in this case was The transmission interval was 7.2 was 1 [ms] [Kbytes] and the ransmission interval . This was because the demodulator's demodulation and packet delivery processes did not arrive in time for reception, and the software running in the demodulator stopped. Therefore, the maximum broadcast bandwidth in the evaluation experiment environment is 3.8 [Mbps]. (2) Delayed delivery due to actual processing The simulation did not take into account delays associated with modulation and demodulation processes, but in a real system, these processes cause delivery delays. Therefore, we measured the delivery delay associated with the actual processing. The chunk size is The delivery delay during a 10-minute broadcast of video data was measured using the 6-split IBPB method with 7.2 [Kbytes] and a transmission interval of 1 [ms] . The results show that the maximum delivery delay is the minimum is 81 [ms], and the average delivery delay is 155 135 [ms], [ms]. The delivery delays associated with the actual processing resulted in an average playback latency of 135 [ms] longer than in the simulation. The difference between the measured and simulated values is not always the same because of fluctuations in the broadcast bandwidth and other factors. Therefore, Figure 5.8 does not appear to be significantly different. Note that since radio waves propagate at the speed of light (about 300 ,000 [km/s]), the longer distance between the transmitting and receiving antennas does not significantly affect the delivery delay. (3) Buffering of video data Although the simulations did not take video data buffering into account, most actual playback software buffers video data to some extent before playback. The time required for buffering affects the playback latency. Therefore, we investigated the time required for buffering. As a result of the investigation, it was found that the implemented wireless broadcast VoD system requires a buffering time of approximately 1 [s]. This buffering time is considered to be the buffering time required when playing which is used as playback software.

5.5.5 Chunk Effects

In the implemented wireless broadcast VoD system, modulation and demodulation are performed in units called chunks. The parameters related to chunks are the chunk size (chunk data size) and the chunk transmission interval. The chunk transmission interval is the time between the completion of transmission of a chunk and the start of transmission of the next chunk. Since a segment consists of chunks, it also corresponds to the transmission interval of a segment (the time between the completion of transmission of the last chunk of a segment and the start of transmission of the first chunk of the next segment to be transmitted). The shorter the interval between transmission of chunks, the shorter the time the broadcast server does not transmit a chunk, and thus the transmission speed (the amount of data broadcast per unit of time) becomes larger. Seg Because a ment is composed of multiple chunks, the segment This results in shorter transmission times. The larger the chunk size, the more data is sent together without any interval, resulting in a higher transmission rate. Thus, the delivery rate varies depending on the chunk size and the transmission interval of the chunks, and the playback latency varies accordingly. In this section, these parameters are varied to investigate the broadcast bandwidth that can be achieved with the implemented system. Because the bandwidth is limited by the processing performance of the demodulator when a demodulator is used, the broadcast bandwidth and playback latency were measured without modulation/demodulation in this experiment. There is no direct relationship between the proposed method and the effect of chunks,

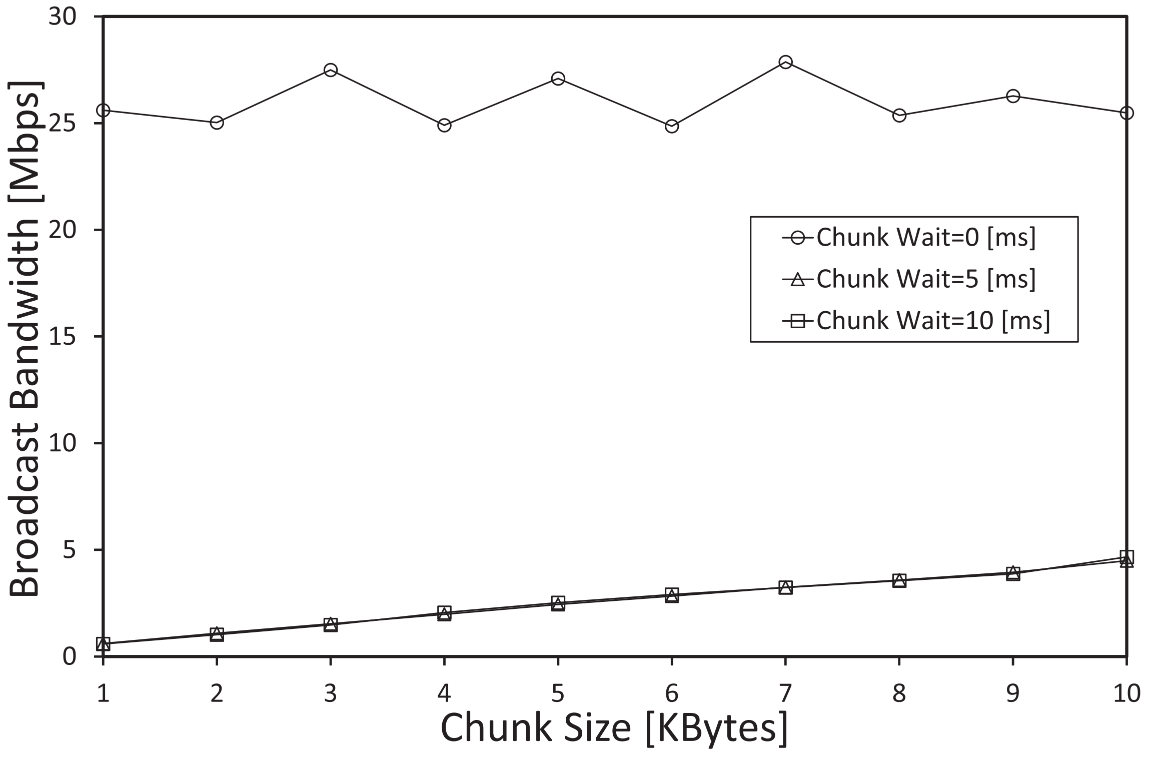


Figure 5.13: Chunk and broadcast bandwidth.

but rather the knowledge of chunk size, chunk transmission interval, and broadcast bandwidth obtained through real-world experiments of the proposed method. Figure 5.13 shows the realized broadcast bandwidth. The horizontal axis is the chunk size and the vertical axis is the broadcast bandwidth. Chunk Wait in the legend refers to the chunk transmission interval. When t h e transmission interval is 0 [ms], the broadcast bandwidth does not change significantly even if the chunk size changes. This is because hunks are transmitted continuously without an interval, and thus the effect of chunk size is less likely to occur. It can also be seen that there is no significant difference in the broadcast bandwidth when the transmission interval is 5 [ms] and 10 [ms]. This may be because the minimum transmission interval of 10 [ms] or more occurs when the chunk interval is set, and the actual transmission interval is 10 [ms] or more even when the interval is set to 5 [ms].

The average replay latency is shown in Figure 5.14. When the transmission interval is 0 [ms] The larger the broadcast bandwidth, the shorter the segment can be broadcast in a shorter time,

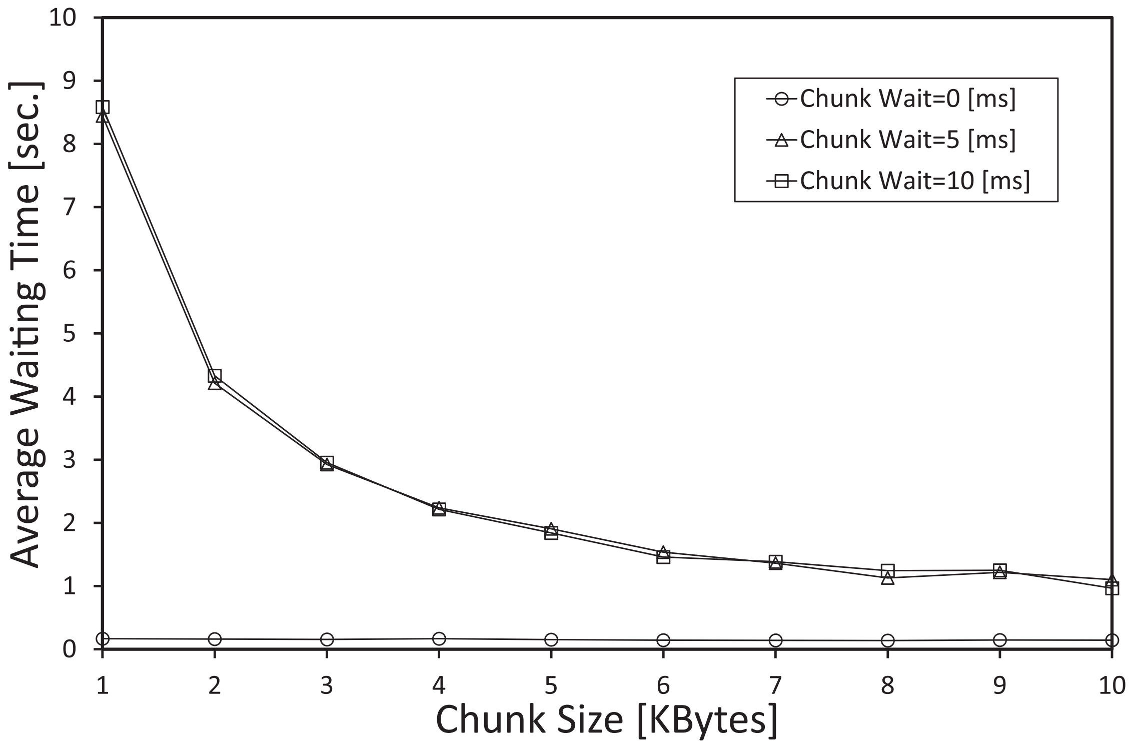


Figure 5.14: Chunk and average waiting time.

resulting in a shorter average playback latency. The fact that the transmission interval did not change significantly between 5 [ms] and 10 [ms] is the same reason why there was no change in the broadcast bandwidth. These results indicate that the larger the chunk size and the shorter the chunk transmission interval, the higher the data delivery rate and the shorter the average playback latency. However, since the extraction rate is limited by the broadcast bandwidth in the evaluation on actual equipment, there is an upper limit to the chunk size and a lower limit to the chunk transmission interval.

For these reasons, the chunk size was set to 7.2 [Kbytes] and the chunk transmission interval to 1 [ms] in the real-world evaluation in this paper. For these reasons, the chunk size was set to 7.2 [Kbytes] and the chunk transmission interval to 1 [ms] for the real-world evaluation in this paper

5.6 Discussion

5.6.1 Number of divisions

In the simulation, the more the video file was divided, the shorter the average playback latency was, but in the real system, it was found that there is a minimum data size that can be divided, and the data cannot be divided into segments smaller than that data size. Since the larger the number of segments, the shorter the average playback latency tends to be, it is considered that the average playback latency is best reduced by dividing the data size of S1 so that the data size of S1 is smaller than the data size of the initial buffering. Therefore, the IBPB method divides the video file so that the data size of S1 is not smaller than the data size of the initial buffering. This creates an upper limit to the number of divisions, which in this experimental environment was limited to 6 divisions

5.6.2 About processing load

In the IPDC used in the evaluation experiments, it was found that delays occur due to processing associated with MPEG encoding, decoding, and modulation/demodulation. The delay time is about 0.1 [s], which is long enough to affect the average playback latency. In addition, it was found that there is an upper limit to the available broadcast bandwidth because the processing load increases as a larger broadcast bandwidth is allocated. Therefore, in a real system, it is necessary to reserve the broadcast bandwidth after considering the performance of the hardware and software.

5.6.3 About playback time and bit rate

In the proposed IBPB method, the longer it takes to replay each segment than it takes to receive it, the lower the average replay latency becomes. The average playback latency can be shortened. The average playback latency can be reduced by splitting a video file with the same bit rate as the broadcast bandwidth, but the average playback latency is the same as that of the simple method, as shown in Figure 5.12. in the case of 6-split video files. In our evaluation experiments, we used video files with bit rates smaller than the bandwidth. This is a common environment in the research field of broadcast video-on-demand [43], [44], [45], [46], [47]. In the evaluation experiment, we assumed a situation in which the average playback latency is 10 [s] as an example of a situation in which playback starts after a short wait even if the video is broadcast without segmentation. In other words, a video file can be broadcast in 10 [s] with a realized broadcast bandwidth of 3.8 [Mbps]. From this, the data size of the video file is assumed to be 3.8 [Mbps] × 10 [s]/8 = 4.75 [Mbytes]. Assuming that short news videos such as earthquake early warnings would be broadcast, the video playback time was set to 60 [s]. Based on the above, the video bit rate was set to 4.75 [Mbps] × 8/60 [s] = 633 [Kbps]. When the video was transcoded with this setting, the video was 650 [Kbps] for 60 [s]. Therefore, the 650 [Kbps] video file of 60 [s] was basically used in the evaluation experiment.

5.7 Conclusion

In this study, we proposed a segment scheduling method for a wireless broadcast-type VoD system. We measured the performance of the implemented wireless broadcast VoD system using the V-High band in terms of playback latency and interruption time on a real system. The performance measurements revealed that, in contrast to previous simulation results, the feasible broadcast bandwidth and delivery delay caused by actual processing and buffering of video data are required. Evaluation experiments revealed that under the condition that playback is uninterrupted, broadcasting video data in 6 segments in parallel can reduce the latency time until playback starts by up to 84%. In the future, playback terminals will also receive video data from the Internet. We are also considering experiments in a broadcastingcommunication convergence environment, and broadcasting-type distribution of interactive content such as selective content and VR broadcasts. We would like to express our sincere thanks to the members of the IPDC Forum for their cooperation in this experiment.

Chapter 6

Conclusion

In future societies, the use of video communication is expected to become increasingly widespread. It is anticipated that broadcasting video logs (vlogs) in real time to a wide audience will become a common practice. Platforms like YouTube enable content creators to deliver live streams and provide on-demand video content. Similarly, viewers widely adopt live streaming and asynchronous viewing of recorded content.

The traditional one-way broadcasting model, where content is delivered identically to all viewers, is transitioning to a two-way communication model. This new model considers individual privacy during video collection and addresses diverse needs in content distribution. This study proposes a real-time facial recognition system designed to protect privacy, particularly during live streaming. The system allows for the identification of registered individuals and masks unauthorized faces, enabling privacy-conscious video distribution.

Furthermore, this research explores systems that optimize video processing and distribution efficiency within a wide-area distributed architecture. The study focuses on two key challenges: first, ensuring privacy protection tailored to individuals; and second, achieving scalability to accommodate varying audience sizes.

To address these challenges, the study proposes a distributed video processing system utilizing Event-Condition-Action (ECA) rules. This approach adjusts video delivery based on viewers' chosen perspectives. Additionally, a novel distribution method combining edge computing and data caching was introduced to resolve playback interruptions for medium-sized audiences. For large-scale audiences, a parallel data distribution method utilizing VHF-band wireless broadcasting was proposed to reduce playback start times.

Based on these findings, this paper discusses specific solutions and practical applications related to facial recognition technologies and VoD systems.

Facial recognition technologies have increasingly been employed in authentication systems, but privacy concerns regarding the misuse of facial images and feature data remain a significant challenge. Traditional quasi-homomorphic encryption methods can ensure privacy but incur high computational costs, rendering them less practical. This study proposes a novel system that enables personal identification in public spaces while preserving privacy by utilizing epsilon noise to obfuscate feature data, balancing security and accuracy. Through a next-generation vlog system, the method demonstrated effective identification for noise levels within 0.2 to 0.6 epsilon, with an optimal balance achieved below 0.4 epsilon. Compared to homomorphic encryption, the proposed method showed superior practicality due to its reduced computational overhead.

To address scalability in video-on-demand (VoD) systems within edge computing environments, this research introduced a pre-caching and redistribution-based approach. The system effectively reduces transmission overlaps and maximizes the number of clients who can stream uninterruptedly. Simulation results indicated a 26% improvement in the shortest transmission interval compared to conventional methods. This study highlights the potential of dynamic caching techniques and workload distribution between edge servers and central distribution servers to optimize future VoD systems.

Further, a multi-viewpoint distributed live Internet broadcasting system was developed to handle the computational load of applying video effects across different perspectives. By implementing Event-Condition-Action (ECA) rules, the system optimally allocated computational tasks to appropriate devices, ensuring efficient video processing. This classification of ECA rules and their integration into edge computing environments shows promise for reducing processing times by leveraging edge devices' shorter turnaround capabilities, paving the way for enhanced real-time broadcasting solutions.

This research proposed a data segment scheduling method for broadcast-type VoD (B-VoD) systems equipped with V-high-band functionality. Experimental results demonstrated that dividing video data into six segments and broadcasting them in parallel reduced playback start times by up to 85% under uninterrupted conditions. Future work will focus on leveraging an integrated broadcast-telecom ecosystem that combines internet and broadcast distribution to deliver interactive content, such as selective streaming and virtual reality broadcasting. Additionally, studies will explore the development of real-time distribution platforms that prioritize seamless content delivery.

Final remarks,this research has contributed significant insights into personal identification in public spaces and next-generation multimedia delivery by developing efficient data distribution systems leveraging privacy-preserving technologies and edge computing. Specifically, it proposed a practical solution for privacy protection in facial recognition by utilizing epsilon noise to preserve privacy while maintaining identification accuracy. Additionally, the effectiveness of edge computing technologies was demonstrated in improving transmission efficiency and reducing processing loads for multi-viewpoint video streaming and VoD systems.

Furthermore, in broadcast-type VoD systems, the segmentation and parallel distribution of video data were shown to significantly reduce playback start times, highlighting the potential for real-time-focused delivery platforms. These findings represent a step forward in the realization of data processing and delivery technologies that integrate security, scalability, and real-time capabilities, laying the foundation for next-generation public systems and media distribution.

Acknowledgments

I would like to express my sincere appreciation to everyone who supported me in various ways throughout my Ph.D. This thesis could not have been accomplished without their assistance. First of all, I express my grate gratitude to my supervisor, Professor Hideyuki Shimonishi of D3 Center, Osaka University, for his insightful suggestions and valuable discussions. He brought me to an attractive research field and made my research life fruitful.

I would like to express my deepest gratitude to Professor Tomoki Yoshihisa, who has guided my research endeavors over the past eight years at the Cybermedia Center, Osaka University, and continued to supervise my doctoral studies as a visiting professor even after his transfer to Shiga University. Professor Tomoki Yoshihisa has exemplified what it means to be an academic and instilled in me the fundamental attitude toward research, making my research life both fulfilling and meaningful.

I would also like to express my deep gratitude to Professor Shinji Shimojo, Professor Yuichi Teranishi, and Associate Professor Tomoya Kawakami for their collaboration in our joint research.

I am heartily grateful to the members of my thesis committee, Professor Masayuki Murata, Professor Hirozumi Yamaguchi, and Professor Takashi Watanabe of Graduate School of Information Science and Technology, Osaka University, and Associate Professor Kenji Ohira of D3 Center, Osaka University, for their multilateral reviews and perceptive comments. Moreover, I am deeply grateful to Associate Professor Yuichi Ohsita, and Assistant Professor Techasarntikul Nattaon of Graduate School of Information Science and Technology, Osaka University.

Finally, I would like to express my heartfelt gratitude to my late father and my family. Thank you for providing me with irreplaceable support throughout my life.

Bibliography

[1] S. Matsumoto, T. Yoshihisa, H. Shimonishi, T. Kawakami, and Y. Teranishi, “Implementation and evaluation of a facial image obscuring method for person identification to protect personal data,” in *2024 IEEE 21st Consumer Communications and Networking Conference (CCNC)*, 2024.

[2] R. N. Zaeem, K. C. Chang, Mittelbach, T.-C. Huang, D. Liau, W. Song, A. Tyagi, M. M. Khalil, M. R. Lamison, S. Pandey, and K. S. Barber, “Blockchain-based self-sovereign-identity: Survey, requirements, use-cases, and comparative study,” in *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT’21)*, 2021.

[3] A. Aleroud, M. Shariah, and R. Malkawi, “Privacy preserving human activity recognition using microaggregated generative deep learning,” in *2nd IEEE International Conference on Cyber Security and Resilience (CSR)*, 2022, pp. 357–363.

[4] S. Takagi, T. Takahashi, Y. Cao, and M. Yoshikawa, “Private high-dimensional data release via privacy preserving phased generative model,” in *P3GM: IEEE ICDE*, 2021, pp. 169–180.

[5] Z. You, S. Li, Z. Qian, and X. Zhang, “Reversible privacy-preserving recognition,” in *IEEE International Conference on Multimedia and Expo (ICME)*, 2021.

[6] C. Dwork, “The promise of differential privacy: A tutorial on algorithmic techniques,” in *IEEE 52nd Annual Symposium on Foundations of Computer Science*, 2011.

[7] Yamanaka.akihiro, kikuchi.ryo, and ikarashi.dai, “Generalization of noises in differential privacy,” *IEICE technical report*, vol. 113, no. 135, pp. 395–402, 2013, (in Japanese).

[8] T. MAEKAWA, yuma KINOSHITA, sayaka SHIOTA, and H. KIYA, “Privacy-preserving svm processing by using random unitary transformation,” *IEICE technical report*, vol. 117, no. 200, pp. 13–18, 2013, (in Japanese).

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” *Journal of Privacy and Confidentiality*, vol. 7, no. 3, pp. 17–51, 2015.

[10] A. Narayanan and V. Shmatikov, “Robust de-anonymization of large sparse datasets,” in *IEEE Symposium on Security and Privacy (S and P)*, 2008, pp. 111–125.

[11] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *ACM Conference on Computer and Communications Security (CCS)*, 2015, pp. 1322–1333.

[12] T. Yang, Y. Zhang, J. Sun, and X. Wang, “Privacy enhanced cloud-based facial recognition,” in *IEEE Symposium on Security and Privacy (S and P)*, 2022.

[13] S. Nakanishi, Y. Narusue, and H. Morikawa, “Response time of cloud-based facial recognition system utilizing homomorphic encryption,” *IEICE Communications Express*, vol. 12, no. 12, pp. 603–606, 2023.

[14] “Insightface: An open source 2d and 3d deep face analysis library,” https://insightface.ai/.

[15] “Youtube faces database,” https://www.cs.tau.ac.il/~wolf/ytfaces/.

[16] Q. Yang *et al.*, “Defending against gan-based image reconstruction attacks on encrypted facial data,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1854–1868, 2020.

[17] S. Matsumoto, Y. Ishi, T. Yoshihisa, T. Kawakami, and Y. Teranishi, “Different worlds broadcasting: A distributed internet live broadcasting system with video and audio effects,” in *Proc. of IEEE International Conference on Advanced Information Networking and Applications (AINA)*, 2017, pp. 71–78.

[18] ——, “A design and implementation of distributed internet live broadcasting systems enhanced by cloud computing services,” in *Proc. of International Workshop on Informatics (IWIN)*, 2017, pp. 111–118.

[19] Y. Gotoh, T. Yoshihisa, H. Taniguchi, and M. Kanazawa, “Brossom: a p2p streaming system for webcast,” *Journal of Networking Technology*, vol. 2, no. 4, pp. 169–181, 2011.

[20] R. Roverso, R. Reale, S. El-Ansary, and S. Haridi, “Smooth-cache 2.0: Cdn-quality adaptive http live streaming on peer-to-peer overlays,” in *Proc. of ACM Multimedia Systems Conference (MMSys)*, 2015, pp. 61–72.

[21] J. Dai, Z. Chang, and G. S. H. Chan, “Delay optimization for multi-source multi-channel overlay live streaming,” in *Proc. of the IEEE International Conference on Communications (ICC)*, 2015, pp. 6959–6964.

[22] T. Yoshihisa and S. Nishio, “A division-based broadcasting method considering channel bandwidths for nvod services,” *IEEE Transactions on Broadcasting*, vol. 59, no. 1, pp. 62–71, 2013.

[23] D. Gibbon and L. Begaja, “Distributed processing for big data video analytics,” *IEEE ComSoc MMTC E-Letter*, vol. 9, no. 3, pp. 29–31, 2014.

[24] W.-C. Ting, K.-H. Lu, C.-W. Lo, S.-H. Chang, and P. C. Liu, “Smart video hosting and processing platform for internet-of-things,” in *Proc. of IEEE International Conference on Internet of Things (iThings)*, 2014, pp. 169–176.

[25] J. Bae, H. Bae, S. Kang, and Y. Kim, “Automatic control of workflow processes using eca rules,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 1010–1023, 2004.

[26] A. Frömmgen, R. Rehner, M. Lehn, and A. Buchmann, “Fossa: Using genetic programming to learn eca rules for adaptive networking applications,” in *IEEE Conference on Local Computer Networks (LCN)*, 2015, pp. 197–200.

[27] M. Yoshida, T. Okuda, Y. Teranishi, K. Harumoto, and S. Shimojyo, “Piax: A p2p platform for integration of multi-overlay and distributed agent mechanisms,” *Transactions of Information Processing Society of Japan*, vol. 49, no. 1, pp. 402–413, 2008.

[28] J. Jeong, H. Kim, B. Kim, and S. Cho, “Wide rear vehicle recognition using a fisheye lens camera image,” in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2016, pp. 691–693.

[29] “Thread building blocks,” https://www.threadingbuildingblocks.org/, (referred October 1, 2017).

[30] S. Matsumoto, Y. Ishi, T. Yoshihisa, T. Kawakami, and Y. Teranishi, “A distributed internet live broadcasting system enhanced by cloud computing services,” *International Journal of Informatics Society (IJIS)*, vol. 10, no. 1, pp. 21–29, 2018.

[31] R. Abuhadra and B. Hamdaoui, “Proactive in-network caching for mobile on-demand video streaming,” in *Proc. IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[32] H. Feng, Z. Chen, H. Liu, and D. Wang, “Optimal cache placement for vod services with wireless multicast and cooperative caching,” in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[33] Y. Gotoh, T. Yoshihisa, M. Kanazawa, and Y. Takahashi, “A broadcasting protocol for selective contents considering available bandwidth,” *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 460–467, 2009.

[34] R. Coutinho, F. Chiariotti, D. Zucchetto, and A. Zanella, “Just-in-time proactive caching for dash video streaming,” in *Proc. Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2018, pp. 1–6.

[35] A. Sheshjavani, B. Akbari, and H. Ghaeini, “An adaptive buffer-map exchange mechanism for pull-based peer-to-peer video-on-demand streaming systems,” *Springer International Journal of Multimedia Tools and Applications*, vol. 76, no. 5, pp. 7535–7561, 2016.

[36] Y. Zhang, C. Gao, Y. Guo, K. Bian, X. Jin, Z. Yang, L. Song, J. Cheng, H. Tuo, and X. Li, "Proactive video push for optimizing bandwidth consumption in hybrid cdn-p2p vod systems," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2018, pp. 2555–2563.

[37] R. Fratini, M. Savi, G. Verticale, and M. Tornatore, "Using replicated video servers for vod traffic offloading in integrated metro/access network," in *Proc. IEEE International Conference on Communications (ICC)*, 2014, pp. 3438–3443.

[38] E. Ghabashneh and S. Rao, "Exploring the interplay between cdn caching and video streaming performance," in *Proc. IEEE Int'l Conference on Computer Communications (INFOCOM)*, 2018, pp. 2555–2563.

[39] S. Yang, Y. Tseng, C. Huang, and W. Lin, "Multi-access edge computing enhanced video streaming: Proof-of-concept implementation and prediction/qoe models," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1888–1902, 2019.

[40] R. Fratini, M. Savi, G. Verticale, and M. Tornatore, "Using replicated video servers for vod traffic offloading in integrated metro/access networks," in *Proc. IEEE International Conference on Communications (ICC)*, 2014, pp. 3438–3443.

[41] G. Zhang, W. Liu, X. Hei, and W. Cheng, "Unreeling xunlei kankan: Understanding hybrid cdn-p2p video-on-demand streaming," *IEEE Transactions on Multimedia*, vol. 17, no. 2, pp. 229–242, 2015.

[42] G. Araniti, P. Scopelliti, G.-M. Muntean, and A. Lera, "A hybrid unicast-multicast network selection for video deliveries in dense heterogeneous network environments," *IEEE Transactions on Broadcasting*, 2018, early Access, 11 pages.

[43] T. Yoshihisa, M. Tsukamoto, and S. Nishio, "A scheduling scheme for continuous media data broadcasting with a single channel," *IEEE Transactions on Broadcasting*, vol. 52, no. 1, pp. 1–10, 2006.

[44] T. Yoshihisa, "Data piece elimination technique for interruption time reduction on hybrid broadcasting environments," in *Proc. IEEE International Conference on Pacific Rim Conference Communications, Computers and Signal Processing (PACRIM)*, 2017, 6 pages.

[45] J. Guo, X. Gong, J. Liang, W. Wang, and X. Que, "An optimized hybrid unicast/multicast adaptive streaming scheme over mbms-enabled wireless networks," *IEEE Transactions on Broadcasting*, 2018, early Access, 12 pages.

[46] C. Tian, J. Sun, W. Wu, and Y. Luo, "Optimal bandwidth allocation for hybrid video-on-demand streaming with a distributed max flow algorithm," *ACM Journal of Computer Networks*, vol. 91, no. C, pp. 483–494, 2015.

[47] F. Boronat, M. Montagud, D. Marfil, and C. Luzon, "Hybrid broadcast/broadband tv synchronization: Demands, preferences and expectations of spanish consumers," *IEEE Transactions on Broadcasting*, vol. 64, no. 1, pp. 52–69, 2018.

[48] F. Boronat, D. Marfil, M. Montagud, and J. Pastor, "Hbbtv-compliant platform for hybrid media delivery and synchronization on single-and multi-device scenarios," *IEEE Transactions on Broadcasting*, vol. 64, no. 3, 2017, 6 pages.

[49] L. Christodoulou, O. Abdul-Hameed, and A.-M. Kondoz, "Toward an lte hybrid unicast broadcast content delivery framework," *IEEE Transactions on Broadcasting*, vol. 63, no. 4, pp. 656–672, 2017.

[50] Y. Hang, C. Gao, Y. Guo, K. Bian, X. Jin, Z. Yang, L. Song, J. Cheng, H. Tuo, and X. M. Li, "Proactive video push for optimizing bandwidth consumption in hybrid cdn-p2p vod systems," in *Proc. IEEE INFOCOM*, 2018, 9 pages.

[51] S. Matsumoto, K. Ohira, and T. Yoshihisa, "A mathematical analysis of 2-tiered hybrid broadcasting environments," in *Proc. International Workshop on Streaming Media Delivery and Management Systems (SMDMS)*, 2019, pp. 454–460.

[52] "Ffmpeg," https://ffmpeg.org.

[53] “Big buck bunny,” https://peach.blender.org.