IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 November 2023, accepted 3 January 2024, date of publication 10 January 2024,
date of current version 19 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3352435

RESEARCH ARTICLE

Pattern Reduction for Low-Traffic Speculative Video Transmission in Cloud Gaming System

TAKUMASA ISHIOKA[1,2], (Member, IEEE), TATSUYA FUKUI[3], TOSHIHITO FUJIWARA[3], SATOSHI NARIKAWA[3], TAKUYA FUJIHASHI[1], (Member, IEEE), SHUNSUKE SARUWATARI[1], (Member, IEEE), AND TAKASHI WATANABE[1], (Member, IEEE)

[1]Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565-0871, Japan
[2]Faculty of Engineering, Department of Information and Computer Science, Kyoto Tachibana University, Kyoto 607-8175, Japan
[3]NTT Access Network Service Systems Laboratories, NTT Corporation, Musashino-shi, Tokyo 180-8585, Japan

Corresponding author: Takumasa Ishioka (ishioka.takumasa@ist.osaka-u.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP19H01101 and Grant JP22H03582; and in part by the NTT Access Network Service Systems Laboratories, Japan.

ABSTRACT Cloud gaming allows users to play high-quality games on low-end devices by offloading game processing to the cloud. However, network latency remains a significant issue affecting the gaming experience. Speculative execution is a promising approach to hide network latency by predicting and transmitting future frames early. However, existing methods generate excessive compute load and network traffic due to many potential input patterns. This paper introduces a pattern reduction method that uses a bit field representation of the input and facilitates efficient speculative execution in cloud games. There are two pattern reduction techniques: analyzing temporal patterns to detect frequent transitions and using LSTM-based predictions to estimate input probabilities. Experiments using actual gaming data show that the proposed methods significantly reduce rendered frames and network traffic versus prior speculative execution methods. The results demonstrate the method's effectiveness and scalability across diverse game genres.

INDEX TERMS Cloud-gaming, speculative execution, low-latency.

I. INTRODUCTION

The spread of smartphones has allowed people to play games anytime and anywhere. Over the past few years, smartphone games have gained a lot of attention [1]. As network technology advances, smartphone games and cloud gaming systems are attracting increasing attention [2]. In a cloud gaming system, the cloud performs game processing based on user operations and sends only the resulting video frames to the user's terminal [3]. This service allows users to play the latest games anywhere as long as they have an Internet connection, even if they do not have a high-spec PC or game console. Additionally, the fact that users do not need to purchase the equipment necessary to play the latest games separately also contributes to an increase in the number of players. However, in cloud gaming systems where players engage in games over a network, the impact of delay emerges as a significant issue. If the response delay from user operation to screen display increases, the game's operational feel will be impaired [4]. Response time is an essential issue for specific games, and requirements vary depending on the game's characteristics in question [5], [6]. Improving the response delay in cloud gaming systems is necessary to enjoy games anytime and anywhere.

Table 1 shows the main research topics related to latency and traffic issues in cloud gaming systems. Some studies have applied platform technologies such as mobile edge computing to reduce response times [7], [8]. Many focused on platform optimization, including dynamic video quality adjustment [9], cloud resource optimization [10], [11], and cloud-native game development. However, no matter how much latency is reduced, the gaming experience in cloud

The associate editor coordinating the review of this manuscript and approving it for publication was Sajid Ali.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

TABLE 1. Major research topics on response time and traffic challenges in cloud gaming.

Research Topics	Description
Low-latency Network Techniques	Developing and applying high-speed, low-latency network technologies (5G/6G) to mitigate latency and improve the cloud gaming experience.
Edge Computing	Processing games on devices closer to the edge of the network to mitigate response time and traffic issues.
Dynamic Video Quality Adjustment	Dynamically adjusting video quality according to bandwidth constraints to mitigate response time and traffic issues.
Data Compression Techniques	Developing and applying efficient compression techniques for images and audio to reduce data transfer volume, thus mitigating response time and traffic issues.
Cloud Resource Allocation Optimization	Optimizing cloud resource allocation and developing algorithms and methods to mitigate response time and traffic issues.
Network Protocol Optimization	Developing and applying new network protocols and techniques to mitigate response time and traffic problems.
Cloud-native Game Development	Developing games are optimized for the cloud environment to address response and traffic issues and improve the gaming experience.
Latency Mitigation using Deep Reinforcement Learning	Optimizing cloud gaming experience using deep reinforcement learning to mitigate response time issues.
Speculative Execution on Cloud Gaming Servers **(A topic of focus in this paper)**	Pre-rendering game frames corresponding to all potential user input patterns and transmitting the frames in advance to mitigate response time issues, ideally achieving zero latency.

gaming systems is different from that in the local game. In order to ensure that a cloud gaming system provides an equivalent gaming experience, the system must be designed to create the illusion for the user that there is no network delay.

On the other hand, [12], [13] proposes speculative execution to reduce response time. In these approaches, the server pre-renders and sends frames corresponding to all potential user input patterns. Because the server sends video frames before user input, speculative execution reduces response time on the user side, ideally reducing the response time to zero. However, two challenges arise when implementing speculative execution in cloud gaming systems. First, cloud gaming servers require enormous computing power. The number of possible input patterns between frames of each game ranges from hundreds to thousands. It increases exponentially as the number of frames that need to be speculatively executed increases. Secondly, this process generates a large amount of video traffic. Even with sufficient computational power for speculative execution, the system must send many generated frames in advance. Consequently, speculative execution in cloud gaming systems leads to a significant increase in traffic.

We are investigating how to solve the problem of speculative execution in cloud gaming systems and improve its feasibility. This research proposes a pattern reduction method that enables speculative execution in cloud gaming systems. By reducing the number of speculative execution patterns, we aim to resolve computing power and traffic issues fundamentally. We developed and proposed a new pattern reduction method for speculative execution based on bit fields to adapt it to various games and devices running on cloud gaming systems. In this system, we save the input signals from each user's device as a bit field input log and analyze the input patterns. We propose and discuss two methods: pattern detection-based analysis and Long Short-Term Memory (LSTM) based pattern analysis. Pattern detection-based analysis analyzes the received input logs over time to obtain the potential transition patterns. The LSTM-based analysis uses LSTM architecture to capture the input log's short-term and long-term temporal patterns. This method enables effective speculative execution in-game environments that require complex input prediction [14]. To verify the effectiveness and scalability of the proposed method, we collected input logs from multiple users playing games of different genres, such as fighting games, action games, and first-person shooter (FPS) games, and based on these input logs, we developed a detailed method. We carried out experimental measurements. The evaluation results revealed that the speculative execution method using our proposed system could reduce the number of rendered game frames and significantly reduce traffic compared to the existing speculative execution methods.

The contributions of this research are as follows.

1) We proposed a bit-field-based pattern reduction method to realize the speculative execution of various games and devices running on cloud gaming systems.
2) To investigate the scalability of the proposed method, we performed experimental measurements using input logs obtained by having multiple users play various games, including fighting games, action games, and FPS games.
3) We clarified the scenarios in which the two proposed analysis methods are most effective and specifically discussed and suggested the types of games and situations to which each method is most suitable.

II. RELATED WORK

Several approaches have been proposed to reduce or hide the effects of response time in cloud gaming systems. We can broadly categorize these into methods for individual

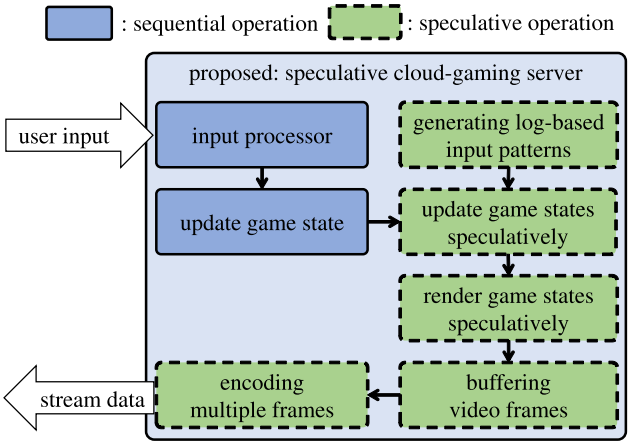


FIGURE 1. Proposed cloud gaming server architecture.

latency reduction, zero-latency solutions through speculative execution, and input prediction methods.

A standard method for reducing individual latencies is to optimize the network architecture specifically for cloud gaming traffic. For individual delay reduction, Suzujevic [15] proposed an adaptive video coding method to reduce transmission and processing delays. Specifically, the cloud gaming server degrades game video frames and their frame rate to reduce the video traffic and corresponding transmission delay. In [7], they aim to reduce propagation delay by decreasing the distance between the cloud gaming server and the user terminal. Specifically, each user connects to a physically nearby cloud gaming server and exchanges packets, thus achieving an experienced quality closer to a local gaming system. Zhang [8] utilized edge networks to reduce network delay and bandwidth consumption. They form edge networks to a data center for cloud gaming and perform computationally demanding operations, such as video rendering, on the edge networks. Other studies [16], [17] adjust the game system to reduce the effect of a network delay on the quality of experience.

Speculative execution has been proposed as a zero-latency method. Outatime [13] and CloudHide [12] transmit predicted game frames to clients in advance based on heuristics. Outatime transmits speculatively generated game frames and correction information early in the response time of the server-user network. Each user outputs a game frame from the received multiple game frames by applying the appropriate image synthesis method according to the user's input. On the contrary, user devices require high computational costs to perform the sophisticated image synthesis method. By transmitting all speculatively generated game frames to the user in advance, CloudHide effectively reduces response time and increases video traffic. The key limitation is the computational overhead required for sophisticated client-side image processing. There is also a tradeoff between latency reduction and additional network bandwidth consumed by sending multiple speculatively generated frames.

Input prediction is another active area of research for hiding latency. GGPO [18] masks network delays by

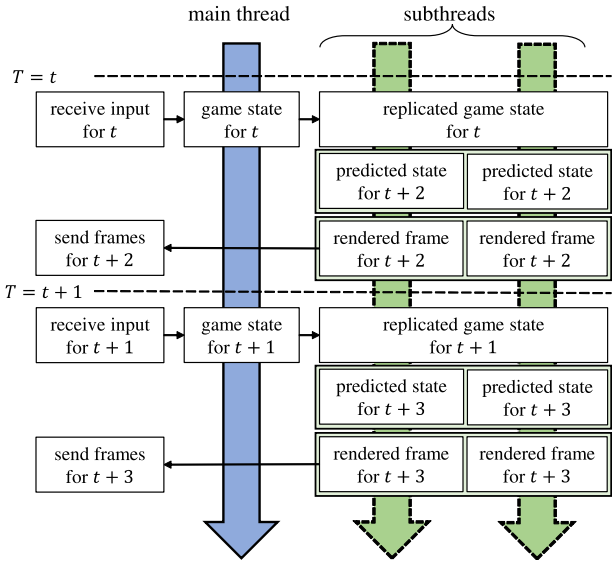


FIGURE 2. Threading process in cloud gaming servers with speculative execution for two future frames. Here, the cloud gaming server has transmitted the user's game frames for $t+1$ at $T=t-1$ and t at $T=t-2$ to the user.

speculatively applying inputs locally, assuming they will match future actual inputs. This approach has proven effective in fast-paced fighting games where experienced players can predict opponents' moves reliably. Extensions apply hidden Markov models [19] and AI agents [20], [21] to improve fighting game input predictions. However, current methods are narrowly focused on one specific genre and may not generalize well to others.

Our proposed methods take inspiration from zero-delay and input prediction solutions. Our methods are novel in introducing heuristic pattern reduction methods and LSTM-based input prediction methods to increase the feasibility of speculative execution. We propose a method that is not limited to specific conditions, accommodating multiple game genres and operating terminals in a cloud gaming system.

III. PROPOSED METHOD

A. OVERVIEW

Fig. 1 shows an overview of the cloud gaming server in the proposed method. First, a user sends inputs over the network to the cloud gaming server. The cloud gaming server collects and analyzes the input logs from the user and generates log-based input patterns for speculative execution. Next, the server updates game states speculatively according to the input the user sends, based on the log-based input patterns generated in advance. Finally, the server renders the game states to video frames and encodes to stream data.

Fig. 2 shows the flow of the speculative execution process on a cloud gaming server. The number of possible input patterns in each frame is assumed to be two, and the server performs speculative execution for two future frames. On the main thread, the cloud gaming server updates the current game state based on the received bit-field data. The cloud

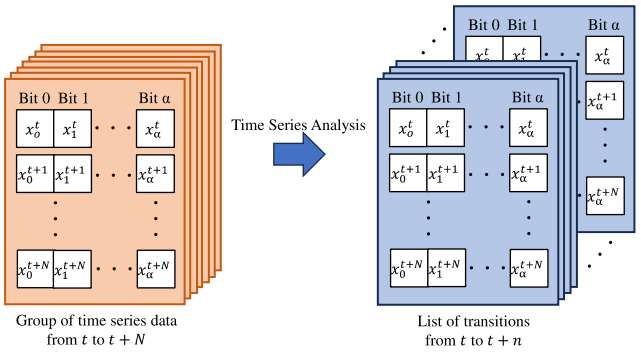


FIGURE 3. Overview of the temporal pattern analysis (TPA).

gaming server updates the current game state on the main thread based on the received bit-field data. The server then replicates the updated game state to multiple subthreads. Each subthread asynchronously updates the game state up to the following two frames. The corresponding game frame is rendered based on the game's final state on each subthread, and the game frame is encoded and transmitted to the user.

B. BIT-FIELD-BASED PATTERN REDUCTION

The cloud gaming system corresponds to various combinations of games and operating devices. Therefore, the pattern reduction methods should not be limited to specific games or devices. Our approach enables pattern reduction regardless of the game or device by treating the input signal independently as bit-field data. More specifically, pattern reduction is performed universally by treating the entire data as time-series bit-field data without establishing a correspondence between a specific input signal and each bit string. This section proposes two pattern reduction methods based on bit-field time-series data: the temporal pattern analysis (TPA) method and the LSTM-based pattern prediction (LBPP) method.

1) TEMPORAL PATTERN ANALYSIS METHOD

Fig. 3 shows the overview of the TPA method. The TPA works by detecting frequent transitions in the input logs over time. More specifically, the server creates a list of patterns through time-series analysis and determines speculative execution patterns from that list. From the input log for period N obtained from the user, the pattern detection procedure obtains and lists the transitions in period n from $t \rightarrow t + n$ and calculates the probability that occurs. Here, x_i^t is a binary value of 0 or 1 that indicates the value of each bit. Therefore, if the bit length is α and the pattern length is n, processing a maximum of $2^{\alpha n}$ patterns is necessary. Each transition is associated with a specific probability, labeled as $p_{0,n}$, $p_{1,n}$, up to $p_{2^{\alpha n},n}$. The proposed system selects a transition from a list if its probability is greater than a threshold and then performs speculative execution. Additionally, the computational space required will be manageable, as we can manage it by analyzing it after extracting a unique bit pattern. Additionally,

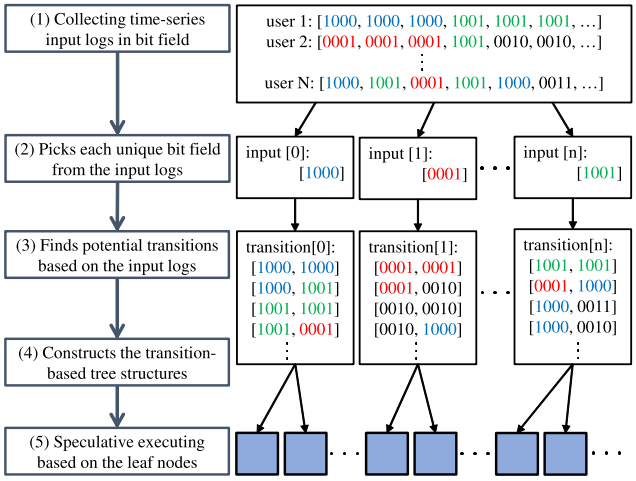


FIGURE 4. Proposed bit-field-based TPA. In this case, the analysis is carried out for two future frames.

by constructing a pattern tree structure in advance, the search speed during actual execution can be completed in a few milliseconds.

Fig. 4 shows the TPA procedure. Here, the length of the bit-field data is assumed to be four. The cloud gaming server analyses a time-series of bit-field data for previous users and uses unique bit-field data listed from all previous users' logs. Redundant input patterns other users have never input can be removed from speculative execution. The cloud gaming server selects each unique bit-field data set and counts the number of transitions from the selected bit-field data to others based on user logs. Redundant transitions that other users have never input can be removed. Note that the cloud gaming server can set the priority of the transitions based on the number of transitions. Finally, the tree structure for each unique bit-field data is constructed based on the above operations, and the bit-field data corresponding to the leaf nodes are used for speculative execution.

Here, we define the average framerate and the number of patterns per frame in the pattern detection procedure. First, if the bit length is α and the pattern length is n, the sum of the probabilities is 1.

$$\sum_{i=0}^{2^{n\alpha}} p_{i,n} = 1 \tag{1}$$

When transitions above the threshold τ are used for speculative execution, the probability of successful prediction is as follows:

$$P_{n,\tau} = \sum_{i=0}^{2^{n\alpha}} p_{i,n}^{[i|0\le i\le 2^{n\alpha}, p_{i,n}\ge\tau]} \tag{2}$$

Therefore, the average framerate is determined as follows:

$$R_{n,\tau} = r \cdot P_{n,\tau} \tag{3}$$

Also, since only transitions that are equal to or greater than the threshold are processed, the number of speculative processing

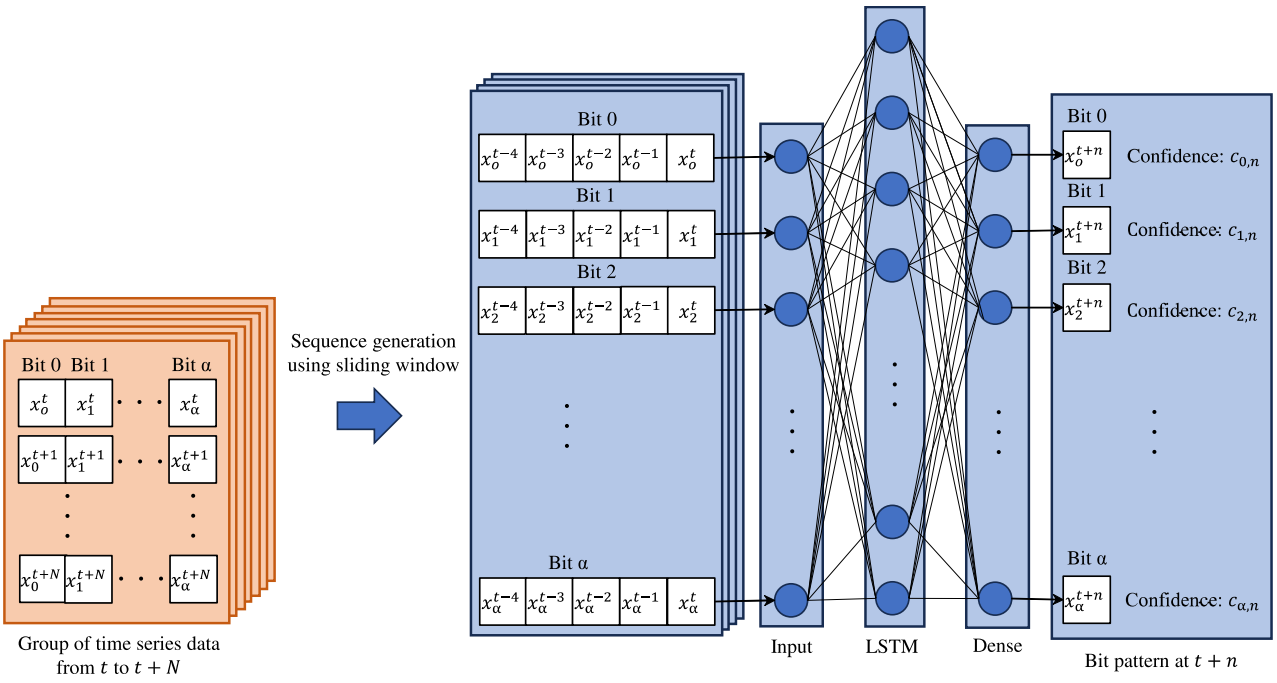


FIGURE 5. The overview of LSTM-based pattern prediction (LBPP).

per frame is determined as follows:

$$F_{n,\tau} = \left|\left\{i|0 \le i \le 2^{n\alpha}, p_{i,n} \ge \tau\right\}\right| \tag{4}$$

2) LSTM-BASED PATTERN PREDICTION METHOD

Fig. 5 shows the overview of the LBPP method. The LBPP uses LSTM neural networks to model short and long-term temporal patterns in the input logs. It then predicts input probabilities to estimate likely future inputs for speculative execution. In the proposed system, we consider generating the most reliable pattern and multiple patterns based on the confidence level of each bit. For example, when building a prediction model for n frames ahead, $N-n+1$ sequences are generated from the log of period N based on a sliding window of period n. Input the generated sequence into the LSTM layer of the neural network. The LSTM layer analyzes the time dependence of the sequence data and inputs the results as features to the Dense layer. Finally, for each bit $\{i|0 \le i \le \alpha\}$, the model outputs the confidence $c_{i,n}$ that the bit will be 0 after n frames. More specifically, bits with confidence greater than or equal to the threshold τ are considered definite, and bits with confidence less than the threshold τ are considered undetermined. This approach improves the prediction n frames ahead and increases the success probability of speculative processing.

The algorithm 1 shows the algorithm of LBPP. LBPP processes the following steps:

1) Initialize data: An empty list named 'data' is initialized to hold the dataset.
2) Read data from files: Data is read from each file and added to the 'data' list after undergoing a transformation process to format it for the LSTM model.
3) Data preprocessing: Input sequences and corresponding labels are prepared from the 'data' list, which is then used to train the LSTM model.

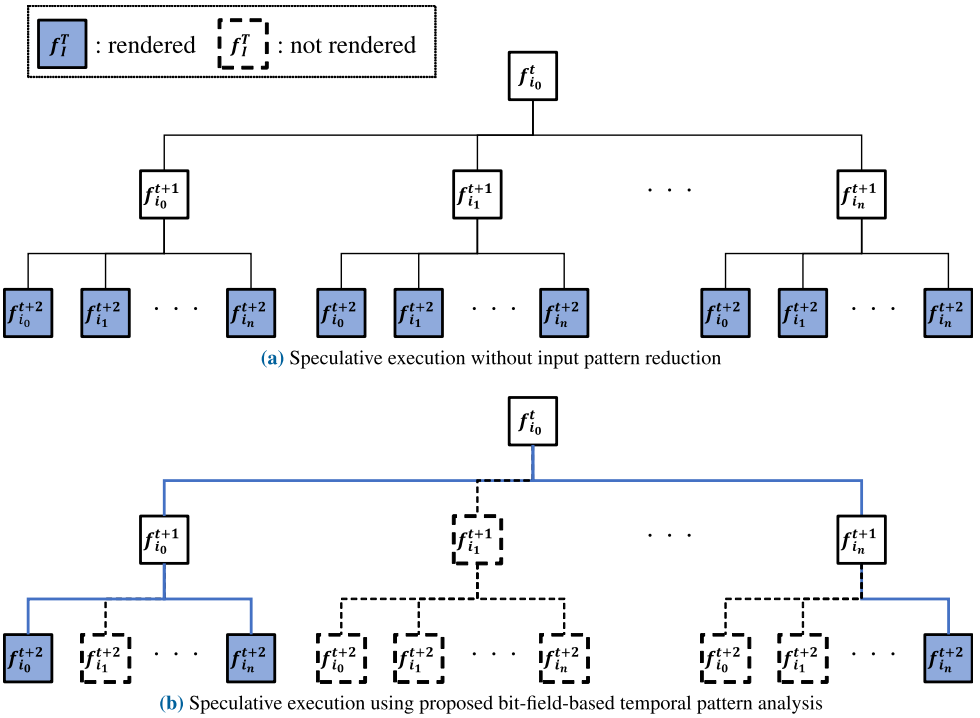
Algorithm 1 The Algorithm of the LBPP

```
Initialize empty list 'data'
for each file do
    Append transformed data from file to 'data'
end for
Prepare training data and labels
Train the LSTM model

Function sliding_window(arr, window_size, start)
return arr[start : start + window_size]

Function evaluate_prediction(model, input_seq, steps)
for each step do
    window = sliding_window(data, seq_length, current)
    Predict the next data point using model and window
    Compute metrics (average probabilities, time taken, etc.)
    Record the metrics
    Update current
end for
return predictions

Function realtime_prediction(model, data_stream, N)
while new data is available from data_stream do
    input_seq = Receive new data point at time t from data_stream
    Initialize empty list for predictions predictions
    for n = 0 to N − 1 do
        window = sliding_window(input_seq, seq_length, t + n)
        prediction = model.predict(window)
        Append prediction to predictions
        Update input_seq with prediction for next step
    end for
    Execute speculative processings based on predictions
end while
```



(a) Speculative execution without input pattern reduction

(b) Speculative execution using proposed bit-field-based temporal pattern analysis

FIGURE 6. The number of game frames rendered and transmitted in the existing and proposed methods. The number of possible input patterns in each frame is *N*, and speculative execution is used for two future frames.

4) Training LSTM model: The LSTM model is trained using the preprocessed data, learning to predict the output labels from the input sequences.
5) Prediction of steps: Two procedures are introduced for evaluation and real-time prediction.
   - Evaluate prediction: This function uses the trained model to predict future data points in a sequence for a given number of steps intended for performance evaluation.
   - Real-time prediction: This function applies the trained model to make real-time predictions for a specified number of future steps.

The confidence that all bits are 0 is represented by the following equation:

$$\prod_{i=0}^{\alpha} c_{i,n} \tag{5}$$

Given a bit number set U, the sum of confidences equals one:

$$\sum_{U \subseteq S} \prod_{i=0}^{\alpha} (c_{i,n}^{[i \in U]} (1 - c_{i,n})^{[i \notin U]}) = 1 \tag{6}$$

Here, $S = \{i | 0 \leq i \leq \alpha\}$. For a bit where $x_i^{t+n} = 0$ and its confidence is greater than $x_i^{t+n} = 1$, the set of bit numbersS_n^0 is determined as follows:

$$S_n^0 = \left\{ i | 0 \leq i \leq \alpha, c_{i,n} \geq (1 - c_{i,n}) \right\} \tag{7}$$

Furthermore, the set of bits for which $x_i^{t+n} = 0$ and the confidence is greater than or equal to ρ is:

$$S_{n,\rho}^0 = \left\{ i | i \in S_n^0, c_{i,n} \geq \rho \right\} \tag{8}$$

Additionally, the set of bits where $x_i^{t+n} = 1$ and the confidence is greater than or equal to ρ is:

$$S_{n,\rho}^1 = \left\{ i | i \notin S_n^0, (1 - c_{i,n}) \geq \rho \right\} \tag{9}$$

The total confidence $C_{n,\rho}$ is the sum of the confidences that are greater than or equal to ρ from the sets described by Eq. (8), (9):

$$C_{n,\rho} = \sum_{i=0}^{\alpha} \left(c_{i,n}^{[i \in S_{n,\rho}^0]} (1 - c_{i,n})^{[i \in S_{n,\rho}^1]} \right) \tag{10}$$

In the proposed system, the average framerate $R_{n,\rho}$ is calculated by multiplying the framerate r of the game executed on the server side by the total confidence $C_{n,\rho}$:

$$R_{n,\rho} = r \cdot C_{n,\rho} \tag{11}$$

Furthermore, the number of undetermined bits, represented as $S'_{n,\rho}$, is the set of bits numbers that are not in either $S_{n,\rho}^0$ or $S_{n,\rho}^1$. Thus, the number of patterns per frame, $F_{n,\alpha}$ is determined as follows:

$$F_{n,\rho} = 2^{|S'_{n,\rho}|} \tag{12}$$

C. SPECULATIVE RENDERING

Fig. 6(a) shows an example of the existing speculative execution for two future frames. Here, the number of input patterns at each frame is considered N, and f_i^t represents the game frame at time t corresponding to input i. In this case, the existing method must render and transmit N^2 game frames for speculative execution. Fig. 6(b) illustrates speculative execution using the proposed bit-field-based temporal pattern

TABLE 2. PC specifications.

	configuration
OS	Windows 11
CPU	Intel Core i9-10850K
RAM	128 GB
GPU	NVIDIA GeForce RTX 3080

analysis. In this scenario, the cloud gaming server does not find input i_1 and transitions of $i_0 \rightarrow i_0 \rightarrow i_1$, $i_0 \rightarrow i_2 \rightarrow i_0$, $i_0 \rightarrow i_2 \rightarrow i_1$ in the logs of past users. Consequently, the cloud game server will not render the game frames corresponding to the input and transitions.

IV. EVALUATION

A. EXPERIMENTAL MEASUREMENTS

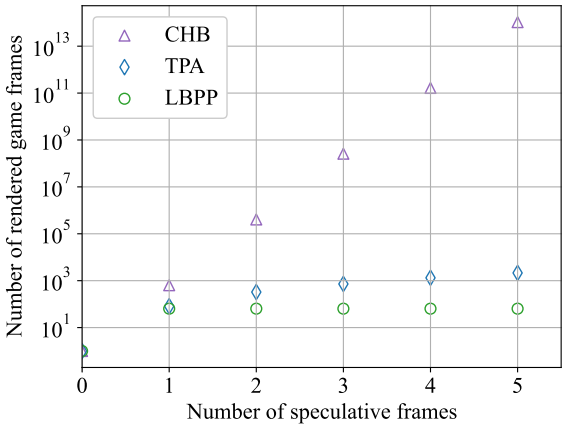
We performed experimental measurements on users' input logs to empirically evaluate the effect of the proposed method. Ten male subjects between the ages of 20 and 30 participated in the experiment. Each subject played the following games using an Xbox One, Microsoft Corp., Redmond, WA, US, controller on a PC for approximately 11 min after a 5-minute tutorial, and the PC collected all inputs. Table 2 shows the PC specifications. Each subject played the following three games sequentially to discuss the effect of game category on performance: *Street Fighter V* (SFV), Capcom Co. and Dimps Corp., Osaka, Japan, a fighting game; *Grand Theft Auto V* (GTAV), Rockstar Games, New York, NY, US, an action game; and *Overwatch 2* (OW2), Blizzard Entertainment, Irvine, CA, US, a first-person shooter game. Specifically, we used the training modes in SFV, the early stages of the story mode in GTAV, and the training mode in OW2 to collect input logs for each subject. In this evaluation, all games ran at 1080p resolution and 60 frames per second. The user's input logs are collected 60 times per second according to the game's framerate.

We implemented the proposed bit-field-based temporal pattern analysis using Python 3.8. We used each subject's 10-minute input log to construct the tree structures. The remaining 1-minute input logs of each subject were used for evaluation.

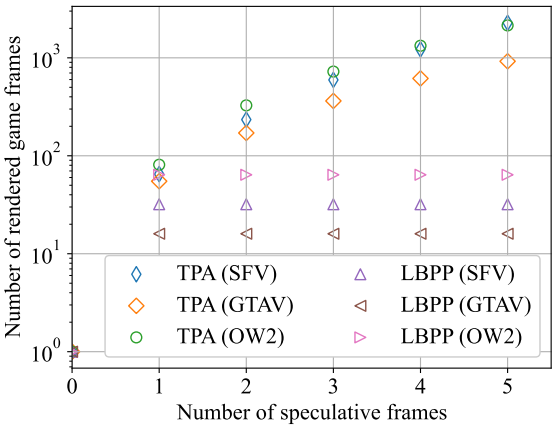
B. EFFECT ON THE NUMBER OF GAME FRAMES

We first verify the baseline performance of the proposed method. We compared the CloudHide-based (CHB) method and the two proposed methods. Here, the CHB renders and sends game frames for every possible input pattern in game processing. In other words, this is where steps up to step (2) in Fig. 4 have been completed.

Fig. 7(a) shows the number of patterns relative to the number of speculative frames in OW2. The average framerate is over 57 fps for all methods. As a result, it was confirmed that both proposed methods suppressed the exponential increase in the number of speculative processes for multiple frames. In particular, LBPP remains almost constant even if the number of frames increases.



(a) Effect in OW2



(b) Effect in all games

FIGURE 7. The number of rendered game frames for the number of speculative frames. The average framerate is set to over 57 fps for all methods in this graph. (a) shows a comparison of the effects of CHB and the proposed method on OW2. (b) shows a comparison of the effects of the proposed method for all games.

Fig. 7(b) shows the number of patterns relative to the number of speculatively processed frames when using the proposed methods in the three games used for evaluation. Both methods maintain an average framerate of over 57 fps in all games. As a result, we found this tendency to be consistent across all games. The number of patterns itself differs depending on the game title. For example, as an FPS game, OW2 may require numerous potential input patterns and transitions due to the need for rapid 3D movement and action. Although the number of input patterns per frame varies depending on the game, input logs can be treated as simple time-series data for any game genre. It was confirmed that the analysis method does not depend on game titles.

The main results can be summarized as follows:

- The proposed methods (TPA and LBPP) reduced speculative execution patterns versus the CHB.
- The number of patterns remained almost constant as the frames increased in the LBPP.
- The effect of analysis methods generalizes and does not depend on the game title.

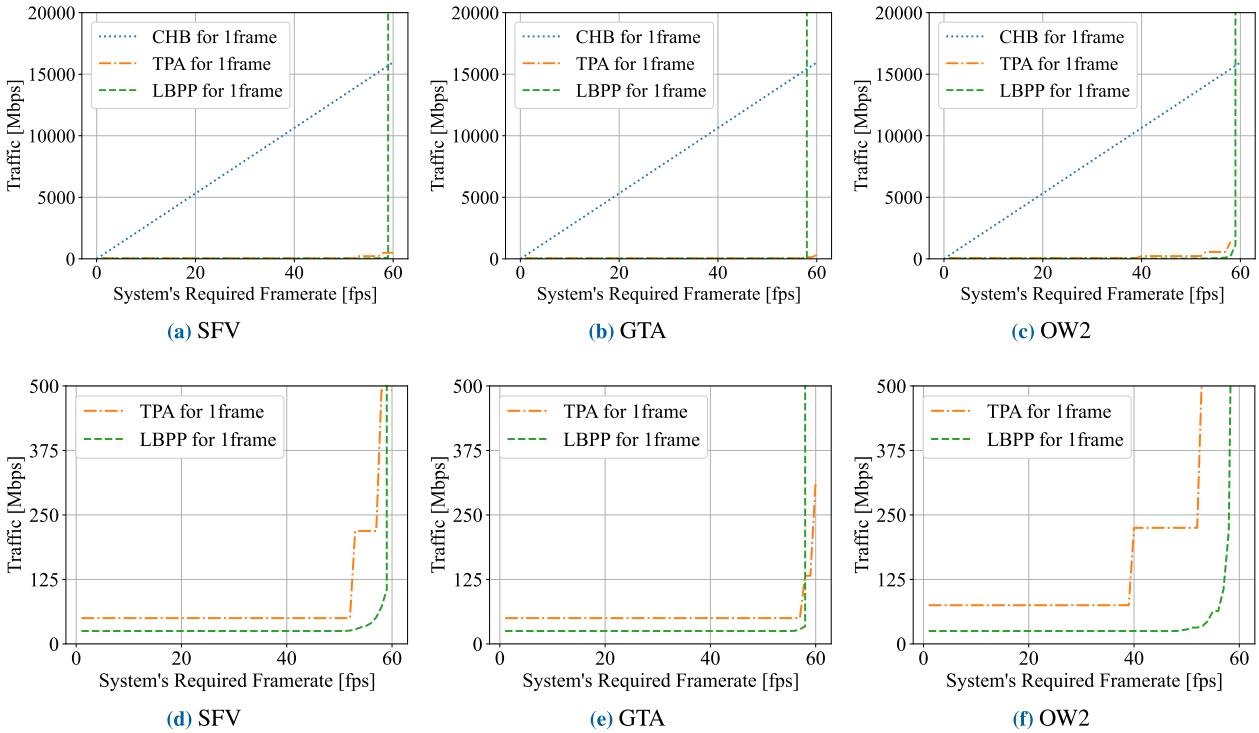


FIGURE 8. The traffic for achieving the system's required framerate. (a)–(c) show a comparison between CHB and the two proposed methods, and (d)–(f) show a comparison of the two proposed methods. In CBH, we change the operating framerate on the server side. In the proposed method, we set a threshold that minimizes the amount of traffic while achieving the system's required framerate.

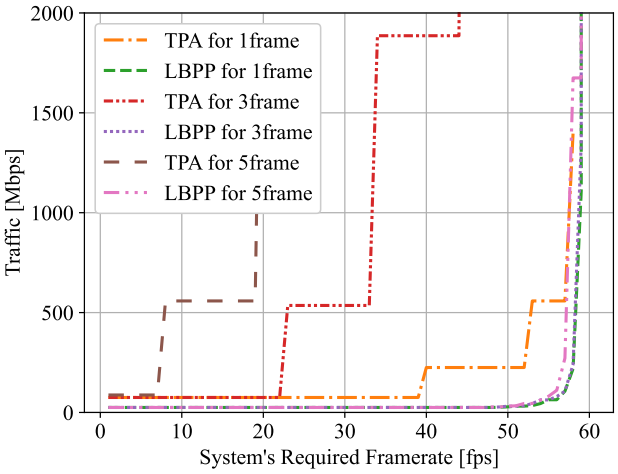


FIGURE 9. A comparison of the proposed methods with 1, 3, and 5 frames speculatively executed in OW2.

C. EFFECT OF PATTERN REDUCTION METHODS

Fig. 8(a)–(f) shows the traffic as a function of the system's required framerate of 1 frame speculative execution. The game on the server runs at up to 60 fps. The CHB generates all possible frames, so adjusting the operating framerate on the server side achieves the required framerate of the system. On the other hand, in the proposed method, the operating framerate on the server side is fixed at 60 fps. In this evaluation, we set a threshold that minimizes the amount of traffic while achieving the system's required framerate.

Fig. 8(a)–(c) shows a comparison between the cloud hide-based method and the two proposed methods. Note that CHB assumes all frames that can be displayed are sent, so the amount of traffic changes linearly with the required framerate. As a result, we found that in any game, when the required framerate is around 57 fps, the proposed method can achieve a very high traffic reduction effect. On the other hand, when the proposed method increases the required framerate above 58 fps and approaches the game operation framerate on the server side, the traffic of the proposed method increases rapidly. Especially for LBPP, the traffic was larger than CHB for all games. LBPP uses a recurrent neural network, and the reliability of each bit is never 100 %. In order to achieve the operating framerate of a server-side game in speculative execution, it is necessary to send all frames that have even the slightest possibility of being displayed. Since it is necessary to consider the bit length of the input signal in LBPP, it is necessary to assume a tremendous amount of traffic.

Fig. 8(d)-(f) shows the results in the range of 500 Mbps or less of the proposed method in Fig. 8(a)-(c), respectively. In Fig. 8(e), the TPA is approximately 300 Mbps, and the game's framerate on the server side is 60 fps. Depending on the genre of the game, if the number of input patterns that can occur is limited, it is possible to achieve a high framerate with a small amount of traffic by using TPA. On the other hand, in Fig. 8(d), (f), the traffic amount at the same framerate was always smaller with LBPP than with TPA. LBPP can achieve stable framerates and traffic reduction regardless of game genre.

Fig. 9 shows a comparison of the proposed methods with 1, 3, and 5 frames speculatively executed in OW2, respectively. As a result, it was confirmed that LBPP is relatively unaffected by an increase in the number of speculatively processed frames. It was confirmed that a certain level of prediction accuracy was maintained with almost no pattern increase, as shown in Fig. 7. On the other hand, TPA increased the amount of traffic to achieve the system's required framerate. TPA performs speculative execution on detected pattern transitions, so sudden actions between each frame damage the framerate, especially in FPS games like OW2.

The main results can be summarized as follows:

- Proposed methods achieved high traffic reduction versus CHB when the target framerate was around 57 fps.
- LBPP is one of the machine learning methods based on RNN, so reliability per bit is never 100 %, requiring more frames to be sent as the target rate approaches 60 fps server rate.
- TPA performed well for some genres, but LBPP achieved stable framerates and traffic reduction across game genres. Moreover, LBPP was relatively unaffected by the increased number of speculative frames.

V. CONCLUSION

This paper proposed pattern reduction methods using bit field representations to enable efficient speculative execution in cloud gaming systems. Temporal pattern analysis and LSTM-based prediction were introduced to analyze input logs and generate likely input patterns for speculative rendering. Experimental results using gaming data from multiple users and genres showed the proposed methods substantially reduced rendered frames and network traffic compared to prior speculative execution methods. Key findings include:

1) The proposed pattern reduction methods suppressed exponential growth in speculative patterns over multiple frames.
2) LBPP performed stable framerates and traffic reduction across game genres.
3) TPA performed well in specific genres and can achieve the system's framerates equivalent to the server-side framerate to a limited extent.

The method demonstrated scalability across diverse games without dependence on specific titles or input devices. Future work includes optimizing the prediction methods and evaluating perceived latency improvements for players. In order to optimize prediction methods, we plan to improve prediction performance and propose a dynamic selection mechanism for multiple prediction methods that take into account performance and execution time. The pattern reduction approach enables the practical deployment of speculative execution to hide network delays and improve the cloud gaming experience.

REFERENCES

[1] Newzoo. (May 2023). *Newzoo's Video Games Market Size Estimates and Forecasts for 2022*. Accessed: Oct. 24, 2023. [Online]. Available: https://newzoo.com/resources/blog/the-latest-games-market-size-estimates-and-forecasts

[2] Newzoo. (Oct. 2022). *Cloud Gaming Revenues to Hit $2.4 Billion in 2022, Up +74% Year-on-Year; Revenues Will Triple by 2025*. Accessed: Oct. 24, 2023. [Online]. Available: https://newzoo.com/resources/blog/cloud-gaming-revenues-to-hit-2-4-billion-in-2022-up-74-year-on-year-revenues-will-triple-by-2025

[3] R. Shea, J. Liu, E. C.-H. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Netw.*, vol. 27, no. 4, pp. 16–21, Jul. 2013.

[4] S. Flinck Lindström, M. Wetterberg, and N. Carlsson, "Cloud gaming: A QoE study of fast-paced single-player and multiplayer gaming," in *Proc. IEEE/ACM 13th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2020, pp. 34–45.

[5] S. S. Sabet, S. Schmidt, S. Zadtootaghaj, C. Griwodz, and S. Moller, "Towards the impact of gamers strategy and user inputs on the delay sensitivity of cloud games," in *Proc. 12th Int. Conf. Quality Multimedia Exper. (QoMEX)*, May 2020, pp. 1–3.

[6] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[7] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proc. 11th Annu. Workshop Netw. Syst. Support Games (NetGames)*, Nov. 2012, pp. 1–6.

[8] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 178–183, Aug. 2019.

[9] A. Alhilal, T. Braud, B. Han, and P. Hui, "Nebula: Reliable low-latency video transmission for mobile cloud gaming," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 3407–3417.

[10] I. Slivar, L. Skorin-Kapov, and M. Suznjevic, "QoE-aware resource allocation for multiple cloud gaming users sharing a bottleneck link," in *Proc. 22nd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2019, pp. 118–123.

[11] L. De Giovanni, D. Gadia, P. Giaccone, D. Maggiorini, C. E. Palazzi, L. A. Ripamonti, and G. Sviridov, "Revamping cloud gaming with distributed engines," *IEEE Internet Comput.*, vol. 26, no. 6, pp. 88–95, Nov. 2022.

[12] B. Anand and P. Wenren, "CloudHide: Towards latency hiding techniques for thin-client cloud gaming," in *Proc. Thematic Workshops ACM Multimedia*, Oct. 2017, pp. 144–152.

[13] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services*, May 2015, pp. 151–165.

[14] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 95–104.

[15] M. Suznjevic, I. Slivar, and L. Skorin-Kapov, "Analysis and QoE evaluation of cloud gaming service adaptation under different network conditions: The case of NVIDIA GeForce NOW," in *Proc. 8th Int. Conf. Quality Multimedia Exper. (QoMEX)*, Jun. 2016, pp. 1–6.

[16] S. S. Sabet, S. Schmidt, S. Zadtootaghaj, B. Naderi, C. Griwodz, and S. Möller, "A latency compensation technique based on game characteristics to mitigate the influence of delay on cloud gaming quality of experience," in *Proc. 11th ACM Multimedia Syst. Conf.*, May 2020, pp. 15–25.

[17] R. Salay and M. Claypool, "A comparison of automatic versus manual world alteration for network game latency compensation," in *Proc. Extended Abstr. Annu. Symp. Comput.-Human Interact. Play*, Nov. 2020, pp. 355–359.

[18] T. P. Cannon. (2015). GGPO Rollback Networking SDK. MIT License. [Online]. Available: https://www.ggpo.net

[19] G. L. Zuin and Y. P. A. Macedo, "Attempting to discover infinite combos in fighting games using hidden Markov models," in *Proc. 14th Brazilian Symp. Comput. Games Digit. Entertainment (SBGames)*, Nov. 2015, pp. 80–88.

[20] K. Asayama, K. Moriyama, K.-I. Fukui, and M. Numao, "Prediction as faster perception in a real-time fighting video game," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2015, pp. 517–522.
[21] K. Yu and N. R. Sturtevant, "Application of retrograde analysis on fighting games," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–8.

TAKUMASA ISHIOKA (Member, IEEE) received the B.E. and M.E. degrees from Osaka University, Japan, in 2019 and 2021, respectively. He has been an Assistant Professor with the Faculty of Engineering, Kyoto Tachibana University, Japan, since April 2023. He is a member of IPSJ. His research interest includes wireless networks.

TATSUYA FUKUI received the B.E. and M.E. degrees from the Faculty of Science and Engineering, Waseda University, Japan, in 2008 and 2010, respectively. He is currently with NTT Access Network Service Systems Laboratories. His current research interests include the research and development of carrier networks, such as wide-area Ethernet systems.

TOSHIHITO FUJIWARA received the B.E., M.E., and Ph.D. degrees in engineering from the University of Tsukuba, Ibaraki, Japan, in 2002, 2004, and 2011, respectively. In 2004, he joined the NTT Access Network Service Systems Laboratories, Japan, where he has been involved in the research and development of optical video transmission system, passive optical network system, content delivery network system, and ultralow latency video system.

SATOSHI NARIKAWA received the B.E., M.E., and Ph.D. degrees from the Department of Electrical and Electronics Engineering, Tokyo Institute of Technology, Japan, in 2001, 2003, and 2012, respectively. He is currently with NTT Access Network Service Systems Laboratories. His current research interests include the research and development of optical access systems.

TAKUYA FUJIHASHI (Member, IEEE) received the B.E. and M.S. degrees from Shizuoka University, Japan, in 2012 and 2013, respectively, and the Ph.D. degree from the Graduate School of Information Science and Technology, Osaka University, Japan, in 2016. He has been an Assistant Professor with the Graduate School of Information Science and Technology, Osaka University, since April, 2019. He was a Research Fellow (PD) of Japan Society for the Promotion of Science, in 2016. From 2014 to 2016, he was a Research Fellow (DC1) of Japan Society for the Promotion of Science. From 2014 to 2015, he was an Intern with Mitsubishi Electric Research Laboratories (MERL), Electronics and Communications Group. His research interests include the area of video compression and communications, with a focus on immersive video coding and streaming.

SHUNSUKE SARUWATARI (Member, IEEE) received the Dr.Sci. degree from The University of Tokyo, in 2007. From 2007 to 2008, he was a Visiting Researcher with the Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana–Champaign. From 2008 to 2012, he was a Research Associate with the RCAST, The University of Tokyo. From 2012 to 2016, he was an Assistant Professor with Shizuoka University. He has been an Associate Professor with Osaka University, since 2016. His research interests include the areas of wireless networks, sensor networks, and system software.

TAKASHI WATANABE (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from Osaka University, Japan, in 1982, 1984, and 1987, respectively. He is currently a Professor with the Graduate School of Information Science and Technology, Osaka University. He joined the Faculty of Engineering, Tokushima University, as an Assistant Professor, in 1987, and moved to the Faculty of Engineering, Shizuoka University, in 1990. He was a Visiting Researcher with the University of California at Irvine, Irvine, from 1995 to 1996. He has served on many program committees for networking conferences, IEEE, ACM, IPSJ, and The Institute of Electronics, Information and Communication Engineers (IEICE), Japan. His research interests include mobile networking, ad hoc networks, sensornetworks, ubiquitous networks, intelligent transport systems, specially MAC, and routing. He is a member of IEEE Communications Society, IEEE Computer Society, IPSJ, and IEICE.