https://doi.org/10.7840/kics.2024.49.8.1110

Hierarchical Reinforcement Learning-Based Delay-Sensitive Video Delivery in Vehicular Networks

Yunoh Kim◆, Tiange Xiang*, Yeongjin Kim**, Minseok Choi°

ABSTRACT

This paper jointly optimizes node scheduling and the delivery of video chunks in delay-sensitive dynamic video streaming using a hierarchical reinforcement learning algorithm. Specifically, we presented an algorithm capable of adjusting node scheduling and the transmission of video chunks at two different slow and fast timescales, respectively. When nodes caching content are randomly distributed, mobile users dynamically select the node from which to receive video and control the number and quality of video chunks from the selected node, depending on the channel conditions with nearby nodes, the quality of cached content, and the user's queue status.

Key Words : Hierarchical Reinforcement Learning, Vehicular Network, Video Delivery, Scheduling

Ⅰ. Introduction

Online video services account for the largest portion of total data traffic, with the number of dynamic video streaming users rapidly increasing[1]. Furthermore, the proliferation of smart devices has recently increased the demand for video services among vehicle drivers and passengers[2]. Therefore, there is a need for an efficient method of providing video streaming services in vehicular networks.

In vehicular networks, frequent changes in the communication environment occur due to user mobility. Hence, a rational method capable of dynamically operating according to the environment is necessary. To this end, this paper aims to use reinforcement learning algorithms that can adapt to the environment through learning. While studies on video delivery[3] and scheduling[4] using reinforcement learning exist, they treat these two processes separately. Moreover, since the user's queuing system information was not used in scheduling, research that reflects the user's queuing system information in scheduling to reduce the actual buffering rate is needed. Similarly, although studies exist on transmitting contents encoded with Scalable Video Coding (SVC) in vehicular networks[5], they do not consider the caching network and the user queuing system. Conversely, studies on transmitting contents in caching networks[6,7] often focus on network latency and cache hits, rather than service quality based on the user's queuing system.

This paper aims to improve the quality of video streaming services in vehicular networks with distributed caching nodes by reflecting the user's queuing system information. Our hierarchical reinforce-

※ This work was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02201, Federated Learning for Privacy-Preserving Video Caching Networks), in part by the National Research Foundation of Korea(NRF) grant funded by the Korea govenment(MSIT) (NRF-2022R1C1C1010766), and in part by NRF grant funded by MSIT (No. 2022R1A4A3033401).
◆ First Author : Department of Electronics and Information Convergence Engineering, Kyung Hee University, rladbsh456@naver.com, 학생회원
° Corresponding Author : Department of Electronic Engineering, Kyung Hee University, choims@khu.ac.kr, 정회원
* Department of Electronics and Information Convergence Engineering, Kyung Hee University, rladbsh456@naver.com, 학생회원
** Department of Electronic Engineering, Inha University, yj.kim@inha.ac.kr, 정회원
논문번호 : 202403-042-0-SE, Received February 21, 2024; Revised April 4, 2024; Accepted April 9, 2024

ment learning-based link scheduling and video chunk delivery scheme can be applied to any delay-sensitive services for highly dynamic networks. For example, the proposed scheme can be applied to video streaming in unmanned aerial vehicle and/or satellite networks where both video source and user can be mobile. The contributions of this research are as follows:

- We propose a scalable content delivery and scheduling method for various communication environments in vehicular networks with distributed caching nodes using reinforcement learning.
- We solve content delivery and scheduling at a double time scale through hierarchical reinforcement learning, modeling both tasks to be optimized towards a single objective.
- We developed an algorithm that increases bitrate and reduces buffering rate by reflecting the user's queuing system information without separate channel estimation.

Ⅱ. System Model

2.1 Communication Scenario

In this paper, we consider a scenario where caching nodes are distributed in the vehicular network according to a Poisson Point Process (PPP) and provide video services to mobile users. We focus on situations where users request delay-sensitive video services from caching nodes, which cache video content with the specific quality level. Users can only receive content from caching nodes within a communication radius R. Here, we assume that all contents are encoded using Scalable Video Coding (SVC), meaning each video stream is partitioned into multiple chunks, each comprising L layers. For video playback to be possible, at least a single basement layer of each chunk must be transmitted to the user, with the delivery of additional enhancement layers enabling higher quality viewing. Video delivery and scheduling are two critical tasks for providing content to users. We denote the set of caching nodes within the radius R of the user at slot t as $N(t)$. Users select a single caching node $\alpha(t)$ from $N(t)$, the number of receiving chunks $M(t)$, and the number of receiving layers $K(t)$. We assume that the numbers of layers for all the received chunks at slot t are identical.

When time is discretized into slots t ∈ {0, 1, 2, ⋯}, the user selects a caching node $\alpha(t)$. Each caching node probabilistically caches l layers of the desired content with a probability of p_l. In this case, the node that caches l layers can provide the quality to the user only up to $K(t) \in \{1, 2, \cdots, l\}$. To enhance the quality of receiving chunks, i.e., $K(t)$, scheduling with a node that caches higher quality content is necessary. However, even if a node caches high-quality content, scheduling with a node far from the user might fail to deliver high-quality content due to insufficient channel capacity $C(t)$. This research presents the appropriate scheduling method based on the cache status of nearby nodes and channel conditions from the user using the Deep Q-Network (DQN) algorithm, without the knowledge of $C(t)$.

After scheduling the node, the desired content becomes delivered via the wireless channel, and we have to determine the number of receiving chunks and their quality level, denoted by $M(t)$ and $K(t)$, respectively. Assuming a Rayleigh fading channel, the channel capacity is given by

$$C(t) = W \ \log_2(1 + \psi|h(t)^2| \ / \ (\gamma + 1)\ ) \quad (1)$$

where $h(t) = \sqrt{w(t)/d^{\beta}}u(t)$ is channel gain, $u(t)$~$CN(0,1)$ is fast fading gain, $w(t)$~$N(0,1)$ is shadowing effect, d is the distance between the scheduled node and the content-requesting user, β is the path loss coefficient, γ is interference-to-noise ratio (INR), W is bandwidth, Ψ is transmit SNR. When t_c represents the playtime of a single chunk, $M(t)$ and $K(t)$ should satisfy the following inequality due to the limited channel capacity:

$$M(t) \cdot \ K(t) \ \leq \ t_c C(t) \quad (2)$$

Therefore, to enjoy content at high quality, $K(t)$ must be increased, but according to (2), increasing $K(t)$ reduces $M(t)$. The transmitted chunks accumulate in the user queue, waiting to be played, with one chunk played at each time t. When the queue length

is $Q(t)$ at slot t, the dynamics equation of the queuing system is as follows:

$$Q(t+1) = max(Q(t) - 1,0) + M(t) \quad (3)$$

If $Q(t) = 0$, there is no chunk to play, resulting in buffering. Therefore, to ensure user satisfaction, both lowering buffering occurrences and higher bitrates should be provided, which are in a trade-off relationship according to equation (2). This study proposes a content delivery method using DQN, which can learn to appropriately determine $M(t)$ and $K(t)$ and control their trade-off without estimating channel conditions.

Content delivery and scheduling operate on different timescales. Scheduling is conducted on a large timescale, updating scheduling in large time units T_c, while the delivery of video chunks occurs on a small timescale, repeating in small time units t. Additionally, content delivery decisions $M(t)$ and $K(t)$ are dependent on the scheduled caching node $\alpha(t)$. Therefore, it is necessary to implement both tasks to operate on different timescales but jointly learn the policy for these tasks. In this study, the combination of these two different decision timescales and tasks has been implemented through hierarchical reinforcement learning with two DQN algorithms.

2.2 Design of Markov Decision Process

To implement reinforcement learning, it is necessary to define a Markov decision process (MDP). An MDP defines the information the learning agent receives as input as the state, the decisions it makes as actions, and the feedback it receives from the environment as rewards. The reinforcement learning algorithm learns to maximize the accumulated rewards during the episode based on the state inputs to determine the optimal actions.

In our content delivery scenario, the user needs to learn to receive content chunks at a higher quality while reducing buffering when transmitting content. Therefore, to prevent situations where $Q(t) = 0$, which leads to buffering, it is essential to know the user's queuing system information $Q(t)$. Additionally, this study has been designed to reduce frequent quality fluctuations while the user is viewing content, ensuring a comfortable viewing experience by also knowing the quality of the last chunk fully received in the user queue. Thus, the state includes the queue length $Q(t)$ and the quality of the last received chunk. The agent receives this state as input and performs content delivery by choosing $M(t)$ and $K(t)$ as actions. The agent learns the policy to maximize the accumulated reward function at a small timescale. The reward function $R_s(t)$ at time t is designed as follows:

$$R_s(t) = k_1\sqrt{M(t)} + k_2 K(t) - k_3 I\{Q(t) = 0\} - k_4 |K_l - K_p| \quad (4)$$

where k_1, k_2, k_3, k_4 are scaling factors for each term. In (4), maximizing $R_s(t)$ leads to the maximization of $M(t)$ and $K(t)$. However, since the user's satisfaction is directly influenced by buffering and quality, regardless of how many chunks are accumulated in the user queue, $\sqrt{M(t)}$ is utilized but this also gives indirect effects to avoid queue emptiness. Additionally, to assign the penalty for buffering, $Q(t) = 0$, $I\{Q(t) = 0\}$ is included with the negative sign where $I\{.\}$ is the indicator function which becomes the unity if the argument is true and zero otherwise. $|K_l - K_p|$ represents the difference in quality between the last received chunk and the current one, constructed as a penalty to reduce quality fluctuation. In summary, our reinforcement learning agent aims to maximize the quality of received chunks while limiting buffering occurrence and quality fluctuations by maximizing $R_s(t)$.

Meanwhile, in scheduling process, it is necessary to schedule in a manner that ensures effective operation of content delivery, making appropriate scheduling decisions between nearby nodes caching low-quality content and distant node caching high-quality content, depending on the channel status. In particular, the state for the scheduling process consists of the quality of cached contents of node candidates for scheduling within the radius R, and the distance between the user and nodes. Choosing caching node $\alpha(t)$ is considered an action based on the state in a slow timescale. The agent learns the policy to max-

imize the reward function $R_{\alpha(t)}(nT_c)$ which depends on the action $\alpha(t)$ and the content delivery process in the near future. Therefore, $R_{\alpha(t)}(nT_c)$ is defined as the summation of the immediate rewards achieved by receiving content chunks from the node $\alpha(t)$ in a fast timescale, as expressed by

$$R_{\alpha(t)}(nT_c) = \sum_{t=nT_c}^{(n+1)T_c-1} R_s(t) \quad (5)$$

In (3), the reward at the slow timescale, $R_{\alpha(t)}(nT_c)$, is set as the cumulative value of the reward function at the fast timescale $R_s(t)$, over the duration of T_c. By designing the reward in this way, the two reinforcement learning algorithms for scheduling and content delivery tasks across the two different timescales can operate hierarchically and converge in a unified learning direction. Consequently, scheduling learns to ensure the effective operation of content delivery for users who moves through regions of multiple caching nodes.

Ⅲ. Hierarchical Reinforcement Learning

In this study, an algorithm that performs well in random environments was needed because the channel capacity changes randomly at each time t without undergoing a separate channel estimation process. We adopt the DQN for designing the hierarchical reinforcement learning described in Section II. Q-Learning does not use a neural network, making it more vulnerable to random environmental changes compared to DQN. Other recent reinforcement learning algorithms such as A2C, PPO, DDPG use a policy network, which requires more resources and incurs more delay when operating online. Therefore, DQN, which utilizes only a value network and can perform well in random channel environments, is adopted for both scheduling and content delivery.

The DQN agent measures the action value given the state and selects the action with the highest action value, using an epsilon-greedy policy. Therefore, in the scheduling task, the distance between each node and the user, and the quality of content cached by each node are considered, and the Q network measures the value for all nodes to select the optimal $\alpha(t)$. In the content delivery task, the Q network measures the action value for all possible combinations of $M(t)$ and $K(t)$ based on the user queue length status $Q(t)$ and the candidate set of potential quality that the scheduled node $\alpha(t)$ (i.e., K_l) can provide. Then, the optimal combination of $M(t)$ and $K(t)$ is selected depending on action values.

The details of hierarchical DQN algorithm for node scheduling and content delivery are described in Fig. 1. In Fig. 1, θ_1 denotes parameters of the Q network for content delivery, and the Q network for node scheduling is parameterized by θ_2. DQN starts the training after a certain number of samples have accumulated in the replay memory, hence there are starting points for learning, denoted as t_{s_1}, t_{s_2} .

Algorithm 1 Hierarchical DQN Algotithm

for $episode = 1$ **to** $max\_episode$ **do**
  $t = 0$
  while not $User\ arrives\ end\ point$ **do**
    if $t \bmod T_c = 0$ **then**
      Check state $[l, d]$, where $l, d \in N(t)$
      θ_2 Network Select Caching node $\alpha(t)$
      Store Replay Memory $\leftarrow [l, d], \alpha(t), R_{\alpha(t)}, [l', d']$
      $R_{\alpha(t)} = 0$
      if $t > t_{s2}$ **then**
        Update θ_2 Network
      end if
    end if
    Check state $[Q(t), K_l]$
    θ_1 Network Select $M(t), K(t)$
    Store Replay Memory $\leftarrow [Q(t), K_l], [M(t), K(t)], R_s(t), [Q(t)', K_l']$
    $R_{\alpha(t)} = R_{\alpha(t)} + R_s(t)$
    if $t > t_{s1}$ **then**
      Update θ_1 Network
    end if
    $t = t + 1$
  end while
end for

Fig. 1. Hierarchical DQN algorithm for node scheduling and content delivery

Ⅳ. Experimental Results

This section provides extensive simulation results to demonstrate the proposed node scheduling and content delivery scheme for delay-sensitive and high-quality video services. We consider the square region of 1 km by 1 km, and caching nodes are distributed under the PPP with the intensity of $\lambda = 30$. The user is moving with the velocity of 2 m/s, and its moving direction is randomly generated. There are $L = 3$ quality levels for the desired content, and the user finds the node for receiving the chunks within

radius of $R = 50m$. Other system parameters are summarized in Table I.

We simulate the entire experiments with Window 10, Python v3.8.5, Tensorflow v2.4.0, CUDA v11.0, CuDNN v8.0.5 for ML software, and NVIDIA GeForce RTX 4090 (1ea), Intel(R) Core(TM) i9-13900K (24ea), and RAM 32GB (16GB x 2ea) for hardware. To train the reinforcement learning agent, the learning rate of 0.001, the replay memory size of 1000 or 2000, and the batch size of 64 are used.

In this paper, we compared the performance of the proposed technique with the "Nearest Node" method and the "Highest-Quality Node" method that heuristically schedule the caching node. The "Nearest Node" method always chooses the caching node closest to the user within the radius R for link scheduling, and utilizes DQN for video chunk delivery. On the other hand, the "Highest-Quality Node" method chooses the closest caching node having the highest-quality content, and utilizes DQN for video chunk delivery.

First, we test the proposed link scheduling and video chunk delivery scheme in different probabilistic caching policies, denoted by $\mathbf{p} = [p_1, p_2, p_3]$, where p_l is the probability of caching the content with quality level l, i.e., l layers of SVC-encoded video content. Probabilistic caching policies of cases 1, 2, and 3 are $\mathbf{p_1}$ =[4/7, 2/7, 1/7], $\mathbf{p_2}$ =[1/3, 1/3, 1/3], and $\mathbf{p_3}$ =[1/7, 2/7, 4/7], respectively.

Table 1. System Parameters.

Intensity Lambda	3
Learning Rate	0.001
User Velocity	2 [m/s]
Replay Memory Size	1000, 2000
Batch Size	64
Epsilon Decay	0.99999
User Radius	50 [m]
Mileage	1 [km]
Coherence Time	1 [s]
T_c	5 [s]
t_c	1 [s]
k_1	1
k_2	30
k_3	10
k_4	1

Case 1 indicates a scenario where nodes caching content at low quality are more prevalent, and the nodes are more likely to cache high-quality content in Case 3. Case 2 represents a scenario where nodes cache content at all quality levels with equal probability.

Figs. 2 and 3 show the average buffering rate and average quality of the received chunks for three comparison techniques. As the probability of caching high-quality files increases, users can, on average, receive higher quality, as seen in Fig 3. Additionally, the buffering rate decreases with the probability of caching high-quality files as shown in Fig 2, because if only nodes that have cached low-quality files are present nearby when the user requests the high-quality one, delays can occur. Here, "Highest-Quality Node" always pursues the highest-quality content irrespective of the channel condition; hence, it could suffer from the communication bottleneck which leads to a large

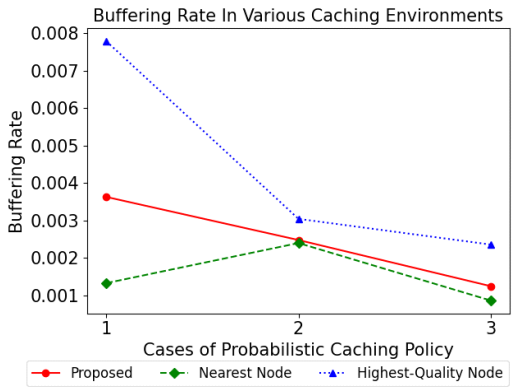

Fig. 2. Buffering Rate in Various Caching Environments.

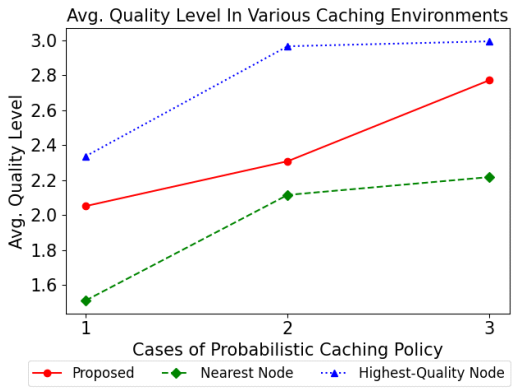

Fig. 3. Avg. Quality in Various Caching Environments.

buffering rate. On the other hand, “Nearest Node” is stable because it always schedules the nearest node, but it does not consider the quality of cached content; hence, its quality performance could be degraded significantly. By adjusting the coefficients in the reward function, the proposed scheme can achieve almost the same quality performance while limiting the buffering rate comparable to that of “Nearest Node,”

Figs. 4 and 5 show the plots of the buffering rate and quality performance versus transmit SNR of the caching node, respectively, when the probabilistic caching policy is random and not known to the user in advance. As the transmit SNR increases, the caching node can deliver higher quality and more video chunks. As shown in equation (2), a combination of decisions on the number and quality of chunks is limited to the channel capacity. Because the channel capacity increases with the SNR, more possible combinations of these decisions are available and the performances of average video quality and buffering rate finally get better as the SNR grows. In addition, similar to Figs. 2 and 3, the proposed method compensates for the disadvantages of “Highest-Quality Node” and “Nearest Node” methods and appropriately controls the trade-off between the quality and buffering rate.

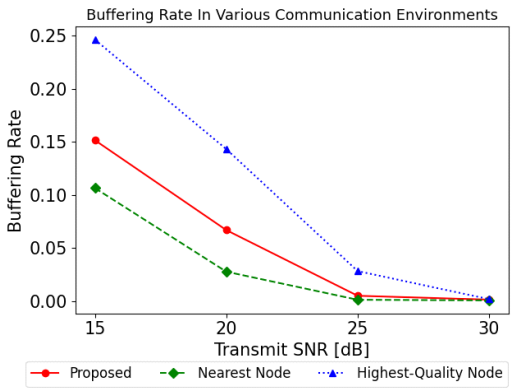

Fig. 4. Buffering Rate vs. transmit SNR

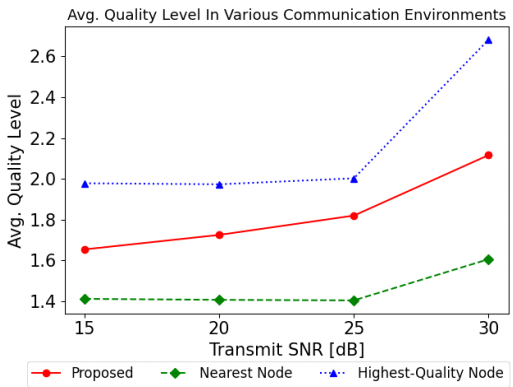

Fig. 5. Avg. Quality vs. transmit SNR

In Figs. 2-5, the buffering rates for all methods show slight differences in results. This is because buffering occurs only when there are no chunks in the user queue, and therefore the buffering event is more dependent on content delivery. On the other hand, the average quality level is more dependent on scheduling, leading to clear differences in average quality level among comparison schemes. Therefore, the average quality is a more meaningful indicator for evaluating different scheduling methods.

In Figs. 6 and 7, we compare our proposed link scheduling and video chunk delivery method to the reinforcement learning-based scheduling with the heuristic video delivery approaches. The “Most Chunk” method sends the maximum number of the lowest-quality chunks within the channel capacity, which is the best. The “Highest-Quality Chunk” method sends the maximum chunks with the highest quality within the channel capacity. From Figs. 6 and 7, it can be observed that the performances of heuristic video delivery methods do not improve despite the improvement in caching environments. This is because both methods do not take into account the user’s queuing system information. If chunks are sufficiently accumulated in the user queue, content can be transmitted with high quality and large capacity without buffering concerns. However, both the “Most Chunk” and “Highest-Quality Chunk” methods attempt to transmit within the average channel capacity, resulting in repetitive attempts at the same content delivery despite improvements in caching network environments. Consequently, no performance improvement is observed. The results show the importance of reflecting the user’s queuing system in content delivery decisions.

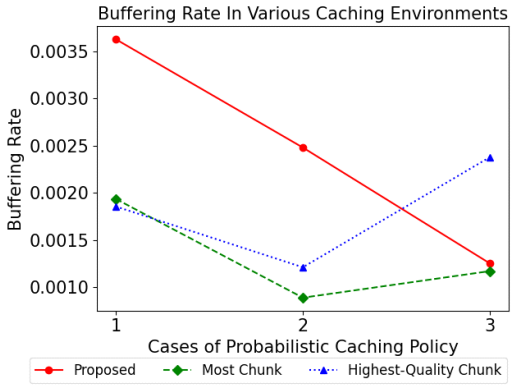

Fig. 6. Buffering Rate in Various Caching Environments.

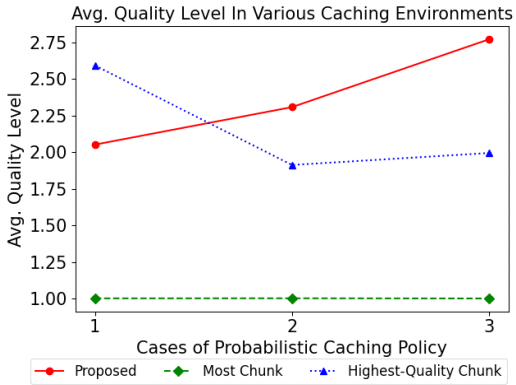

Fig. 7. Avg. Quality in Various Caching Environments.

Table 2. Quality Fluctuation Rate

SNR[dB]	15	20	25	30
Fluctuation Rate	0.00141	0.00155	0.00217	0.02340

Table 2 summarizes the rate of quality fluctuations, defined as the average change in subsequent quality decisions, in video playback for Case 1 across different SNRs. Since we consider the difference in quality between the last received chunk and the current one as a reward component in equation (4), Table 2 verifies that even in poor caching network environments, quality fluctuation occurs very infrequently.

V. Conclusion

In this paper, we propose a methodology for efficiently transmitting video content in vehicular networks when caching nodes are distributed. Through hierarchical reinforcement learning, we presented results that mitigate the drawbacks of existing approaches by addressing double timescales for two different tasks, node scheduling and content delivery.

Vehicular networks are exposed to various caching network environments due to the randomness of user mobility and distributions of caching nodes. Therefore, there is a need for a content delivery method that performs well not only in good network conditions but also in poor ones. In this study, we propose an hierarchical reinforcement learning algorithm that dynamically control scheduling and delivery decisions depending on the time-varying environment, resulting in well operation even in adverse caching network environments. By adjusting the coefficients in the reward function, it is possible to optimize learning towards the desired performance, considering the trade-off between the buffering rate and bitrate. Thus, this study developed a flexible content delivery method adaptable to various environments.

References

[1] Cisco, "*Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022 White Paper*," 2019.

[2] Y. Guo, et al., "Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach," *IEEE Trans. Veh. Technol.* vol. 67, no. 6, pp. 5445-5459, 2018. (https://doi.org/10.1109/TVT.2018.2817210)

[3] Y. Kim, H. Kim, S. Oh, and M. Choi, "A study on deep Q-network algorithm for delay-sensitive video streaming service," in *Proc. Symp. KICS,* pp. 1090-1091, Korea, Feb. 2023.

[4] R. F. Atallah, C. M. Assi, and M. J. Khabbaz, "Scheduling the operation of a connected vehicular network using deep reinforcement learning," *IEEE Trans. Intell. Transport. Syst.,* vol. 20, no. 5, pp. 1669-1682, 2018. (https://doi.org/10.1109/TITS.2018.2832219)

[5] E. Yaacoub, F. Filali, and A. Abu-Dayya, "QoE enhancement of SVC video streaming

over vehicular networks using cooperative LTE/802.11p communications," *IEEE J. Sel. Topics in Sign. Process.,* vol. 9, no. 1, pp. 37-49, 2014. (https://doi.org/10.1109/JSTSP.2014.2330343)

[6] S. Zhang, et al., "Towards fresh and low-latency content delivery in vehicular networks: An edge caching aspect," *2018 10th Int. Conf. WCSP, IEEE*, 2018. (https://doi.org/10.1109/WCSP.2018.8555643)

[7] H. Wu, et al., "Delay-minimized edge caching in heterogeneous vehicular networks: A matching-based approach," *IEEE Trans. Wireless Commun.,* vol. 19, no. 10, pp. 6409-6424, 2020. (https://doi.org/10.1109/TWC.2020.3003339)

Yunoh Kim

Feb. 2023 : B.S. degree, Kyung Hee University
Mar. 2023~Current : M.S. student, Kyung Hee University
<Research Interests> Reinforcement Learning, Federated Learning, and Wireless Caching Networks.

Tiange Xiang

Feb. 2020 : B.S. degree, Konkuk University
Feb. 2022 : M.S. degree, Hanyang University
Mar. 2023~Current : Ph.D. student, Kyung Hee University
<Research Interests> Semantic Communication, Multi-Agent Reinforcement Learning, and Stochastic Network Optimization.

Yeongjin Kim

2011 : B.S. degree, Korea Advanced Institute of Science and Technology (KAIST)
2013 : M.S. degree, KAIST
2018 : Ph.D. degree, KAIST
2018~2020 : Samsung Electronics
2020~Current : Assistant Professor with the Department of Electronic Engineering, Inha University, Incheon, South Korea.
<Research Interests> Hybrid Cloud/Edge Computing, Storage, and Networking Architecture.

Minseok Choi

2011 : B.S. degree, Korea Advanced Institute of Science and Technology (KAIST)
2013 : M.S. degree, KAIST
2018 : Ph.D. degree, KAIST
2019~2020 : Visiting Postdoc, University of Southern California.
2020~2022 : Assistant Professor, Jeju National University.
2022~Current : Assistant Professor with the Department of Electronic Engineering, Kyung Hee University, Yongin, South Korea.
<Research Interests> Wireless Caching Networks, Federated Learning, Reinforcement Learning, and Stochastic Network Optimization.