International Journal of Internet, Broadcasting and Communication Vol.16 No.4 36-45 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.4.36

IJIBC 24-4-5

Energy-Efficient Offloading with Distributed Reinforcement Learning for Edge Computing in Home Networks

Ducsun Lim* and Dongkyun Lim

**Post-Doc, Department of Computer Software, Hanyang University, Korea*
Professor, Department of Computer Science Engineering, Hanyang Cyber University, Korea
imcoms@hanyang.ac.kr, eiger07@hycu.ac.kr

Abstract

This paper introduces a decision-making framework for offloading tasks in home network environments, utilizing Distributed Reinforcement Learning (DRL). The proposed scheme optimizes energy efficiency while maintaining system reliability within a lightweight edge computing setup. Effective resource management has become crucial with the increasing prevalence of intelligent devices. Conventional methods, including on-device processing and offloading to edge or cloud systems, need help to balance energy conservation, response time, and dependability. To tackle these issues, we propose a DRL-based scheme that allows flexible and enhanced decision-making regarding offloading. Simulation results demonstrate that the proposed method outperforms the baseline approaches in reducing energy consumption and latency while maintaining a higher success rate. These findings highlight the potential of the proposed scheme for efficient resource management in home networks and broader IoT environments.

Keywords: *Distributed Reinforcement Learning, Internet of Things, Edge Computing, Home Networks*

1. Introduction

With the rapid advancement of the Internet of Things (IoT), smart devices (SDs) have become integral components in numerous aspects of our daily lives [1]. The increasing utilization of SDs in domains such as smart homes, smart healthcare, and smart cities necessitates the development of complex applications and large-scale data-processing capabilities [2]. Nevertheless, SDs are limited in their capacity to effectively address these requirements because of their constrained battery life and computational capabilities.

Cloud computing was introduced to address these issues; however, the geographical distance between cloud data centers and smart devices results in high latency and energy consumption [3]. This issue is particularly problematic for applications that require real-time processing, for which cloud computing may not be a suitable solution.

To address these challenges, edge computing is emerging as a solution that provides cloud-like

Manuscript Received: September. 5, 2024 / Revised: September. 10, 2024 / Accepted: September. 15, 2024
Corresponding Author: eiger07@hycu.ac.kr
Tel: +82-02-2290-0301, Fax: +82-02-2290-0600
Professor, Department of Computer Science Engineering, Hanyang Cyber University, Korea

functionalities in close proximity to devices, thereby reducing latency and energy consumption [4]. Nevertheless, conventional edge-computing solutions predominantly focus on industrial environments, rendering them cost-prohibitive and intricate for implementation in residential network settings.

Therefore, there is a growing need for lightweight edge-computing solutions suitable for home networks [5]. A lightweight edge-computing system for home networks can address the resource limitations of SDs while improving user convenience and energy efficiency.

In this study, we propose an offloading decision method using Distributed Reinforcement Learning (DRL) to minimize energy consumption and ensure reliability in a lightweight edge-computing environment for home networks. DRL considers each smart device an agent, allowing them to learn individually or cooperatively. Through this, SDs can monitor their status (e.g., battery level, workload, and network conditions) in real-time and adapt to environmental changes to make optimal offloading decisions.

By leveraging DRL, each device learns to optimize energy consumption and reliability by selecting actions, such as local processing, edge offloading, and cloud offloading. The reward function is designed to reflect the goal of minimizing energy consumption and maximizing reliability, thereby improving the resource utilization efficiency among devices. Furthermore, the overall performance of the home network system can be enhanced by enabling distributed agents to cooperate in learning.

The structure of this paper is as follows: Section 2 introduces related work; Section 3 explains the proposed system model and problem definition; Section 4 presents the offloading decision algorithm using DRL in detail; Section 5 evaluates the algorithm's performance through simulations; and finally, Section 6 concludes the paper and discusses future research directions.

2. Related Work

2.1 Edge Computing and Offloading Techniques

Cloud computing has been introduced to address the limited computational power and battery life of IoT devices. However, owing to its high latency and energy consumption, it is unsuitable for real-time applications [6]. To overcome this, edge computing has been proposed, which provides computational resources close to the devices, thereby reducing latency and energy consumption [7].

In edge computing environments, offloading decision techniques primarily establish optimal offloading strategies by considering factors such as the characteristics of tasks, network conditions, and resource status of devices [8]. For instance, [9] proposed an offloading policy aimed at minimizing latency and energy consumption, while [10] studied a multi-user offloading technique to enhance the energy efficiency of user devices.

2.2 Offloading Decisions Using Reinforcement Learning

Lightweight edge computing solutions tailored for home network environments can enhance user convenience and energy efficiency [11]. For example, [12] proposed a method to improve energy efficiency through cooperation between SDs in home networks, whereas [13] developed a lightweight platform for resource management in household IoT devices. Nevertheless, these studies have limitations in optimizing energy consumption and reliability in offloading decisions. Therefore, there is a growing need for research on offloading decision techniques that apply distributed reinforcement learning to respond to real-time changes in the SD status while optimizing energy consumption and reliability.

2.3 Lightweight Edge Computing in Home Networks

Lightweight edge computing solutions tailored for home network environments can enhance user convenience and energy efficiency [14]. For example, [15] proposed a method to improve energy efficiency through cooperation between SDs in home networks, while [16] developed a lightweight platform for resource management in household IoT devices.

Nevertheless, these studies have limitations in optimizing energy consumption and reliability in offloading decisions. Therefore, there is a growing need for research on offloading decision techniques that apply distributed reinforcement learning to respond to real-time changes in SD status while optimizing energy consumption and reliability.

3. System Model

In this paper, we present a system model for offloading decisions in a lightweight edge-computing environment tailored for home networks. The proposed model provides a foundation for establishing offloading strategies for each device, considering energy consumption and reliability.

The smart-home network environment consists of *N* smart devices, each generating tasks that need to be processed. The system comprises SDs, lightweight edge-computing nodes, and a cloud server. SDs are IoT devices with limited computational capabilities and battery life that generate tasks like sensing data processing or responding to user requests. Lightweight edge computing nodes are deployed near the SDs to handle offloaded tasks. At the same time, the cloud server, located at the core of the network, processes tasks that cannot be handled by the edge nodes or those requiring large-scale computations.

Each SD_i generates a task τ_i, characterized by its data size D_i (in bits), computational workload C_i (in cycles), and maximum allowable delay τ_i^{max} (in seconds). To process a task, the device can choose from three options: local processing, where the task is handled by the device itself; edge offloading, where the task is offloaded to a nearby lightweight edge node; and cloud offloading, where the task is offloaded to the cloud server for processing.

3.1 Energy Consumption and Delay

The task-processing method determines the energy consumption and delay for an SD, which consists of computational and communication energy. The local processing energy consumption, E_i^{loc}, refers to the energy consumed when the task is processed on the device and is defined in (1).

$$E_i^{loc} = \kappa\left(f_i^{loc}\right)^2 C_i \tag{1}$$

where κ is the circuit constant of the device, f_i^{loc} is the local CPU frequency of device *i*, and C_i is the computational workload of the task. The transmission energy required when offloading a task is the energy consumed for data transmission and is calculated in (2).

$$E_i^{tx} = P_i^{tx} T_i^{tx} \tag{2}$$

where, P_i^{tx} is the transmission power of the device, represents the power consumed during data transmission, and T_i^{tx} is the data transmission time, which refers to the time taken to send the task input data to an edge node or cloud server. The reception energy E_i^{rx} consumed when receiving the results of the

offloaded task is defined by (3) in.

$$E_i^{rx} = P_i^{rx} T_i^{rx} \tag{3}$$

P_i^{rx}represents the device's reception power, and T_i^{rx} is the time required to receive the processed task results. Hence, the transmission energy increases proportionally with the transmission power and data transmission time, directly affecting the energy consumption of the device. The total offloading energy, E_i^{off} is the sum of the transmission and reception energy, and is defined in (4).

$$E_i^{off} = E_i^{tx} + E_i^{rx} \tag{4}$$

The total delay incurred while processing a task is the sum of all the times required for task completion, which includes the data transmission time, processing time, and result reception time. The local processing delay, T_i^{loc}, refers to the time taken to process the task on the device. It is calculated by dividing the computational workload by the device's local CPU frequency, as in (5).

$$T_i^{loc} = \frac{C_i}{f_i^{loc}} \tag{5}$$

Where C_i is the task's computational workload, and f_i^{loc} is the local CPU frequency of device *i*. A higher computational capacity of the SD reduces the processing time, whereas larger workloads increase it. The offloading delay, T_i^{off}, is the total time incurred when offloading a task, consisting of data transmission, task processing, and result reception time. Each time is defined as follows:

- This is the time to transmit task input data from the device to the edge node or cloud server. It is calculated by dividing the task data size D_i by the uplink transmission rate R_i^{up}, as in (6).

$$T_i^{tx} = \frac{D_i}{R_i^{up}} \tag{6}$$

- This is the time required to process the task on the server, calculated by dividing the task's computational workload C_i by the server's CPU frequency f^{ser}. A higher computational capacity of the server reduces the processing time and is defined in (7).

$$T_i^{proc} = \frac{C_i}{f^{ser}} \tag{7}$$

- This refers to the time required to receive the processed task results from the device. It is calculated by dividing the result data size D_i^{res} by the downlink transmission rate R_i^{down}, defined in (8).

$$T_i^{rx} = \frac{D_i^{res}}{R_i^{down}} \tag{8}$$

Therefore, when processing a task, the total offloading delay is the sum of the data transmission time, task processing time, and result reception time and is defined in (9).

$$T_i^{off} = T_i^{tx} + T_i^{proc} + T_i^{rx} = \frac{D_i}{R_i^{up}} + \frac{C_i}{f^{ser}} + \frac{D_i^{res}}{R_i^{down}} \quad (9)$$

This equation provides the total delay incurred during offloading, which varies based on network conditions, data size, and server processing capacity.

3.2 Problem Statement

The goal of this study was to optimize the offloading decisions of each device to minimize overall energy consumption while ensuring reliability. The offloading decision for each device *i*, denoted as $a_i \in \{0, 1, 2\}$, is formulated as an optimization problem. Here, $a_i = 0$ represents local processing, $a_i = 1$ represents edge offloading, and $a_i = 2$ represents cloud offloading.

- The task is processed on the device itself, incurring no network costs but constrained by its computational capacity and battery life.
- The task is offloaded to the edge node, where it is processed. Although network transmission is required, the edge node offers higher computational power and is located near the device, thereby reducing latency.
- The task is offloaded to a cloud server with the highest computational power. However, the distance to the cloud may introduce a higher latency.

The objective of this optimization problem is to minimize the energy consumption resulting from each device's offloading decision a_i. The objective function is defined by Equation (10).

$$\min_{\{a_i\}} \sum_{i=1}^{N} E_i(a_i) \quad (10)$$

where $E_i(a_i)$ represents the energy consumption of device *i*, which is calculated based on the selected offloading option. Energy consumption includes computational energy for local processing and communication and processing energy for offloading. Offloading incurs additional energy for data transmission and reception as well as for processing energy at the server. In contrast, local processing consumes only computational energy on the device itself.

This problem is addressed in a distributed environment, where the offloading decisions of each device can influence the decisions of the others. Each device learns to make optimal offloading decisions based on state and network conditions.

4. Proposed Scheme

This study applies DRL as an offloading decision-making technique to minimize the energy consumption of smart devices and ensure reliability in a lightweight edge-computing environment for home networks. The proposed method allows each device to make offloading decisions independently or cooperatively with the aim of optimizing resource allocation while adapting to various environmental changes.

4.1 DRL-based Offloading Decision

Each device makes offloading decisions using reinforcement learning. DRL enables each SD to autonomously learn optimal actions based on its state and surrounding environment without relying on a central server. This reduces network load and allows each device to make independent decisions.

- **State** $(\boldsymbol{S_i})$: The current state of each device consists of battery level, task data size, computational requirements, and network conditions (uplink and downlink transmission rates).
- **Action** $(\boldsymbol{a_i})$: Each device chooses between local processing, edge offloading, or cloud offloading.
- **Reward** $(\boldsymbol{r_i})$: Devices receive a reward based on energy consumption and reliability. The reward function is designed to minimize energy consumption while ensuring reliability and is defined in (11).

$$r_i = -E_i(a_i) + \lambda \cdot R_i(a_i) \tag{11}$$

where $E_i(a_i)$ represents the energy consumption for the selected offloading option, and $R_i(a_i)$ represents the reliability of the chosen action. The parameter λ controls the trade-off between energy consumption and reliability.

4.2 Training Process of the Proposed Scheme

In the proposed method, Deep Q-Network (DQN) enables smart devices to learn optimal offloading decisions that minimize energy consumption while maintaining reliability. The DQN extends Q-learning using a neural network to approximate Q-values when the state-action space is enormous. Table 1 presents the pseudocode of the training process of the proposed DQN-based scheme.

Table 1. Proposed scheme

Step	Description
1	Initialize the Q-network with random weights and initialize the experience replay memory
2	***for*** each episode ***do***
3	Observe the current state s_i is (e.g., battery level, task size, network status)
4	Select an action ai using ϵ-greedy: explore with probability ϵ, or exploit using the Q-network
5	Execute the selected action ai (local processing, edge offloading, or cloud offloading)
6	Observe the reward r_i and the next state $s_i{}'$
7	*Store the experience* (s_i, a_i, r_i, s_i') *in the replay memory*
8	*Sample a minibatch of experiences from the replay memory*
9	*Compute the target Q-values and update the Q-network by minimizing the loss function*
10	*Decrease the exploration rate* ϵ *to balance exploration and exploitation*
11	***end for***

The Q-network was initialized to approximate state-action values, and an experience replay memory was set up to store the learning samples. Action selection is determined randomly or by the Q-network, depending on the exploration-exploitation trade-off. After performing an action, the device observes the new state and reward, which are stored in replay memory. The Q-network was trained by minimizing the loss function using mini-batches sampled from the replay memory. As the training progresses, the exploration rate is gradually reduced, increasing the frequency of selecting optimal actions. Through this iterative process, each device learns to make offloading decisions that optimize the energy consumption and reliability in the home network environment, ultimately leading to efficient resource utilization.

5. Performance Evaluation

In this study, we conducted experiments to evaluate the performance of the proposed DRL-based scheme in a home network environment by comparing it with several existing approaches. The primary performance metrics used for evaluation included energy consumption, latency, reliability, offloading ratio, and system resource utilization. We selected baseline algorithms, such as local processing, edge offloading, and cloud offloading.

The performance evaluation of the proposed scheme focuses on key metrics, including energy consumption, latency, reliability, offloading ratio, and system resource utilization. Energy consumption was measured as the total energy consumed by the devices, with the average and maximum/minimum energy consumption analyzed. Latency was assessed by measuring the total time required for task processing, including the average latency and the rate of tasks exceeding the latency threshold. Reliability is quantified as the ratio of successfully processed tasks and is a critical metric for evaluating system robustness. The offloading ratio analyzes which offloading strategy—local, edge, or cloud— is selected for task processing. Finally, system resource utilization is evaluated by measuring the CPU and memory usage in edge and cloud servers, providing insight into the system load and resource efficiency. Table 2 summarizes the main hyperparameters used in the experiments.

Table 2. Hyperparameters

Parameter	Description	Value
Learning Rate	The rate of updating the model	0.001
Discount Factor	Determines the present value of future rewards	0.99
Exploration rate	Probability of selecting random actions. Higher at the start to encourage exploration	1.0
Batch Size	Number of data points processed per batch	64
Replay buffer size	The size of the memory storing past experiences for learning. Larger sizes allow more diverse learning	50000

Local Processing (AL) evaluates energy consumption and latency by processing all tasks directly on the device. The Edge Offloading method (AE) transmits tasks to edge computing nodes for processing, considering network transmission costs and load. The Cloud Offloading method (AC) sends tasks to cloud servers for processing, assessing the trade-off between higher processing capabilities and longer latency and guiding the offloading decision.

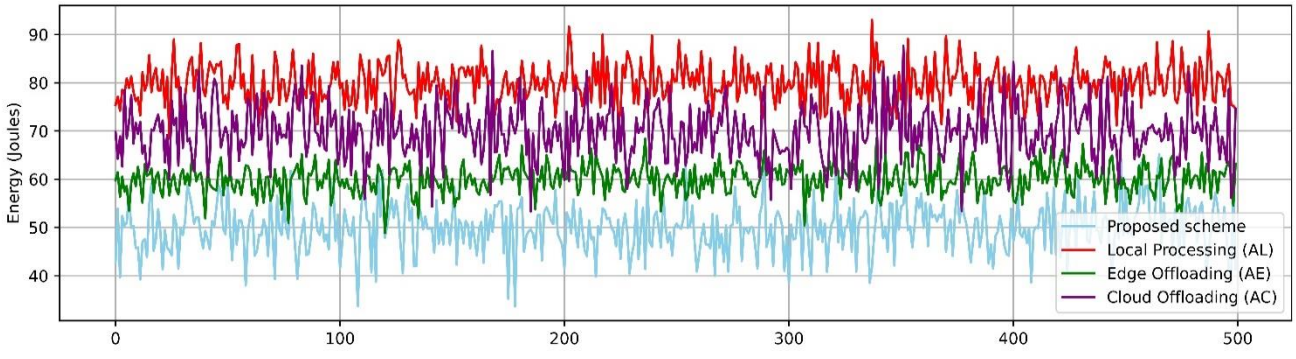

Figure 1. Energy Consumption Over 500 Episodes

Figure 1 presents a comparative analysis of the energy consumption of the proposed scheme and several benchmark methods. The proposed scheme achieves the lowest overall energy consumption compared with the other approaches. This demonstrates the effectiveness of the proposed algorithm in minimizing energy consumption through efficient resource utilization during task processing.

In contrast, local processing exhibited the highest energy consumption, reflecting the significant energy expenditure of handling all tasks directly on the device. Although the edge-offloading method consumes less energy than local processing, it still demonstrates higher energy usage than cloud offloading and the proposed algorithm. Cloud offloading tends to consume slightly more energy than edge offloading but remains more energy-efficient than local processing.

In summary, the proposed scheme outperforms other methods in terms of energy efficiency, highlighting its potential for application in resource-constrained environments such as home networks.

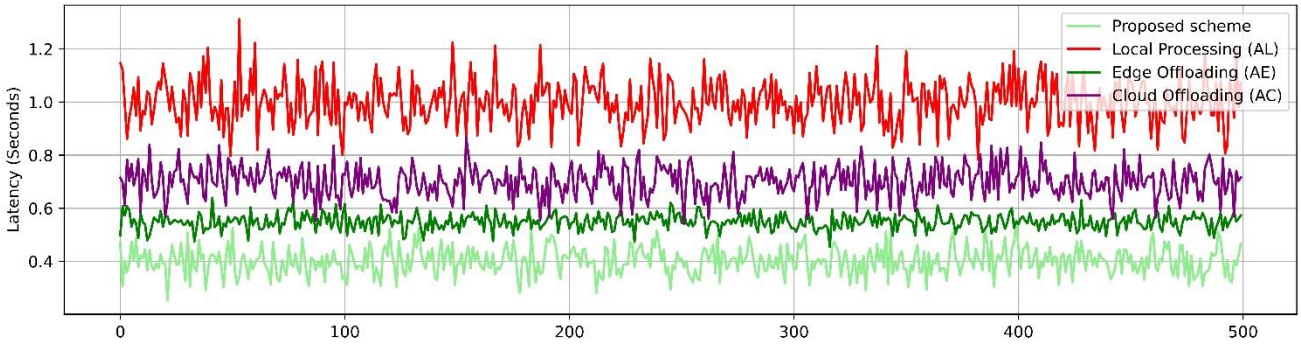

Figure 2. Latency Over 500 Episodes

Figure 2 compares the latency of the proposed scheme with those of the other benchmark methods. The proposed scheme consistently demonstrated the lowest latency among the compared methods, indicating its effectiveness in minimizing delays by efficiently distributing tasks for processing. Although the edge offloading method shows a relatively low latency, it is still slightly higher than the proposed scheme. In contrast, cloud offloading exhibits a longer latency than edge offloading, primarily because of the additional time required for data transmission and processing on the cloud server. The local processing method has the highest latency, which can be attributed to the device handling all tasks internally, resulting in significant delay. In conclusion, the proposed scheme outperforms the other methods in terms of latency, making it particularly advantageous for latency-sensitive applications.

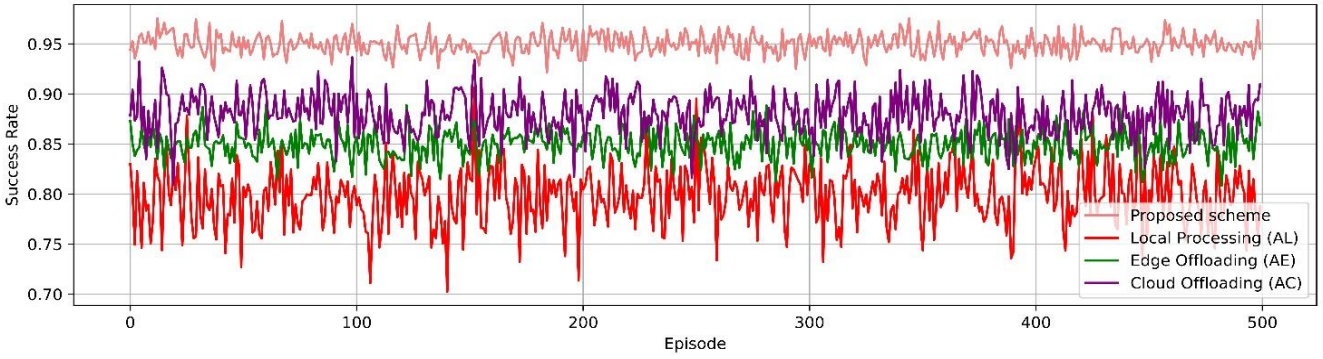

Figure 3. Success Rate Over 500 Episodes

Figure 3 presents a comparison of the success rates of the proposed scheme and several other methods. The proposed scheme maintained the highest success rate, consistently achieving an average of over 0.95. This demonstrates their stable and reliable performance. In contrast, the local processing method exhibits a relatively lower success rate, which is attributed to limited resources and high latency when handling all tasks on the device. Edge and cloud offloading methods show intermediate success rates, with some variability depending on the availability of edge and cloud resources.

In conclusion, the proposed scheme achieves a high success rate and provides a more stable performance compared to other methods. This suggests that it can significantly enhance reliability and prevent performance degradation in resource-constrained home network environments. The performance improvements observed with the proposed scheme can also be effectively applied to other IoT environments in which resource management is critical.

6. Conclusion

This study proposes a DRL-based decision-making scheme to minimize energy consumption and ensure the reliability of smart devices within home network environments. By enabling each device to learn the optimal offloading decision based on its own state and network conditions, the proposed algorithm demonstrated superior energy efficiency and latency performance compared to traditional rule-based offloading methods. The performance evaluations indicate that the proposed scheme outperforms existing algorithms in terms of energy consumption, latency, and reliability.

The proposed scheme is expected to adapt flexibly to more complex network- and resource-management challenges. Future research will explore solutions to offloading decision problems in more complex network scenarios and investigate applications for various services.

Acknowledgment

This work was partially supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2020-0-01373, Artificial Intelligence Graduate School Program (Hanyang University)) and the research fund of Hanyang University (HY-2024)

References

[1] Atzori, L., Iera, A., & Morabito, G., "The internet of things: A survey," *Computer networks*, Vol. 54, No. 15, pp. 2787-2805, 2010. DOI: https://doi.org/10.1016/j.comnet.2010.05.010

[2] Lim, D., Lee, W., Kim, W. T., & Joe, I., "DRL-OS: a deep reinforcement learning-based offloading scheduler in mobile edge computing," *Sensors*, Vol. 22, No. 23, pp. 9212, 2022. DOI: https://doi.org/10.3390/s22239212

[3] Khan, A., Al-Zahrani, A., Al-Harbi, S., Al-Nashri, S., & Khan, I. A., "Design of an IoT smart home system," in *Proc. 2018 15th Learning and Technology Conference (L&T)*, pp. 1-5, Feb. 2018. DOI: https://doi.org/10.1109/LT.2018.8368484

[4] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A., "Cloud computing—The business perspective," *Decision support systems*, Vol. 51, No. 1, pp. 176-189, 2011. DOI: https://doi.org/10.1016/j.dss.2010.12.006

[5] Morabito, R., Cozzolino, V., Ding, A. Y., Beijar, N., & Ott, J., "Consolidate IoT edge computing with lightweight virtualization," *IEEE network*, Vol. 32, No. 1, pp. 102-111, 2018. DOI: https://doi.org/10.1109/MNET.2018.1700175

[6] Satyanarayanan, M., "The emergence of edge computing," *Computer*, Vol. 50, No. 1, pp. 30-39, 2017. DOI: https://doi.org/10.1109/MC.2017.9

[7] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L., "Edge computing: Vision and challenges," *IEEE internet of things journal*, Vol. 3, No. 5, pp. 637-646, 2016. DOI: https://doi.org/10.1109/JIOT.2016.2579198

[8] Chiang, M., & Zhang, T., "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, Vol. 3, No. 6, pp. 854-864, 2016. DOI: https://doi.org/10.1109/JIOT.2016.2584538

[9] Chen, X., Jiao, L., Li, W., & Fu, X., "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM transactions on networking*, Vol. 24, No. 5, pp. 2795-2808, 2015. DOI: https://doi.org/10.1109/TNET.2015.2487344

[10] Huang, L., Feng, X., Zhang, L., Qian, L., & Wu, Y., "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, Vol. 19, No. 6, pp. 1446, 2019. DOI: https://doi.org/10.3390/s19061446

[11] Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., & Wang, W., "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, Vol. 5, pp. 6757-6779, 2017. DOI: https://doi.org/10.1109/ACCESS.2017.2685434

[12] Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B., "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, Vol. 19, No. 4, pp. 2322-2358, 2017. DOI: https://doi.org/10.1109/COMST.2017.2745201

[13] Huang, L., Bi, S., & Zhang, Y. J. A., "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, Vol. 19, No. 11, pp. 2581-2593, 2019. DOI: https://doi.org/10.1109/TMC.2019.2928811

[14] Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., & Shi, W., "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, Vol. 107, No. 8, pp. 1697-1716, 2019. DOI: https://doi.org/10.1109/JPROC.2019.2915983

[15] Alhasnawi, B. N., Jasim, B. H., Siano, P., & Guerrero, J. M., "A novel real-time electricity scheduling for home energy management system using the internet of energy," *Energies*, Vol. 14, No. 11, pp. 3191, 2021. DOI: https://doi.org/10.3390/en14113191

[16] Al Salami, S., Baek, J., Salah, K., & Damiani, E., "Lightweight encryption for smart home," in Proc. 2016 11th International conference on availability, reliability and security (ARES), pp. 382-388, 2016. DOI: https://doi.org/10.1109/ARES.2016.40