Deep-Reinforcement-Learning-Based Age-of-Information-Aware Low-Power Active Queue Management for IoT Sensor Networks

Taewon Song and Yeunwoong Kyung

Abstract—**As the number of Internet of Things (IoT) sensors increases and their deployment becomes denser, power management for IoT sensor networks becomes more important. In most IoT sensor networks, one cluster head (CH) collects data from a large number of sensors and forwards them to backbone networks. Thus, managing CH's queue condition is crucial in order to extend the network's lifespan or satisfy Quality-of-Service (QoS) requirements. Meanwhile, reducing Age of Information (AoI), a metric describing how fresh information is, has become one of the most important metrics, from simple data, such as temperature and humidity to more complex data that must be timely, such as vehicle information and road dynamics. However, it is shown that AoI may heavily fluctuate depending on the medium access control protocol. In this article, we propose a deep-reinforcement-learning-based AoI-aware low-power active queue management for IoT sensor networks. To this end, we formulate a Markov decision process model in which the CH can select one of the actions, including forward, flush, or leave buffered data from associated cluster member nodes. Extensive simulations show that compared with traditional queue management methods, our queue management method can reduce the power consumption of CH while trying not to exceed the AoI value threshold, thereby enabling IoT sensor networks to be stable while ensuring satisfactory QoS.**

Index Terms—**Active queue management (AQM), deep-*Q* network (DQN), deep reinforcement learning (DRL), wireless sensor networks (WSNs).**

I. Introduction

A WIRELESS sensor network (WSN) in Internet of Things (IoT) is a wireless network used to monitor a large number of sensors without infrastructure. It is commonly employed to oversee physical and environmental conditions of various systems [1]. Wireless IoT sensors are convenient, easy to use, and can monitor conditions that were previously difficult to monitor. With the increasing number of IoT devices and systems, wireless IoT sensors are becoming more important for ensuring efficient and effective operation.

One of the major challenges of WSNs is that the devices are usually battery-powered and have limited energy resources. Indeed, in the era of "big data," a massive amount of data is putting a huge burden on the devices, making low power consumption one of its most significant performance metrics. Therefore, energy efficiency is an important issue in WSN design.

Besides energy efficiency, another important challenge for WSNs is to guarantee Quality of Service (QoS), which includes latency, throughput, reliability, and so on. Among these features, we are focusing on Age of Information (AoI), which is the time elapsed from the creation of the most recently generated data until it is received by the destination. Among the large number of applications that use WSN-based IoT networks, especially monitoring-related applications, it can be important to know how up-to-date the data is from the time it was generated. Therefore, AoI is an appropriate performance metric to measure the performance of WSN applied to the IoT environment.

Maintaining low AoI values while minimizing battery consumption is a difficult task. For instance, frequent transmissions may ensure low AoI, but this approach can lead to a higher number of transmissions and retransmissions caused by collisions, which can significantly drain the battery. Consider, for example, a smart IoT sensor system deployed in a city infrastructure. These sensors gather data on various environmental factors, such as temperature, air quality, and noise levels, and communicate this information to a central hub. The hub then processes the data to provide real-time insights for city management, like environmental monitoring. Keeping the AoI low is crucial to ensure timely and relevant data, but it is equally important to minimize the energy consumption of the IoT sensors to prolong their battery life. Designing a WSN that can balance a low AoI with energy efficiency requires careful consideration of these tradeoffs.

In this article, we propose DeepAAQM, a deep reinforcement learning (DRL)-based algorithm for low-power active queue management (AQM) in IoT WSNs. With DeepAAQM, the agent in a cluster head (CH) can decide to forward, flush, or leave collected data based on timestamps that represent the moment the frame was created on the connected CM. We formulate a Markov decision process (MDP) model and

Manuscript received 28 December 2023; revised 31 December 2023; accepted 13 January 2024. Date of publication 18 January 2024; date of current version 25 April 2024. This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIT) under Grant 2022R1F1A1076069; in part by the "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) under Grant 2021RIS-004; and in part by the Soonchunhyang University Research Fund. *(Corresponding author: Yeunwoong Kyung.)*

Taewon Song is with the Department of Internet of Things, SCH MediaLabs, Soonchunhyang University, Asan 31538, Chungcheongnam, South Korea (e-mail: twsong@sch.ac.kr).

Yeunwoong Kyung is with the Division of Information and Communication Engineering, Kongju National University, Chunan 31080, Chungcheongnam, South Korea (e-mail: ywkyung@kongju.ac.kr).

Digital Object Identifier 10.1109/JIOT.2024.3355410

use deep-Q networks (DQN) [2] algorithm which is one of the well-known value-based algorithms that updates parameters using action–state value function to obtain an optimal policy. We simulate DeepAAQM on top of two well-known channel access methods, slotted ALOHA and CSMA/CA, and verify the proposed technique through extensive simulations. Performance analysis shows that DeepAAQM significantly reduces power consumption while satisfying an arbitrary peak AoI level.

To sum up, the contributions of this article are threefold.

1) We propose DeepAAQM, a DRL-based algorithm for low-power AQM in IoT WSNs that can maintain an AoI value while minimizing energy consumption with the optimized policy acting at the head of the queue.
2) We formulate an MDP model and obtain an optimal policy for DeepAAQM by using the DQN algorithm.
3) We conduct simulations on top of slotted ALOHA and CSMA/CA to verify the proposed algorithm, showing that DeepAAQM significantly reduces power consumption while maintaining a competent AoI value level.

The remainder of this article is organized as follows. Section II offers a concise overview of relevant studies as the initial step. We next describe the system model of DeepAAQM in Section III. The problem is formulated in Section IV. We next describe a learning-based algorithm for DeepAAQM in Section V. Performance evaluation results and conclusion are given in Sections VI and VII, respectively.

II. Related Works

A. Energy-Efficient Cluster-Based Communication Protocol

Routing protocols can be divided into two types: 1) flat routing and 2) cluster-based routing. Flat routing protocols have all sensor nodes performing the same role and functions in the network. On the other hand, in cluster-based routing, the network is divided into clusters and some nodes are selected as special nodes based on certain criteria. Since DeepAAQM is based on cluster-based routing, the introduction of related works will be limited to cluster-based routing protocols.

There has been a lot of research on energy-efficient cluster-based communication protocols, which can extend battery life and ensure devices are always operational and ready to transmit data, which is essential for many IoT applications. LEACH [3] is a cluster-based protocol that evenly distributes power consumption among sensors by randomly rotating the role of CH. Fuzzy-logic-based clustering methods, such as those in [4] and [5], build on and enhance LEACH. Q-LEACH [6] partitions the network to enable wide coverage and enhances stability period, network lifetime, and throughput. Haque et al. [7] proposed a hybrid approach that enhances the lifespan of WSNs by jointly considering network topology maintenance and construction algorithms. O-LEACH [8] minimizes energy usage by covering the entire network with a minimum number of orphaned nodes. However, in order to implement these approaches, the routing table must be dynamically modified and CH candidate nodes should have reception capability, which can be a hurdle to widely adapting IoT networks.

Existing research has mostly focused on routing optimization, while dynamic power management has received comparatively little attention. Queue management protocol based on queue threshold technique proposed by Maheswar et al. [9] can reduce the number of transmissions to improve network lifetime. Reinforcement learning for dynamic power management to ensure QoS for diverse applications is adopted by Hosahalli and Srinivas [10]. Alwasef et al. proposed an energy-efficient queue management scheme in [11]. The proposed algorithm saves buffer space and thus can decrease power consumption at CHs. However, these approaches require many parameters and they may require high operation capabilities for sensor nodes.

B. AoI Analysis

Besides energy-efficient communication protocols, analyzing the AoI has become crucial in WSNs for IoT. AoI measures the timeliness of information, which is critical in monitoring systems. This section reviews studies on AoI analysis in WSNs.

Kaul et al. [12] defined a metric known as the AoI that described how "fresh" collected data was. Specifically, if the most recent update at the CH carries a time stamp $U(t)$ at time t, then the AoI at the time $\Delta(t) = t - U(t)$. Therefore, AoI is a suitable metric to comprehensively describe the freshness of information that cannot be expressed only with the throughput and the delay. In addition, Sun et al. [13] argued that "zero-wait" policy, i.e., work-conserving scheduling, did not always minimize the average age. Yates et al. [14] introduced variations of AoI and mathematically analyzed them. Among the variations, we focus on peak AoI which is the maximum AoI within a particular duration as it can reflect the minimum freshness of information [15].

C. Active Queue Management

AQM is an intelligent packet-dropping technique that has been widely studied in the field of networking. It is designed to prevent network congestion by proactively dropping packets before the queue becomes full. Therefore, AQM is crucial for WSNs to balance low AoI values with low power consumption.

Random early drop (RED) [16] is one of the most extensively studied algorithms for AQM. RED tracks queue size with two thresholds, min and max. If the queue size is below the min threshold, no packet is discarded. If the queue size is between the min and max thresholds, packets can be discarded with a probability calculated using the max probability, average queue size, and min threshold. If the average queue size exceeds the max threshold, all packets are discarded. Stabilized RED (SRED) [17] is a representative variant algorithm based on RED. The SRED algorithm stabilizes queue size by estimating the number of active flows and instantaneous queue size, and computing packet drop probabilities based on this estimation. Controlled delay (CoDel) [18] is a relatively recently developed AQM technique published as RFC8289 [19] to address the bufferbloat issue, leading to

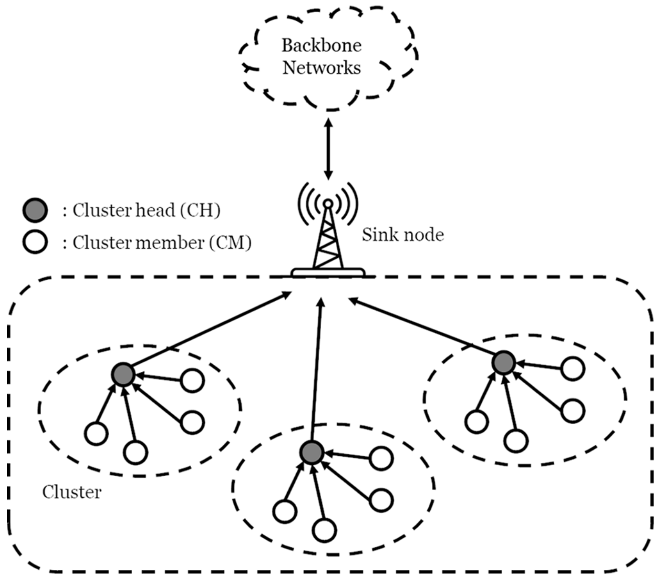


Fig. 1. WSN architecture.

reduced latency and jitter in packet-switched networks due to excessive packet buffering.

In a programmable network, such as software-defined network (SDN), the control plane is separated from the hardware and implemented in software. Integrating AQM into SDN [20], [21] improves network performance by reducing delays and boosting throughput, while keeping the data plane simple and using the global network view of its control plane. Further, a low-latency communication technology in SDN is reviewed in [22], especially focusing on traffic management and congestion control. Packet rank-aware AQM (PR-AQM) is introduced in [23], a system designed to execute AQM on a programmable data plane. This is achieved by real-time capture and analysis of packet rank distribution and the level of queue congestion. Although these studies underscore SDN's potential in queue management, our algorithm differs by addressing the AoI, which is a crucial metric for services using WSNs.

As machine learning has proven its practical applicability through multiple use cases in fields ranging from robotics to process automation, it is also being adopted and studied to optimize AQM. Reinforcement-learning-based AQM (RL-AQM) [24], based on reinforcement learning, is proposed to manage the network resources and keep the queue delay low at the same time. Deep Q-network-based AQM (DQN-AQ) [25] is proposed based on the DRL technique and a scaling factor to achieve the tradeoff between queuing delay and throughput is introduced. Meanwhile, the algorithms mentioned above decide whether to admit or discard the packet when it enters the queue. Due to the very dynamic nature of the wireless environment, these policies may have large fluctuations in AoI.

III. System Model

Suppose we have a WSN consisting of several cluster members (CMs), CHs with which some CMs are associated, and a sink node with which all CHs are associated. Fig. 1 illustrates a typical type of WSN. The CMs can either be monitoring IoT sensors, home appliances, platooning vehicles, or unmanned aerial vehicles (UAVs), according to their applications. One CH and several CMs form an area, which is called a cluster. Each CH performs the role of forwarding data monitored by and transmitted from its associated CMs. The sink node can typically be a base station (BS) for cellular networks or an access point (AP) for wireless local area networks. The CH is responsible for forwarding data from the sensors to a sink node. In order for the AoI value not to exceed a certain threshold, for example, it may be possible for the CH to transmit the data collected from the CMs in a fast cycle, but this may cause a serious amount of power consumption. Therefore, we need to minimize the amount of power consumed while guaranteeing to maintain an appropriate AoI value level as much as possible.

We consider a WSN network, as shown in Fig. 1, composed of one sink node, several wireless CHs, and lots of wireless CMs. Each wireless CM collects sensing data and transmits them to corresponding CH.[1] Each CH forwards the buffered sensing data to the sink node. It is assumed that the CH is within transmission range of the sink node and the CMs are within range of the associating CH. Transmission channels of clusters are not overlapped and thus each agent in each CH can operate independently. The transmissions occur through wireless medium via a certain wireless medium access protocol; in this literature, we consider slotted ALOHA and CSMA/CA without RTS/CTS exchange. During the medium access, time synchronization among the CH and the CMs may be needed. This could be achieved by additional GPS system [26] or introduced time synchronization algorithm [27]. However, since time synchronization technique is beyond the scope of this article, this will not be discussed further.

A specific system model, especially the timing diagram is elaborated as follows, assuming one cluster to make it easy to illustrate. Fig. 2 shows a timing diagram for DeepAAQM. In this situation, a sink node periodically disseminates a beacon frame that may contain various fields.[2] During one beacon interval, which is one *episode*, CMs compete with each other to acquire wireless channel to forward monitored data that might be temperature, velocity, or location depending on their applications. Once a CM gets a channel access opportunity, it transmits a frame containing the monitored data and the timestamp that data was observed as well. A *decision epoch* is defined as a frame duration or slightly longer than the duration for frame processing time.[3] At each decision epoch, an agent in a CH decides an *action* among action sets, including forward, flush, and leave. When the beacon interval ends, the sink node will broadcast the beacon frame including a calculated *return*, that is the accumulated sum of the *rewards*, through DeepAAQM. Once the CH receives the beacon frame,

[1] The role of CH can be rotated among CMs, but in this article, it is assumed that the role of CH is assigned to a specific CM during the simulation period.

[2] The name of the beacon frame and the fields contained in the beacon frame may differ depending on the communication protocol, so general terms are used in this article and are not limited to specific names.

[3] It is assumed that the data rates of the frames from CMs are constant in this article.

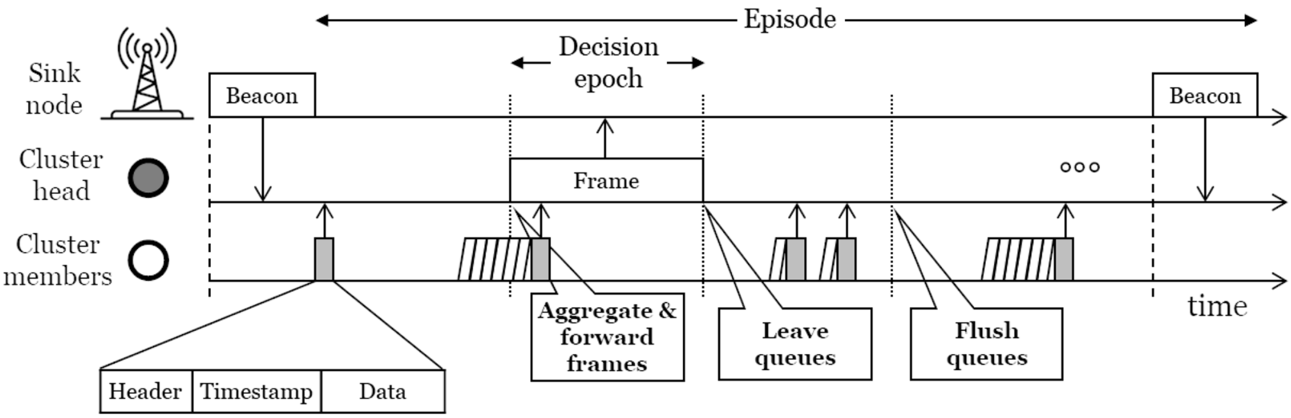


Fig. 2. Timing diagram for DeepAAQM.

the policy network learns using the return values of one episode included in the beacon frame the CH receives.

IV. DRL-Based Active Queue Management

The MDP model is an appropriate mathematical decision-making framework. In this section, we present an MDP model for queue control operation and derive the optimal policy. In addition, one sink node, one CH, and many CMs associated with the CH are assumed hereafter.

A. State Space

We define the state space of a finite set $\mathbf{S}$ as follows:

$$\mathbf{S} = \mathbf{C} \times \mathbf{T} \times \boldsymbol{\Delta}^{\text{AoI}} \times \boldsymbol{\Delta}^{\text{TS}} \times \mathbf{L} \tag{1}$$

where $\mathbf{C}$ is the set of channel conditions, $\mathbf{T}$ is the current time normalized by beacon interval duration, $\boldsymbol{\Delta}^{\text{AoI}}$ and $\boldsymbol{\Delta}^{\text{TS}}$ are arrays of the sets of normalized current AoI value and timestamp at the transmission moment for each CM normalized by beacon interval duration, and $\mathbf{L}$ is a relative location from the head of the queue normalized by queue length.

In this article, we utilize a classical two-state Gilbert–Elliot model to represent wireless channel [28], [29]. There are two states: 1) a good state and 2) a bad state. Based on the Rayleigh fading model, we consider the present state to be a good state when signal-to-noise ratio (SNR) is larger than the threshold value. Otherwise, the channel is said to be in a bad state. With the assumption mentioned above, $\mathbf{C}$, the set of channel conditions is represented by

$$\mathbf{C} = \{0, 1\} \tag{2}$$

where $c(\in \mathbf{C}) = 0$ represents a good state and $c = 1$ be a bad state.

The current time space normalized by beacon interval $\mathbf{T}$ can be represented as follows:

$$\mathbf{T} = \{t \mid 0 \le t \le 1\}. \tag{3}$$

Assuming there are N CMs associated with the CH, $\boldsymbol{\Delta}^{\text{AoI}}$ can be represented as follows:

$$\boldsymbol{\Delta}^{\text{AoI}} = \prod_{i=1}^{N} \boldsymbol{\Delta}_i^{\text{AoI}} \tag{4}$$

where $\boldsymbol{\Delta}_i^{\text{AoI}}$ stands for the set of normalized current AoI value for CM i. Through the normalization process with respect to beacon duration, $\boldsymbol{\Delta}_i^{\text{AoI}}$ is within [0, 1].

The remaining state spaces, $\boldsymbol{\Delta}^{\text{TS}}$ and $\mathbf{L}$ are defined through information on frames entered into the queue. Similar to the definition of $\boldsymbol{\Delta}^{\text{AoI}}$, $\boldsymbol{\Delta}^{\text{TS}}$ and $\mathbf{L}$ can be represented as follows:

$$\boldsymbol{\Delta}^{\text{TS}} = \prod_{i=1}^{N} \boldsymbol{\Delta}_i^{\text{TS}} \tag{5}$$

and

$$\mathbf{L} = \prod_{i=1}^{N} \mathbf{L}_i. \tag{6}$$

Here, $\mathbf{L}_i$ is the relative location for the frame transmitted by CM i which is the closest to the head of queue, and $\boldsymbol{\Delta}_i^{\text{TS}}$ is the normalized timestamp of the corresponding frame.

B. Action Space

Based on the state information in Section IV-A, the CH chooses the action a configured to aggregate and forward buffered frames, flush the queue, or leave the queue based on the MDP model. To do this, we define the action state as follows:

$$\mathbf{A} = \{0, 1, 2\} \tag{7}$$

where 0, 1, and 2 stand for each defined action, respectively. A specific procedure based on Fig. 2 will be stated as follows.

1) $a = 0$ *(Aggregate and Forward Frames):* If this action is taken, one or more frames that are in head of line of the queue are aggregated and forwarded to the sink node and then removed.[4,5] Since most of the time the CH may transmit frames, we assume that the CH has two or more transceivers to receive frames from the associated CMs while transmitting the aggregated frame

[4] The aggregation of frames may vary depending on the applied transmission protocol, but typically, in WLAN standards after IEEE 802.11n, the aggregate MAC service data unit (A-MSDU) or aggregate MAC protocol data unit (A-MSDU) method can be used.

[5] Since we assume frame durations from CM to CH and from CH to sink node are 30 and 270 μs, respectively. Hence, the CH can aggregate up to nine stored frames in this article.

to the sink node. The number of frames that can be aggregated is determined by the length of the frame duration transmitted by the CM and the length of the decision epoch.

2) $a = 1$ *(Flush the Queue):* If the CM decides to flush out the queue, all frames stored in the queue are discarded. When there is little space left in the queue while the frames in the queue are "stale," that is to say, a lot of time has passed since the recorded timestamp, this action may be selected.

3) $a = 2$ *(Leave the Queue):* The CM may select leave, which means the CM just listens to the medium without forwarding or discarding a frame in the queue.

C. State Transition Function

We indicate two arbitrary states, the current state s and the next state s' in $\mathbf{S}$ as

$$s = \{c, t, \delta^{\text{AoI}}, \delta^{\text{TS}}, l\} \tag{8}$$

and

$$s' = \{c', t', \delta^{\text{AoI}\prime}, \delta^{\text{TS}\prime}, l'\} \tag{9}$$

where δ^{AoI}, $\delta^{\text{AoI}\prime}$, δ^{TS}, $\delta^{TS\prime}$, l, and l' are N length-size vectors, and an arbitrary action in $\mathbf{A}$ as $a \in \{0, 1, 2\}$. The components of the state, that are $c, t, \delta^{\text{AoI}}, \delta^{TS}$, and l, belong to $\mathbf{C}, \mathbf{T}, \boldsymbol{\Delta}^{\text{AoI}}, \boldsymbol{\Delta}^{TS}$, and $\mathbf{L}$, as stated in (1), respectively.

The state transition function is the probability of reaching the state s' from the state s when the system does the action a. For ease of understanding, each entry consisting of state s is explained in detail.

The next channel state c' only depends on its current channel state c and the next time t' also does. Meanwhile, since the other components consist of the number of CMs. The next normalized current AoI value $\delta_i^{\text{AoI}\prime}$ depends on the current AoI values δ_i^{AoI}, the timestamp when the frame sent by CM i was created δ_i^{TS}, the channel quality c, current time t, as well as the performed action a. On the other hand, the timestamp of CM i's frame $\delta_i^{TS\prime}$. Next frame location of CM i, l_i' will be change according to its current location l_i and the action, such as $a = 0$ or $a = 1$.

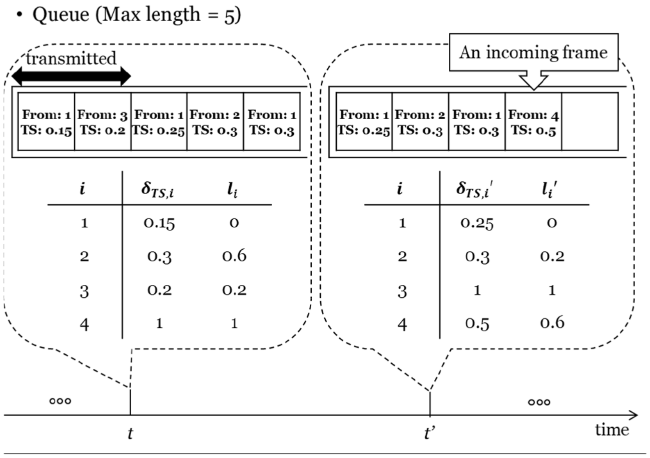
To sum up, the state transition function can be expressed as follows:

$$\Pr\left[s' \mid s, a\right] = \Pr\left[c' \mid c\right] \times \Pr\left[t' \mid t\right] \times \prod_{i=1}^{N} \Pr\left[\delta_i^{\text{AoI}\prime} \mid c, t, \delta_i^{TS}, l_i, a\right] \times \prod_{i=1}^{N} \Pr\left[\delta_i^{TS\prime} \mid \delta_i^{TS}, a\right] \times \prod_{i=1}^{N} \Pr\left[l_i' \mid l_i, a\right]. \tag{10}$$

Looking at each element in (10) one by one, we can define state transition probability regarding the channel state as shown in [29] as follows:

$$\Pr\left[c' \mid c\right] = \begin{cases} \frac{f_D T_p \sqrt{2\pi\rho}}{e^{\rho} - 1}, & \text{if } c' = 1,\ c = 0 \\ 1 - \Pr\left[c' = 1 \mid c = 0\right], & \text{if } c' = 0,\ c = 0 \\ f_D T_p \sqrt{2\pi\rho}, & \text{if } c' = 0,\ c = 1 \\ 1 - \Pr\left[c' = 0 \mid c = 1\right], & \text{if } c' = 1,\ c = 1 \end{cases} \tag{11}$$

where $c = 0$ and $c = 1$ represent a good state and a bad state, respectively, as mentioned in (2), T_p is the state transition duration, which is set to 300 μs and is the same as the duration of decision epoch, ρ is the threshold level normalized to the average SNR that is represented as $\Gamma/\bar{\gamma}$, f_D is the Doppler frequency, defined as $(v/v_c)f_0$, where v is the mobility speed, v_c is the speed of light, and f_0 is the center of the frequency band.



i	$\delta_{TS,i}$	l_i
1	0.15	0
2	0.3	0.6
3	0.2	0.2
4	1	1

i	$\delta_{TS,i}{}'$	$l_i{}'$
1	0.25	0
2	0.3	0.2
3	1	1
4	0.5	0.6

Fig. 3. δ_i^{TS} and l_i during two adjacent epochs.

As time elapses, regardless of the other circumstances, we can define state transition probability for time t as follows:

$$t' = t + \frac{T_p}{T_b} \tag{12}$$

where the normalized current times t, t' belong to $[0, 1]$ and T_b is the beacon duration, which is the length of one episode and is set to 100 ms. When t' becomes 1, an episode ends and the next episode begins.

Next, we define current AoI value state transition probability. Basically, current AoI value of CM i increases proportionally to the current time unless the frame transmitted by the CM i is successfully forwarded to the sink node. Hence, δ_i^{AoI} is stated as

$$\delta_i^{\text{AoI}\prime} = \begin{cases} t - \delta_i^{TS} + \frac{T_p}{T_b}, & \text{if } c = 0,\ l_i \leq \frac{t_s}{t_l},\ a = 0 \\ \delta_i^{\text{AoI}} + \frac{T_p}{T_b}, & \text{otherwise} \end{cases} \tag{13}$$

where t_s and t_l are durations for frame from CM to CH, and from CH to sink node, respectively. In other words, current AoI value of the CM i is updated if buffered frame transmitted by the CM i in the CM's queue is successfully forwarded to the sink node.

When it comes to the information on frames in the queue, there are two substates: δ_i^{TS} and l_i. A CH maintains a table containing CM's ID and corresponding CM's normalized timestamp value, δ_i^{TS} and its relative location, l_i and it is updated every decision epoch. δ_i^{TS} is the value recorded when the frame at the head of the queue is created. If there is not any frame from CM i, δ_i^{TS} equals to 1. In a similar way, l_i lies within $[0, 1)$ in the direction from head to tail of the queue. Otherwise, if there is not any frame from CM i, l_i is equal to 1.

For ease of understanding, we briefly illustrate an example trend of δ_i^{TS} and l_i in Fig. 3 showing two adjacent decision

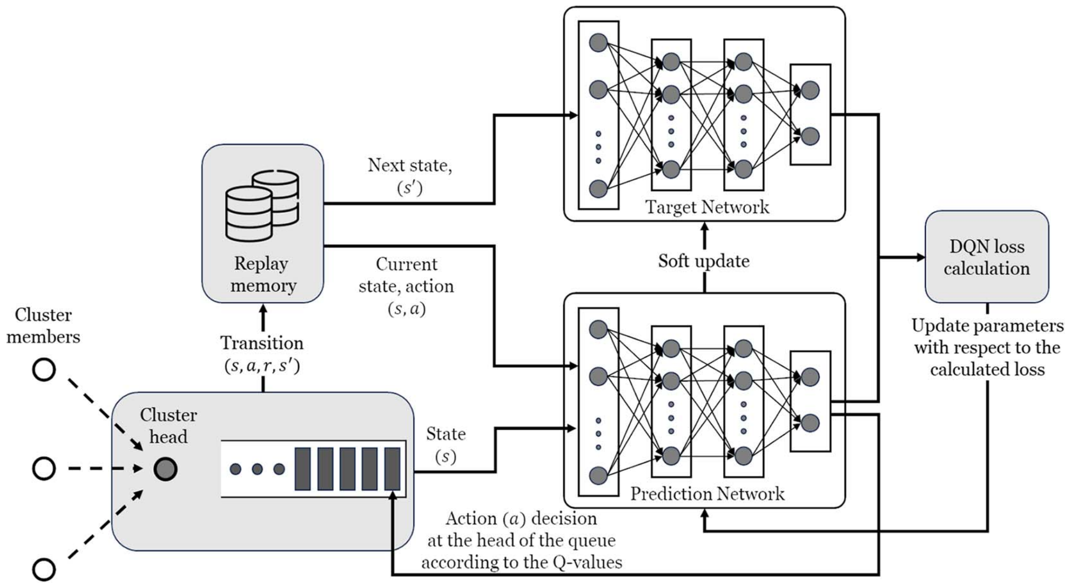


Fig. 4. DeepAAQM architecture with a learning agent and network components.

epochs. The values of δ_i^{TS} and l_i are only relevant for the frame transmitted by CM i which is the closest to the head of the queue. Accordingly, $\delta_{TS,1}$ is 0.15 and l_1 is 0, which means the frame from CM 0 is located at the very head of the queue. When two frames are aggregated and forwarded to the sink node, since the frame from CM 0 is still positioned at the beginning, l'_1 is 0. However, the value of $\delta'_{TS,1}$ would change according to the timestamp of the new frame. In addition, $\delta_{TS,4}$, l_4, $\delta'_{TS,3}$, and l'_3 are all 1 because the corresponding frame does not exist.

D. Reward Function

To define the reward function, we consider the peak AoI values among CMs and their energy consumption. Since we are assuming fixed decision epoch duration, we replace the consumed energy with consumed power hereafter.

In particular, the reward when action a is selected on state s is as follows:

$$R(s, a) = c_{\text{pow}} \cdot R^{\text{pow}}(s, a) + c_{\text{aoi}} \cdot R^{\text{aoi}}(s, a) \tag{14}$$

where $R^{\text{pow}}(s, a)$ and $R^{\text{AoI}}(s, a)$ are power consumed by CH, and the maximum peak AoI value among CMs, respectively, upon the state s and the action a, and c_{pow} and c_{aoi} stand for balance parameters between energy consumption and the maximum peak AoI values.

Subsequently, the power consumed by CM is represented as

$$R^{\text{pow}}(s, a) = \begin{cases} -P_{\text{tx}}, & \text{if} \quad a = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

In (15), P_{tx} can be any parameter, however in this article, we set P_{tx} as 0.280, according to common power model parameters introduced for IEEE 802.11ax [30]. To evaluate the value of the consumed power, we omit the unit, use the number itself, and normalize it by using parameter c_{pow}.

Next, the maximum peak AoI value is represented as

$$R^{\text{AoI}}(s, a) = -\sum_{\forall i} \left(\delta_i^{TS} - \frac{T_{\text{target}}}{T_b} \right)^{+} \tag{16}$$

where t_k is the time of the kth decision epoch and x^+ stands for the ramp function which is defined as follows:

$$x^+ = \begin{cases} x, & \text{if} \quad x \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

A specific peak AoI value may be needed depending on the requirements of different applications. With (14), DeepAAQM tries to satisfy these AoI service requirements while minimizing consumed power.

V. Deep Reinforcement Learning Algorithm

DRL has demonstrated the ability to learn and make optimal decisions in dynamic and uncertain network environments through trial and error, particularly when large state spaces complicate optimization problems. Therefore, we propose using the DRL algorithm to solve the MDP model shown in Section IV.

We adopt a DRL model based on the DQN algorithm with experience replay [31]. When it comes to AoI values within a beacon interval, collected transitions may be correlated and have nonstationary distributions. The experience replay mechanism randomly selects previous transitions, resulting in a more balanced training distribution that accounts for a variety of past behaviors. This section, therefore, describes how DRL applies to action decision problems for the CH.

We depict the DeepAAQM architecture accompanied with network components in Fig. 4. Once a CM gets an opportunity to access the wireless medium, it transmits a frame including a timestamp. The agent then queues the received frame into CH's queue by first-in–first-out policy. The agent also manages a table, named AoI table, to record received frames' transmitter IDs and recorded AoI. One room of the queue and one row of

Algorithm 1 Training Phase of DeepAAQM Using DQN With Experience Replay

```
Initialize prediction network $Q$ with parameter $\theta$
Initialize target network $\hat{Q}$ with parameter $\hat{\theta}$
Initialize replay memory $D$
for each episodes do
    Initialize the state of an agent
    while episode is not done do
        With probability $\epsilon$, select a random action $a$. Otherwise, select $a$ that maximizes $Q(s, a)$
        Agent takes an action $a$
        Agent observes reward $r$ and next state $s'$
        Store transition $(s, a, r, s')$ in the memory $D$
        if $D$ is full then
            Sample $N$ transitions from $D$
            for $n$th transition in the samples do
                $y_i = r_i + \gamma \max_{a' \in A} \hat{Q}(s'_i, a')$
            end for
            Calculate loss $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y_i - Q(s_i, a_i))^2$
            Update $\theta$ in $Q$ by minimizing the loss $\mathcal{L}$ with stochastic gradient descent
            Soft parameter update, $\hat{\theta} = \tau\theta + (1 - \tau)\hat{\theta}$
        end if
    end while
    return $\theta$
end for
```

the AoI table have one-to-one mapping. The agent keeps track of the queue and AoI table and reorganizes them in accordance with the indexes of CMs. They can then be regarded as entries of a state, $\delta_i^{\mathrm{AoI}}(t)$ and $\delta_i^{TS}(t)$, and the state comprises them across CMs. When it comes to the neural network, the input layer consists of $2N + 1$ nodes, as defined in (1). We adopted two hidden layers with $N + 2$ nodes each, including activation functions of ReLU. At the end of the output layer, the action can be determined through the softmax function. The structure of the policy network has been empirically optimized through several simulations. Once the policy is calculated by the policy network, the agent, i.e., the CH, determines the action for the frame of the head of line. Next, a reward can be calculated by means of the chosen action and it is used to update the policy network. The overall algorithm proceeds with the revised neural network parameters in sequence. Algorithm 1 demonstrates how to utilize the DRL algorithm to solve the CH queue management problem.

VI. Performance Analysis Results

For performance analysis, we conduct extensive simulations with Python 3.10 along with gymnasium [32] which is a standard API for reinforcement learning, NumPy [33] which is a Python package for scientific computing, and PyTorch [34] which is an open-source machine learning framework.

We compared DeepAAQM with conventional AQM methods, which are 1) SRED [17] and 2) CoDel [18]. Specifically, SRED algorithm pre-emptively discards frames with a load-dependent probability, as introduced in [17]. Frame drop probability $p_{\mathrm{sred}}(q)$ is set as follows:

TABLE I
Summary of Notations

Description	Value
Hyperparameters	
Number of hidden layers	2
Optimizer	Adam [35]
Learning rate, α	0.001
Loss function	Smooth L1 loss [36]
Threshold for ϵ-greedy	0.1
Activation function	ReLU
Discount rate, γ	1
Replay memory batch size, N	128
Soft parameter update weight, τ	0.005
Network and device parameters	
Beacon interval, T_b	100 msec
Decision epoch, T_p	300 μsec
Frame duration from CM to CH, t_s	30 μsec
Frame duration from CH to sink node, t_l	270 μsec
Simulation duration	10 sec (100 episodes)
Number of CMs	5, 10, 20
CMs' velocity, v	10 km/sec
Target AoI value, T_{target}	20 msec
Queue size in CH	20
Aggregation limit, t_l/t_s	9
Medium access protocol	Slotted ALOHA, CSMA/CA

$$p_{\mathrm{sred}}(q) = \begin{cases} 0, & \text{if} \quad 0 \le q < \frac{1}{6} \\ \frac{p_{\max}}{4}, & \text{if} \quad \frac{1}{6} \le q < \frac{1}{3} \\ p_{\max}, & \text{if} \quad \frac{1}{3} \le q < 1 \end{cases} \tag{18}$$

where q is the ratio of occupied queue amount to maximum queue amount and $p_{\max}$ is set to 0.15.

In CoDel, we set parameters in conformity with https://queue.acm.org/appendices/codel.html, the pseudocode in [18], and with RFC8289 [19]. CoDel uses two key variables: 1) *target* and 2) *interval* and they are set to 100 and 5 ms, respectively. To explain in detail, CoDel measures packet delay each *interval*. If delay is below *target*, no packets are dropped. If delay exceeds the target, however, CoDel drops a packet and adjusts *interval* based on the inverse square root of consecutive drop-mode intervals, with a typical sequence being 100, $100/\sqrt{2}$, and $100/\sqrt{3}$. When delay falls below *target*, CoDel exits drop mode, stops dropping packets, and resets the *interval*.

In this analysis, we cover two well-used wireless medium access protocols: 1) slotted ALOHA and 2) CSMA/CA. For slotted ALOHA, we set the medium access probability as $1/N$, which is known to be an optimal access probability. For CSMA/CA, we use a conventional IEEE 802.11 distributed coordination function (DCF) that employs CSMA/CA with a binary exponential backoff algorithm. Hyperparameters for DRL and network/device parameters are listed in Table I. Underlined values in the table are used in the default option when unmentioned in simulations.

A. Convergency Examination of DeepAAQM

Before analyzing the simulation results, we first want to ensure that the DeepAAQM algorithm is stable by examining whether the return values converge as the episode progresses. Fig. 5 shows the trend of the algorithm's return values in various environments. Although there are some differences, it can be observed that the return values generally converge from

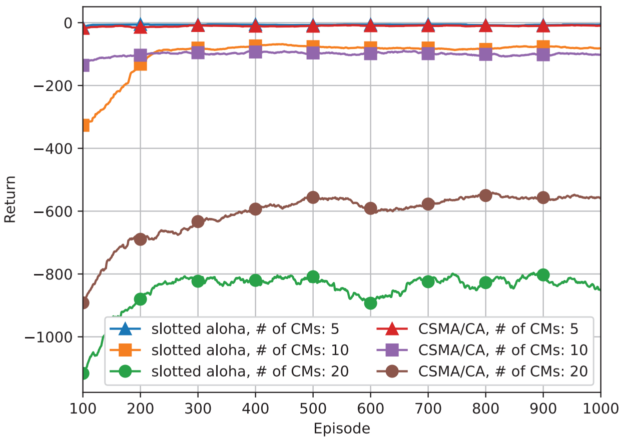
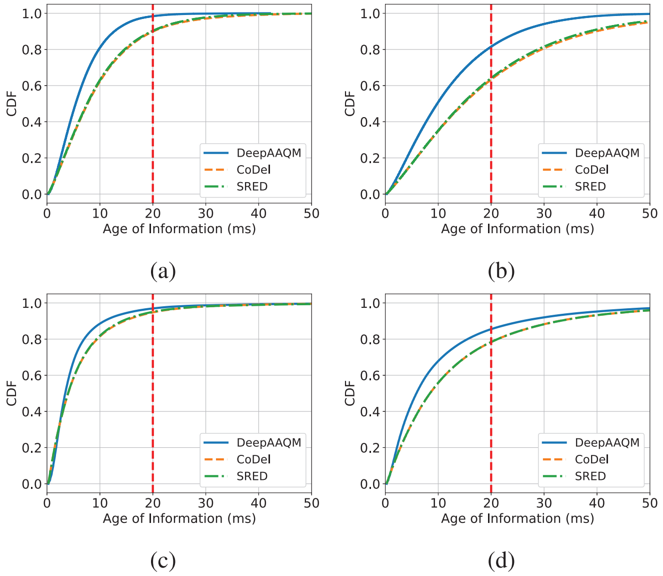


Fig. 5. Trends of DeepAAQM's moving averages of the returns for various numbers of nodes and access algorithms. The values are unweighted running averages of the previous 100 episodes.



(a) (b) (c) (d)

Fig. 6. Cumulative distribution function of AoI values for DeepAAQM and the comparison algorithms in (a) slotted ALOHA with five CMs, (b) slotted ALOHA with ten CMs, (c) CSMA/CA with five CMs, and (d) CSMA/CA with ten CMs.

approximately the 200th episode. Also, considering that the return value converges reliably, it is believed that there is no catastrophic forgetting [37] on DeepAAQM both for slotted ALOHA and CSMA/CA.

On the other hand, as the number of CMs increases, competition for occupying shared wireless medium intensifies. This increases the probability of collisions and thus lowers the probability of a certain CM occupying the channel. Of course, as the number of CMs increases, the probability of individual CMs accessing the media also decreases. Therefore, the return value when the number of CMs is 20 can be expected to be lower than that when the number of CMs is 5 or 10, because the CMs are likely to have large AoI values, and this can be seen in the figure. Additionally, due to the collision avoidance feature of CSMA/CA, we can observe that the return value of CSMA/CA is much higher than that of Slotted ALOHA, especially when the number of CMs is large.

B. AoI Comparison

In this section, the analysis of AoI values over DeepAAQM and the comparison algorithms through Figs. 6 and 7. First, we show CDFs of collected AoI values over 100 individual episodes for each management algorithm in Fig. 6. The experiment is conducted under the numbers of CMs of 5 and 10. T_{target} is 20 ms and is marked with a red dotted vertical line in the figure.

Through Fig. 6(a)–(d), it can be found that DeepAAQM outperforms compared algorithms in terms of minimizing AoI value. Let p_{out} be the ratio of AoI values exceeding T_{target}, it is shown that DeepAAQM reduces p_{out} by 83.7%, 49.0%, 39.9%, and 33.0%, compared to other algorithms in slotted ALOHA with five CMs, slotted ALOHA with ten CMs, CSMA/CA with five CMs, and CSMA/CA with ten CMs, respectively. In addition, the AoI values of CoDel and SRED are almost similar. From these results, we can observe that the reward function using T_{target} was appropriately designed and that the queue management mechanism is working well as intended by this reward function. We can also see that DeepAAQM shows greater performance improvement when using slotted ALOHA than CSMA/CA, and when using a small number of CMs rather than a large number of CMs. What is noteworthy in these figures is that in the situation of CSMA/CA with ten CMs in Fig. 6(b), the performances of all cases decrease significantly, although DeepAAQM still significantly outperforms the comparisons. Due to the nature of the binary exponential backoff used in CSMA/CA, some CMs will likely have very large contention windows, and there is a possibility that some of these devices will rapidly increase the AoI value. The use of DeepAAQM in this situation can speed up frame transmission from these devices with large backoff values through quick resolution of the head of the queue.

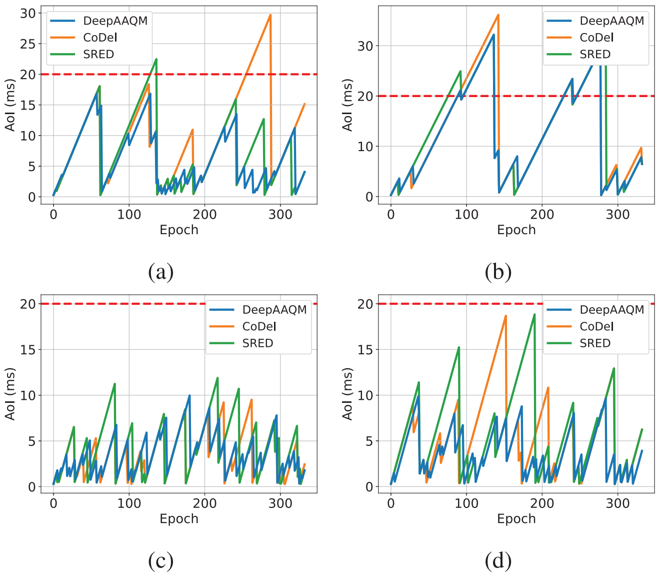
Through Fig. 7(a)–(d), we observe trends of the AoI values of the queue management algorithms for the same CM in the same episode. T_{target} is depicted as a red dotted horizontal line. Intuitively, the trends of ten CMs have larger peak AoI values than those of five CMs. Additionally, in Slotted ALOHA, each CM has the same channel access probability, so the peak AoI value is small, and the sawtooth graph is expressed in a finer form compared to CSMA/CA, accordingly.

C. Consumed Power Comparison

From Fig. 8(a)–(d), power consumptions of DeepAAQM and the comparative algorithms are presented. The consumed power values of each episode are averaged over one episode. Throughout the figures, SRED appears to consistently consume a lot of energy while others do not. The reason for this result can be analyzed as follows. SRED adaptively sets the frame drop probability based on the ratio of occupied frames in a queue, as seen in (18). However, media access algorithms such as slotted ALOHA and CSMA/CA preemptively perform the functionality of the queue management themselves. That is, as the number of nodes increases, a collision occurs and the channel becomes idle in slotted

TABLE II
REPRESENTATIVE PERFORMANCE VALUES FOR 100 TEST EPISODES

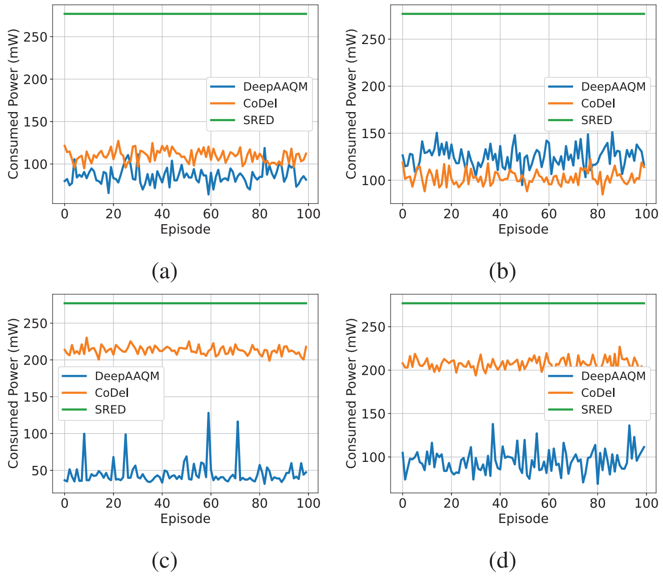
No of CMs = 5	slotted ALOHA			CSMA/CA		
	DeepAAQM	CoDel	SRED	DeepAAQM	CoDel	SRED
Peak AoI (ms)	42.12	63.81	59.94	90.67	90.67	92.82
Average mean AoI (ms)	6.49	9.72	9.51	5.31	6.42	6.40
Average consumed power (mW)	86.20	109.63	276.93	45.56	213.06	276.92
No of CMs = 10	slotted ALOHA			CSMA/CA		
	DeepAAQM	CoDel	SRED	DeepAAQM	CoDel	SRED
Peak AoI (ms)	70.23	99.90	99.90	99.90	99.90	99.90
Average mean AoI (ms)	12.23	18.72	18.34	10.69	13.67	13.68
Average consumed power (mW)	124.88	102.79	276.92	95.35	207.35	276.92



(a) (b)

(c) (d)

Fig. 7. Trends of AoI values of the identical CM for DeepAAQM and comparison algorithms in (a) slotted ALOHA with five CMs, (b) slotted ALOHA with ten CMs, (c) CSMA with five CMs, and (d) CSMA with ten CMs.



(a) (b)

(c) (d)

Fig. 8. Cumulative distribution function of AoIs for DeepAAQM and comparison algorithms in (a) slotted ALOHA with five CMs, (b) slotted ALOHA with ten CMs, (c) CSMA with five CMs, and (d) CSMA with ten CMs.

ALOHA, and a collision avoidance function through binary exponential backoff operates in CSMA/CA. Therefore, in this simulation environment, the ratio of occupied frames in the queue does not increase in slotted ALOHA or CSMA/CA. In environments with very small queue lengths, the frame drop probability of SRED is expected to vary.

What is notable is that DeepAAQM consumes more power than CoDel, as seen in Fig. 8(b). When a large number of CMs compete for channel access through CSMA/CA, the queue is likely to be empty due to extreme contention. So in the CoDel algorithm, it is more likely to be in idle. On the other hand, DeepAAQM attempts to empty the queue as much as possible even when there are a small amount of packets in the queue in order to maximize the reward, as seen in (16), so it is observed that power consumption increased. This can also be supported by the CDF figure of AoI in Fig. 7(b).

D. Summary

In Table II, we show peak AoI, average AoI, and average consumed power calculated from all CMs after 1000 episodes of training. In this scenario, it is shown that DeepAAQM can reduce peak AoI and average mean AoI up to 33.99% and 34.67%, respectively. The average power consumption can be reduced by up to 83.55% compared to SRED and 78.62% compared to CoDel, and in some cases, about 21.49% more power is consumed to reduce the AoI value. According to the results, it was shown that DeepAAQM can satisfy AoI values. DeepAAQM is particularly suitable for services where guaranteeing AoI bounds is crucial, such as disaster emergency communication systems, especially when they have energy limitations.

VII. CONCLUSION

In this article, we proposed DeepAAQM, an AQM algorithm for CH that adaptively selects an action for a head-of-line frame in IoT sensor networks that requires target AoI value and low power consumption. To find out the optimal policy to maximize the reward consisting of the power consumption and the peak AoI value, we formulated an MDP that considers the current channel condition and the current AoI value. Performance analysis demonstrated that DeepAAQM could outperform other comparative algorithms, specifically, maintaining the target AoI value with low power consumption. We then investigate the convergency of DeepAAQM, the trend of AoI, and consumed power. It was shown that DeepAAQM can be adopted in various conditions through intensive simulations.

References

[1] M. T. Lazarescu, "Design of a WSN platform for long-term environmental monitoring for IoT applications," *IEEE J. Emerg. Select. Topics Circuits Syst.*, vol. 3, no. 1, pp. 45–54, Mar. 2013.

[2] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, p. 223.

[4] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," in *Proc. Annu. Commun. Netw. Services Res. Conf.*, 2005, pp. 255–260.

[5] A. K. Dwivedi and A. K. Sharma, "EE-LEACH: Energy enhancement in LEACH using fuzzy logic for homogeneous WSN," *Wireless Pers. Commun.*, vol. 120, pp. 3035–3055, Oct. 2021.

[6] B. Manzoor et al., "Q-LEACH: A new routing protocol for WSNs," *Proc. Comput. Sci.*, vol. 19, pp. 926–931, Jan. 2013.

[7] M. E. Haque et al., "A hybrid approach to enhance the lifespan of WSNs in nuclear power plant monitoring system," *Sci. Rep.*, vol. 12, p. 4381, Dec. 2022.

[8] G. A. Senthil, A. Raaza, and N. Kumar, "Internet of Things energy efficient cluster-based routing using hybrid particle swarm optimization for wireless sensor network," *Wireless Pers. Commun.*, vol. 122, pp. 2603–2619, Feb. 2022.

[9] R. Maheswar, P. Jayarajan, S. Vimalraj, G. Sivagnanam, V. Sivasankaran, and I. S. Amiri, "Energy efficient real time environmental monitoring system using buffer management protocol," in *Proc. Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, 2018, pp. 1–5.

[10] D. Hosahalli and K. G. Srinivas, "Enhanced reinforcement learning assisted dynamic power management model for Internet-of-Things centric wireless sensor network," *IET Commun.*, vol. 14, pp. 3748–3760, Dec. 2020.

[11] H. A. Alwasef, "An energy-efficient buffer management scheme based on data integrity and multivariate data reduction for wireless sensor networks," *J. Control Eng. Appl. Inform.*, vol. 23, no. 3, pp. 53–61, 2021.

[12] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?," in *Proc. IEEE INFOCOM*, 2012, pp. 2731–2735.

[13] Y. Sun, E. Uysal-Biyikoglu, R. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.

[14] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[15] X. Diao, X. Guan, and Y. Cai, "Joint offloading and trajectory optimization for complex status updates in UAV-assisted Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23881–23896, Dec. 2022.

[16] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[17] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proc. IEEE INFOCOM*, vol. 3, 1999, pp. 1346–1355.

[18] K. Nichols and V. Jacobson, "Controlling queue delay," *Commun. ACM*, vol. 55, no. 7, pp. 42–50, Jul. 2012, doi: 10.1145/2209249.2209264.

[19] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled delay active queue management," Internet Eng. Task Force, RFC 8289, Jan. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8289

[20] D. Palma et al., "The Queuepusher: Enabling queue management in openflow," in *Proc. Eur. Workshop Softw. Defined Netw.*, 2014, pp. 125–126.

[21] S. Gu, J. Kim, Y. Kim, C. Moon, and I. Yeom, "Controlled queue management in software-defined networks," in *Proc. Int. Conf. IT Converg. Security (ICITCS)*, 2015, pp. 1–3.

[22] B. Yan, Q. Liu, J. Shen, D. Liang, B. Zhao, and L. Ouyang, "A survey of low-latency transmission strategies in software defined networking," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100386.

[23] Z. Li, Y. Hu, L. Tian, and Z. Lv, "Packet rank-aware active queue management for programmable flow scheduling," *Comput. Netw.*, vol. 225, Apr. 2023, Art. no. 109632.

[24] D. A. AlWahab, G. Gombos, and S. Laki, "On a deep Q-network-based approach for active queue management," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, 2021, pp. 371–376.

[25] M. Kim, M. Jaseemuddin, and A. Anpalagan, "Deep reinforcement learning based active queue management for IoT networks," *J. Netw. Syst. Manag.*, vol. 29, no. 3, p. 34, 2021.

[26] H. Baek, J. Lim, and S. Oh, "Beacon-based slotted ALOHA for wireless networks with large propagation delay," *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2196–2199, Nov. 2013.

[27] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted ALOHA on LoRaWAN-design, analysis, and deployment," *Sensors*, vol. 19, no. 4, p. 838, 2019.

[28] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. Commun.*, vol. 47, no. 11, pp. 1688–1692, Nov. 1999. [Online]. Available: http://ieeexplore.ieee.org/document/803503/

[29] H. Boujemaa, M. B. Said, and M. Siala, "Throughput performance of ARQ and HARQ I schemes over a two-states Markov channel model," in *Proc. IEEE Int. Conf. Electron., Circuits Syst.*, 2005, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/4633423/

[30] S. Merlin et al., "TGax simulation scenarios," IEEE, Piscataway, NJ, USA, document P802.11, 2015. [Online]. Available: https://mentor.ieee.org/802.11/dcn/14/11-14-0980-16-00ax-simulation-scenarios.docx

[31] L.-J. Lin, "Reinforcement learning for robots using neural networks," Ph.D. dissertation, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, Pennsylvania, 1992.

[32] "Gymnasium documentation." Accessed: Sep. 16, 2023. [Online]. Available: https://gymnasium.farama.org

[33] "NumPy." Accessed: Sep. 16, 2023. [Online]. Available: https://numpy.org

[34] "PyTorch." Accessed: Sep. 16, 2023. [Online]. Available: https://pytorch.org

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[36] R. Girshick, "Fast R-CNN," 2015, *arXiv:1504.08083*.

[37] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motiv.*, vol. 24, pp. 109–165, Jan. 1989.

Taewon Song received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, South Korea, in 2010 and 2017, respectively.

In 2020, he joined Soonchunhyang University, Asan, South Korea, where he is currently a Research Assistant Professor with SCH Convergence Science Institute. From 2017 to 2020, he was a Senior Researcher with the Advanced Standard Research and Development Laboratory, LG Electronics, Seoul, where he worked on standardization for wake-up radio and next-generation WLANs. He was a visiting student supported by the BK21+ Project with the University of Florida, Gainesville, FL, USA, from February 2015 to April 2015. From January 2012 to February 2012, he was a Visiting Researcher supported by the National Research Foundation of Korea with the National Institute of Information and Communications Technology, Tokyo, Japan. His current research interests include AI-empowered networking, low-power wake-up radio, and the wireless medium access control protocol of next-generation WLANs.

Yeunwoong Kyung received the B.S. and Ph.D. degrees from the School of Electrical Engineering, Korea University, Seoul, South Korea, in 2011 and 2016, respectively.

He was a Staff Engineer with the Advanced CP Laboratory, Mobile Communications Business, Samsung Electronics, Suwon, South Korea. He is currently an Assistant Professor with the Division of Information and Communication Engineering, Kongju National University, Cheonan, South Korea. His current research interests include mobile core network, mobility management, mobile cloud computing, SDN/NFV, and IoT.