EURASIP Journal on Wireless Communications and Networking

RESEARCH **Open Access**

A Q-learning-based distributed queuing Mac protocol for Internet-of-Things networks

Chien-Min Wu[1*], Yu-Hsiu Chang[1], Cheng-Tai Tsai[1] and Cheng-Chun Hou[1]

*Correspondence: cmwu@nhu.edu.tw

[1] Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan

Abstract
Many previous studies showed that some nodes should be in a sleeping state while the traffic is high. These nodes can wake up periodically to transmit data while they have data to transmit. The system throughput will be increased while each node can change between wake-up and sleeping states periodically. If a node can estimate the active rate and wake up at the optimal time, the collision probability decreases. This study proposes a Q-learning-based distributed queuing medium access control (QL-based DQMAC) protocol for Internet-of-Things (IoT) networks. In the proposed QL-based DQMAC, we derive the optimal number of contention IoT nodes. Each node calculates the active rate by itself through the Q-learning algorithm. Then, each node determines whether it will be active or in sleeping mode in the next contention period according to the active rate. Finding the optimal IoT nodes in each contention period decreases the probability of collision. The energy consumption due to the contention and delay for MAC contention is reduced owing to the lower number of contentions. Protocol comparison with other DQMAC protocols shows that the proposed QL-based DQMAC protocol achieves higher performance in IoT networks.

Keywords: Internet of Things, Medium access control, Q-learning, Distributed queuing, Active rate

1 Introduction

Each device in an Internet of Things (IoT) network has a unique address and connects to other devices. Any device in an IoT network can communicate with other devices, and the communication is not affected by the location, network, or internet service provider. The IoT is a hot topic in the field of communication. For machine-to-machine (M2M) communication of portable devices, communication with a fixed location may not meet the requirements of mobile users. Therefore, communication in mobile IoT is a general application [1, 2]. The communication of applications in IoT can be attained through wireless technology, thereby achieving ubiquitous and seamless communication access [3, 4].

However, the schedule and channel access of data transmission remain a critical problem for efficient system performance in IoT networks. Therefore, the implementation of an efficient MAC protocol is important to achieve high channel utilization.

In addition, the TDMA MAC protocol still presents the collision problem, especially when the traffic increases. To solve the collision problem, some authors proposed a distributed queuing random access protocol (DQRAP) MAC protocol [5, 6]. In DQRAP, the channel can share all contention nodes, thereby detaching the performance from the number of nodes. In DQRAP, the transmission delay does not increase and the system throughput does not decrease. Moreover, DQRAP presents two queues. One is the data transmission queue (DTQ) and the other is the collision resolution queue (CRQ).

In DQRAP, all nodes randomly select the time slot during the contention period. Therefore, the collision problem becomes significant when the traffic load is heavy. In addition, the length of the collision resolution queue increases as the traffic load increases.

In [7, 8], the authors proposed an schedule mechanism of wake-up and sleep to overcome the hidden terminal problem and reduce energy consumption.

In [7], the authors proposed a synchronized sensor-MAC (S-MAC) that is suitable for wireless sensor networks (WSNs). S-MAC assumes that the sensor nodes will be in the sleeping state most of the time. When a sensor node detects the signal, it returns to the wake-up mode. Therefore, the MAC protocol design in WSNs is different from that of traditional wireless networks such as IEEE 802.11. If the propagation delay and fairness are no longer important, energy consumption and self-configuration will be the main goals of the MAC protocol in sensor networks. Therefore, the sensor nodes in the proposed S-MAC will go into the sleeping mode periodically. The nodes in a sensor network are divided into clusters. All these nodes will be self-configured by adaptive listening to obtain the status of the neighbors.

In [9], the authors proposed a non-synchronized B-MAC using an extended preamble. The operation of the sensor nodes in B-MAC is low-power. In addition to operating at lower power, B-MAC achieves collision efficiency, and increases the channel utilization by using the adaptive preamble mechanism. In [8], the authors proposed a low-power MAC (X-MAC) protocol based on a shorter preamble for WSNs. X-MAC also features a low energy consumption and decoupling mechanism of the sleep schedule for both the transmitter and receiver.

In previous MAC protocols, the number of contention nodes for dynamic traffic load is different and unknown. If the number of contention nodes is larger than the network load, the collision probability will increase. If we can control the number of contention nodes within each contention period, then the collision probability will be reduced. Therefore, predicting a suitable number of contention nodes in each contention period in IoT networks is required for dynamic traffic load. In this study, we let some contention nodes go into sleep during the contention period. Only a reduced set of contention nodes is permitted to transmit data. The motivation of this study was to propose an adaptive protocol based on the Q-learning (QL) algorithm to improve the channel utilization under dynamic traffic load in IoT networks.

There are three types of machine-learning (ML) algorithms, namely supervised learning, unsupervised learning, and reinforcement learning. Inputs and outputs are required for supervised learning to map an optimal model. In general, supervised learning involves classification and regression. There are only inputs but no outputs in unsupervised learning. Unsupervised learning determines the rules from the training data during the learning process. Reinforcement learning is an interactive learning process that determines actions from observation of the environment [10].

ML can enable IoT nodes to perform intelligent actions in IoT networks. IoT nodes can access the channel on the basis of an ML algorithm under dynamic traffic load in IoT networks. An reinforcement-learning(RL) algorithm is a possible scheme for ML. Q-learning(QL) is an RL scheme. ML, RL, and QL were used in previous studies to evaluate the system performance in wireless networks [10].

In [11], the authors proposed a novel Markov decision process (MDP) model. The transmission power and transmission probabilities were adaptively adjusted in communication pairs in wireless networks. Consequently, high system performance was achieved.

RL in previous studies was used to address the channel access problem in wireless networks. The IEEE 802.11 backoff scheme was formulated and optimized using MDP [12]. In [12], the authors proposed an RL algorithm to solve the IEEE 802.11 backoff problem. In [13], the authors proposed an energy-efficient channel utilization mechanism based on an RL algorithm for wireless networks. In [14], the authors proposed a unicast and delay-aware MAC protocol based on a QL algorithm in vehicle ad hoc networks.

RL can also decrease the end-to-end delay, energy consumption, and increase the system throughput when applied to wireless networks. Furthermore, in other environments, ML mechanisms have been widely used to solve problems without prior knowledge.

In this study, a QL-based MAC protocol for IoT networks is proposed to improve the system performance. In this QL-based MAC protocol, the frame length of the contention period is adaptive and regulated by the QL algorithm to ensure the quality of service of the system in an IoT network.

The main contributions of the proposed QL-based DQMAC protocol are as follows:

1. A Q-learning-based distributed queuing MAC protocol is proposed for energy-efficient delay-aware IoT networks.
2. The proposed protocol is applied to reduce the collision probability of distributed queuing MAC protocols in IoT networks.
3. The proposed protocol is applied to decrease end-to-end delay, reduce energy consumption, and increase the system throughput for MAC contention.

The remainder of this paper is organized as follows. Methods/Experimental is introduced in Sect. 2. Related work is introduced in Sect. 3. The proposed Q-learning algorithm is introduced in Sect. 4. The proposed QL-based DQMAC protocol is introduced in Sect. 5, and detailed principles and steps of QL-based DQMAC are introduced in Sect. 6. The results and discussion are discussed in Sect. 7, and Sect. 8 presents our conclusions.

2 Methods/experimental

This section presents the simulation results for the QL-based DQMAC protocol for an IoT network. The programming language C was used to complete the simulation. We ran our simulation programs on a computer with a single CPU (Intel Core i7-9700K) and

GPU (NVIDIA Quadro RTX 4000). We used Keras as the learning platform to train the neural networks [15].

The data transmission time and propagation delay for wireless transmission, reception, and listening for each node were measured. The energy consumption for each node was calculated multiple times under the wireless operation. The energy consumptions of transmission, reception, and sleeping mode according to the radio-transceiver datasheet are 13.5 mW, 24.75 mW and 15 μW, respectively. Here, the energy consumption of reception and listening in the radio-transceiver model is the same [7].

The difference between QL-based DQMAC and traditional DQMAC protocols is the design of the contention period. For the QL-based DQMAC protocol, the optimal number of contentions in each contention period is calculated based on Q-learning. In addition, part of the contention nodes before entering the beacon interval go into the sleeping state. All the contention nodes in a beacon interval for traditional DQMAC go to the contention period and then to DTQ or CRQ.

In traditional DQ MAC, the slot selection of the *SAR* control frame in the contention period is random. Each node randomly selects one slot from the *SAR* contention field. No sleeping mechanism and no optimal number of contention nodes concept in the traditional DQMAC protocol are executed. Therefore, the collision probability of the traditional DQ MAC is higher than that of the QL-based DQMAC.

Thus, the end-to-end delay of the traditional DQMAC protocol will increase for a high collision probability resulting from heavy traffic load. In addition, the system performance will be decreased owing to the high collision probability. In this study, the proposed model incorporates a control channel. The data transmission for each node is achieved by the exchange of control frames in the control channel.

In addition, owing to the time-varying characteristics of traffic loads, traditional DQMAC result in low-resource transmission efficiency.

The main difference between QL-based DQMAC and traditional DQMAC is the use of QL learning and a sleeping mechanism. Therefore, we compared the performance of QL-based DQMAC with that of the traditional DQMAC scheme. The transmission rate of the channel was 2 Mbps. The frame length of each contention period was 3, and the system throughput was maximized in the DQ mechanism [5]. The simulation parameters of the proposed QL-based DQMAC protocol are listed in Table 1.

In this simulation, 25 different topologies were created using 25 different seeds. The presented results are the average of the simulation results for 25 seeds. System throughput, end-to-end delay, and energy consumption constituted the metrics of performance evaluation.

A simple energy model based on basic tasks or activities performed by nodes in an IoT network is proposed. All energy components that contributed to the overall energy consumption in the active mode are considered in this model. First, the node is turned on at the activation time (t_{ON}). A switching time (t_{SW}) is then required to change the state before sending the packets to the medium. Here, the node initially uses the time of (t_{CSMA}) to execute the carrier sense multiple access (CSMA) mechanism. Next, the node spends the transmission time (t_{TX})sending messages. After the transmission of the packet is completed, the node requires a switching time (t_{SW}) to enter the inactive state (t_{inActive}) and changes the task again. Furthermore, the node requires switching time

Table 1 Parameters for Our Proposed QL-based DQMAC Scheme

Simulation time	10,000 s
Number of IoT nodes	100
Bounded region	200 m × 200 m
Channel data rate	2 Mbps
Number of contention nodes	$3, 4, 5, \cdots, 49$ and 50
Number of *SAR* slots for one contention period	3
Variation in the number of contention nodes (d)	1
Power consumption for transmission	24.75 mW
Power consumption for reception	13.5 mW
Power consumption for listening	13.5 mW
Power consumption for sleeping	15 μW
Discount factor (γ)	0.9
Learning rate (α)	0.1

(t_{SW}) to start receiving messages with reception time (t_{RX}). The number of packet transmissions and receptions were equal. Finally, the node shuts down with a shutdown time (t_{OFF}). This process measured the energy consumption of each active node in the IoT network. Based on the tasks and the time required for nodes to perform them, which corresponds to a given voltage and current, the total energy consumed by each activity in the IoT network can be obtained according to the previous model [16].

3 Related work

S-MAC can reduce the energy consumption and support network scalability by overcoming collision problems in sensor networks. Sources of energy consumption in sensor networks may be monitoring, collision among nodes, overhearing, and control overhead for successful transmission. S-MAC uses periodic wake-up and sleep to reduce energy consumption owing to the above problems.

In Fig. 1, each node sleeps for a period of time and then wakes up to monitor whether there is a neighboring node to communicate with it. After the monitor operation, the sensor nodes return to the sleep mode. All nodes in the sensor network will go over wake-up and sleep cycles. The node turns off the power during the sleep cycle and sets a start time point to return to the awake state. To reduce the control overhead of the communication among nodes, the duration of the wake-up and sleeping periods is fixed. The sensor nodes can thus be synchronized. However, all the nodes in a sensor network will be switched off during the sleeping period because of the fixed length of the wake-up and sleeping periods. This is not suitable for IoT networks [7].

In Fig. 2, an asynchronous extended preamble of B-MAC is proposed. The preamble is longer than the length of the sleeping period for other nodes. When a sender wants to transmit data, it must first send a leading preamble to the receiver. This leading

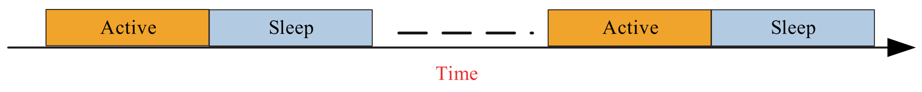

Fig. 1 The S-MAC protocol

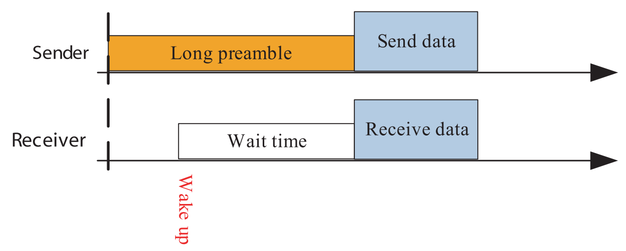

Fig. 2 The B-MAC protocol

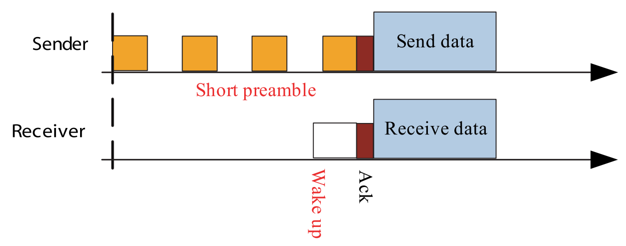

Fig. 3 The X-MAC protocol

preamble increases the wake-up period of the receiver. Thus, the receiver node can receive the leading preamble and communicate with the sender. The one-hop neighbors will receive the preamble and judge whether the destination node belongs to itself. If the destination is not a node, the node will return to the sleeping mode [9].

In [8], the authors proposed a short preamble MAC (X-MAC) protocol to improve the extended preamble of B-MAC. X-MAC can reduce unnecessary energy consumption by reducing the monitoring time to one-hop neighboring nodes. In Fig. 3, if the sender wants to transmit data to the receiver, the sender will send a short but inconsistent preamble until the sender receives an ACK from the receiver. Although X-MAC reduces unnecessary energy waste during node sleep, it also increases the control consumption because the leading symbols are used more often. The advantage of X-MAC is that the sensor nodes can determine the wake-up time by itself.

Distributed queuing features stable performance, and nearly optimal channel access rate, propagation delay, and energy consumption. There are two queues in distributed queuing. One is the contention resolution queue (CRQ), and the other is the data transmission queue (DTQ). When a collision occurs, any collision node enters the CRQ, whereas the successful nodes enter the DTQ. In Fig. 4a–c, six nodes perform the so-called tree-splitting algorithm. Each node sends an access request sequence (ARS) control frame in the selected slot. The successful nodes are assigned to the DTQ and then transmit data. The nodes in which collisions occur will go into the CRQ sequentially. In Fig. 4, nodes $d1$, $d2$ and $d3$ send the ARS control frame at slot 1 of frame 1 and nodes $d4$ and $d6$ send the ARS control frame at slot 3 of frame 1. Therefore, $d1$, $d2$, $d3$, $d4$ and $d6$ will enter the CRQ. Only node $d5$ sends the ARS control frame at slot 2 of frame 1. Thus, $d5$ successfully enters the DTQ of frame 2. The collision nodes will go into the CRQ sequentially, and then into the queue step by step. The successful nodes sequentially enter the DTQ after queuing resolution. All the nodes transmit data successfully owing to the distributed queuing mechanism [17].

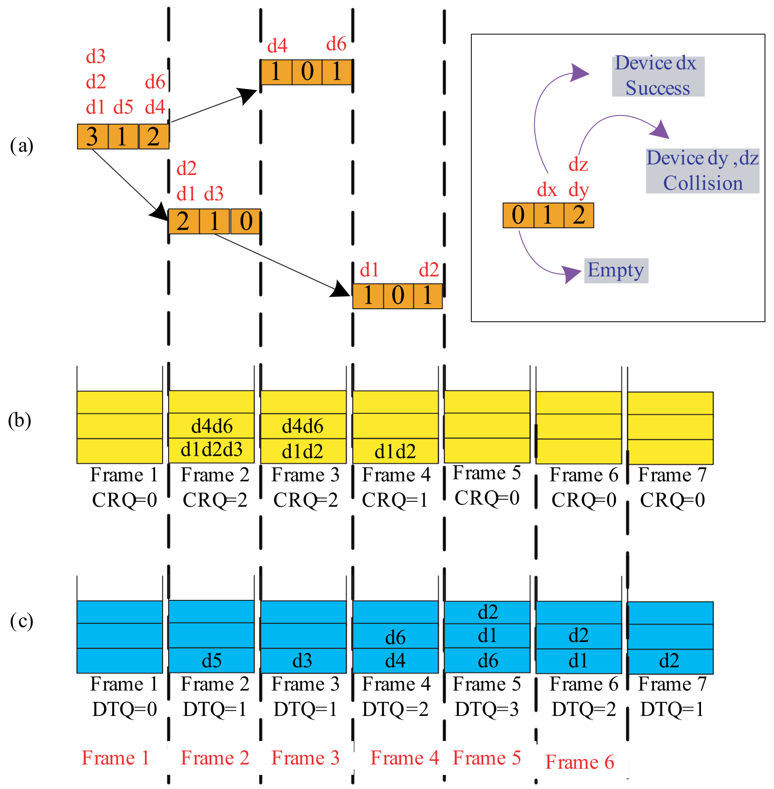

Fig. 4 The distributed queuing MAC protocol

In [18], the authors proposed a QL-based cooperative algorithm to achieve high channel utilization and low energy consumption in wireless networks. In [19], the authors proposed an ML-based algorithm to improve the channel utilization efficiency for opportunistic spectrum access in wireless networks. The ML-based algorithm does not require prior characteristics of wireless networks. The ML-based scheme can achieve high system performance by learning from observation of wireless networks.

In [20], the authors proposed an improved Sarsa mechanism named as expected Sarsa. The updated target was used by the average value, and eligibility traces were introduced for expected Sarsa in wireless communication networks. All possible actions were averaged as the updated target. The historical access recording of each state-action pair was applied by the eligibility trace. The authors demonstrated that expected Sarsa can improve the convergence rate and learning efficiency.

In [21], the authors proposed a deep-reinforcement learning multiple access (DLMA) scheme. This DLMA considers the time-slot sharing problem in multiple wireless networks and learns the experience from observations of state-action-reward without pre-knowledge of the co-existing networks.

System lifetime is a key metric for obtaining high network performance in WSNs. In [22], the authors proposed a QL-based MAC protocol (QL-MAC) that adaptively schedules the sleep/wake-up time of sensor nodes according to the traffic load. An efficient sleep/wake-up mechanism will reduce the energy consumption by using idle listening and overhearing.

Table 2 Comparing the advantages and disadvantages of the proposed QL-based DQMAC and related protocols

	Advantage	**Disadvantage**
Our QL-based MAC	Optimal nodes in each contention achieved	Slow convergence in large environments
DQRAP [5, 6]	Transmission delay does not increase.	Unsuitable for large networks
S-MAC [7]	Preserving energy	Time synchronization is required.
X-MAC [8]	Low power listening	The receiver node needs to remain awake.
B-MAC [9]	Can be scaled to a large network	Unable to provide multi-packet mechanisms
ML-based MAC [19]	Does not need prior characteristics of the node	Unsuitable for large networks
Sarsa [20]	Model-free algorithm	Slow convergence in large environments
DLMA [21]	Faster convergence and more robust	High computational cost
QL-MAC [22]	Adaptively schedule according to the traffic load	Increase energy consumption
RL-MAC [23]	Individual traffic load for each node	The trade-off for latency and energy efficiency

In [23], the authors proposed an RL-based MAC (RL-MAC) protocol for WSNs. For an efficient schedule, the sleep/wake-up period minimizes the energy consumption and optimizes the energy utilization. Energy utilization was optimized through adaptive duty cycles. Previous protocols determine the sleep/wake-up duty cycles according to the traffic node itself. RL-MAC schedules the sleep/wake-up period based on RL. The comparing the advantages and disadvantages of the proposed QL-based DQMAC and related protocols is listed in Table 2.

This study proposes novel concepts using a QL-based DQMAC protocol for IoT networks. First, the proposed model can achieve a higher throughput, lower delay, and lower energy consumption in IoT networks. Improved system performance is achieved by the QL-based distributed queuing MAC. Second, a decrease in transmission delay is achieved by the QL-based DQMAC protocol through control of the number of contention nodes. An increase in the system throughput is also achieved.

4 Q-learning

4.1 Q-learning in Markovian environments

If the estimation of the number of contention nodes can be achieved under a fixed frame length of the contention period, then the collision probability will decrease. When the number of contention nodes is large under a fixed frame length, the collision probability increases, and the packet propagation delay increases as well. Thus, the channel utilization of IoT networks is low. Therefore, the number of contention nodes will be limited and suitable for a fixed frame length.

However, the traffic load is dynamic in IoT networks, and the number of contention nodes is also dynamic. The QL algorithm does not require a prior knowledge of the environment's characteristics. Therefore, the QL algorithm is suitable for estimating the number of contention nodes. QL can use the ongoing interaction between IoT nodes and the environment, and then estimates the optimal number of contention nodes to increase the system performance.

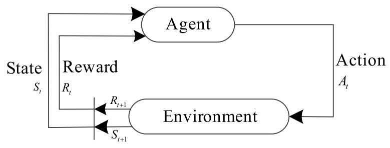

Fig. 5 The agent-environment interaction in a Markov decision process

The QL algorithm does not require complex computational capability from MAC controllers and has a low communication overhead for IoT network nodes. In wireless networks, the acknowledgment of reception is applied to verify the reliability of unicast communication.

Most contention-based MAC protocols have this function to confirm whether the packet is received by the receiver. Therefore, contention-based MAC protocols have a high overhead owing to the acknowledgement scheme [24].

In Fig. 5, the MDP defines a problem under a simple and direct outline. Interaction learning is used to achieve this goal. The number of elements for states (S), actions (A), and rewards (R) in the MDP is finite.

R_t and S_t are random discrete probability distribution random variables. R_t and S_t are also dependent on the previous state and action. In fact, all previous states and actions have a probability under the specified values for all $s', s \in S, r \in R$ and $a \in A(s)$ [25]:

$$p(s', r|s, a) \doteq \Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a), \tag{1}$$

with

$$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1, \quad s \in S, a \in A(s), \tag{2}$$

where p is the dynamic function and is defined in the MDP and $p : S \times R \times S \times A \longrightarrow [0, 1]$ is an ordinary deterministic function with four arguments.

The policy is a possible action that is selected by one state in the IoT environment.

If the agent follows policy π at time step t, then $\pi(a|s)$ is the probability of $A_t = a$ under $S_t = s$.

$\pi(a|s)$ is the probability that $A_t = A$ under $S_t = s$ if the agent follows policy π at time t. The MDP is solved using an RL mechanism without requiring complete information. The agent, environment, policy, reward, and Q-value function constitute the elements of the QL system.

4.2 Q-learning function

A policy is an executed action in specified states within an IoT environment. A policy is generally a function or lookup table. In fact, a policy is the core of an RL agent. The policy determines the behavior, which is sufficient. In addition, the policy for each action may also be determined by the stochastic or specified probabilities.

The reward defines the purpose of the RL mechanism. The reward is a single number that is sent to the agent in the RL IoT environment. The only purpose of the agent is to maximize the reward. If the reward under the policy selection of action results is low, then the other action will be selected and the policy will be changed in the future for that situation.

In contrast, the reward under the policy selection of action results is high, and the Q-value creates a good signal for that situation. Therefore, the Q-value is the cumulative reward of a state. The action will be decided by this Q-value and will be performed by the agent in the future [25]. The number of successful transmissions of time slots will be reduced, and the number of failed transmissions will be increased under high collision probability in IoT networks.

In an IoT environment, a greater channel collision probability takes place when the number of time slots with successful transmission is reduced, and the number of time slots with failed transmission is increased. Here, we consider two items to calculate the reward. One is the number of contention periods for each beacon interval. Another item is the number of contentions for each node until the slot is successfully accessed. A higher channel collision probability indicates that the number of contention periods will be higher than that of lower channel collision probability. Thus, the reward is lower and may be negative. In contrast, the reward is higher under a lower channel collision probability. The additional reward is obtained by the Q-value estimation. The agent selects the optimal action based on the Q-value. The reward is zero, while the Q-value is absent. The agent decides the optimal action in accordance with the information provided by the Q-value function. Then, the optimal action maximizes the Q-value function. In the long term, the greatest reward is obtained through a series of actions. In addition, maximizing an individual reward is not necessary for an agent [10].

4.3 Q-learning algorithm

A $Q(S_t, A_t)$ table is maintained by the agent in QL. The state S_t of the MDP is observed by the agent in an IoT network for $t = 1, 2, 3, ...$ The agent will selects an action A_t from the set of actions (A). The agent receives a reward $R(t)$ and then observes the next state S_{t+1} after the action A_t. The sequence of events creates the learning experience $(S_t, A_t, R_(t), S_{t+1})$ of the agent. The sequence of events under the (S_t, A_t) will be updated in the Q-table according to the QL function (3).

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{3}$$

An action based on the state S_t will be selected by the agent. The maximum Q-value can be determined for the next state S_{t+1} according to the action A_t. Then the maximum Q-value updates the current Q-value.

The preferred value range of the discount factor γ is $0 < \gamma < 1$. The learning speed is defined as the learning rate α, with $0 < \alpha < 1$.

The discount factor models the importance of rewards in the future. The agent ignores the future reward and only considers the current reward when the discount factor γ is set to 0. By contrast, the agent attempts to obtain a high long-term reward when the discount factor γ is set to 1. In general, the discount factor γ is set between 0.6 and 0.99 [24], and is considered as a part of the problem.

The extent to which the new message overrides the old message is determined by the value of the learning rate. When the estimated value is modified, the estimated speed is also controlled by the learning rate. In general, an increase in the learning rate increases the estimated speed. A decrease in the learning rate will decrease the estimated speed. There is no new learning message when the learning rate α is set at 0. By contrast, only the most recent message must be considered when the learning rate α is set to 1.

5 QL-based DQMAC protocol

The proposed IoT network architecture is shown in Fig. 6. There are two types of nodes in IoT networks, namely the IoT cluster head and IoT sensor nodes. The cluster head is powerful and provides a reliable source of energy. An IoT network can have many clusters, and each cluster has its own cluster head. Large IoT networks may have many clusters. These clusters create a multihop environment among nodes. However, in this study, we focused only on one cluster and a single-hop environment by applying the QL mechanism to an IoT network.

In the proposed QL-based DQMAC protocol, there are one control channel for node contention and data transmission.

The control channel of an IoT network is shown in Fig. 7. The time is divided into beacon intervals. Each beacon interval includes the sensing period, contention periods, and data transmission period. Each contention period has slots for slot access request (*SAR*). First, each IoT node wants to send data using a selected one time slot in the contention period. The cluster head receives all *SAR* control frames in the single-hop IoT network. After an *SAR*, the cluster head broadcasts a slot access confirm (*SAC*) control frame to all cluster members, and then the successful sensor nodes perform data transmission.

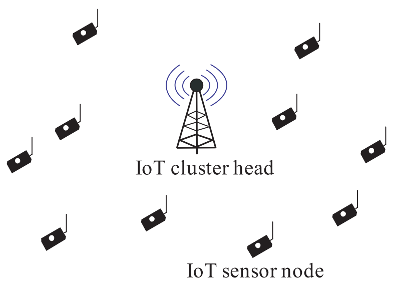

Fig. 6 The architecture of IoT networks

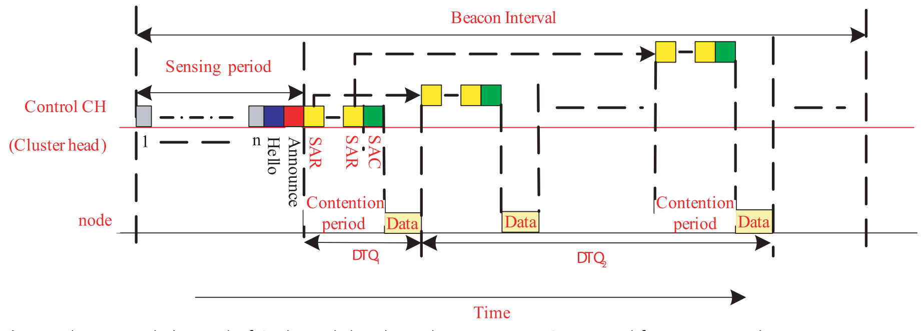

Fig. 7 The control channel of QL-based distributed queuing MAC protocol for IoT networks

In this study, the optimal number of contention nodes within the contention period in IoT networks according to the dynamic traffic rate is achieved by the QL mechanism. In Fig. 5, the agent is a learner and a decision IoT node. Each IoT node must interact with other IoT nodes. The agent selects the optimal action and creates a new situation. Then, the environment will make it suitable for the new situation and create a reward. The agent selects the optimal action to maximize the reward. The procedure is repeated until the maximized reward is achieved.

The agent receives the state information of the environment at each time step t, $S_t \in S$. Then, the agent selects an action, such that $A_t \in A$. After one t under A_t, the agent receives a reward, $R_t \in R$. Then, the agent changes to the new state, S_{t+1} [25].

In the proposed model of QL-DQ MAC protocol, the IoT node is the agent. The state, S_t, is the optimal contention node of the contention period at time t. The action, A_t, is the step size for adjusting the number of the contention nodes in the contention period. The step size may be plus d contention nodes or minus d contention nodes under the current number of contention nodes. In addition, the step size may also be zero and then the next number of the contention nodes is the same as the number of contention nodes. Note that d is an integer. The reward, R_t, depends on the contention results of the contention period at time t.

Therefore, the interactions between the IoT node (agent) and the environment at time step t are as follows [20]:

1. The IoT node observes the environment in the IoT network and obtains the current number of contention nodes, S_t, after one beacon interval.
2. The IoT node obtains the active rate in the next contention period by itself and then determines the next action A_t.
3. The IoT node applies the selected action A_t in the MAC protocol. After one time step, the IoT node obtains the feedback reward R_{t+1} of the IoT network.
4. The IoT node moves from the state S_t to the new state S_{t+1}.

5.1 Control channel descriptions

For the traditional DQ mechanism [5], all nodes enter the contention period by randomly selecting a time slot. The collision increases as the traffic rate increases. Consequently, the propagation delay is also increased. In fact, the maximum number of successful nodes equals the number of contention slots. Therefore, the number of contention nodes will not be larger than the number of slots in each contention period. To achieve the above concept, the specified contention nodes will enter the sleeping state, with a greater number of contention nodes than contention slots. The rest of contention nodes will go into the sleeping state. The optimal number of contention nodes will enter the contention period.

However, what is the optimal number of contention nodes in each contention period? How many contention nodes should go into the sleeping state? For the proposed QL-based DQMAC, the optimal number of contention nodes is calculated using the QL algorithm. Then, the active rate for all the contention nodes is calculated. The QL algorithm for the proposed QL-based DQMAC is used in DQMAC to determine the optimal number of contention nodes. When the optimal number of contention nodes is determined, the probability of collision among contention nodes is decreased, the system throughput will increase, and the propagation delay will decrease. The energy consumption is also decreased by the proposed QL-based DQMAC.

Next, we describe the scheme of QL-based DQMAC, which determines the optimal number of contention nodes in each contention period by using the QL algorithm to improve the distributed queuing MAC protocol.

When one node wants to send data, it sends the *SAR* control frame during the contention period. The cluster head will broadcast the *SAC* control frame when the contention period ends. The *SAC* contains detailed contention information during the contention period. Thus, the successful nodes will send the data and the collision nodes will continue to perform the QL-based DQMAC sequentially. Figure 7 shows a QL-based DQMAC protocol control channel for IoT networks. The two phases of QL-based DQMAC are as follows:

- Sensing period: The decision about whether the licensed channel can be used or not is made according to the sensing success probability. The IoT node senses the licensed channel; its sensing success probability is larger than the sensing threshold. The previous three-channel status will be used to calculate the sensing success probability. *Hello* contains the channel-sensing status and the selected control channel for cluster head. *Announce* is sent by a new node when it wants to *JOIN* or to *LEAVE* a cluster.
- Contention period: This period involves the exchange of *SAR*. In addition, there is a *SAC* for the cluster head to announce the IoT nodes that win the contention and assigned slots in the contention period.

5.2 Contention period descriptions

Each beacon interval includes the sensing period, contention period, and data. Each contention period contains the following control frames:

- *SAR* contains the following fields: DCC_{id}, $Head_{id}$, IoT_{snd}, IoT_{rcv}, NAV_{data}, and $SLOT_{id}$. DCC_{id} denotes the ID for the control channel, $Head_{id}$ denotes the ID of the cluster head, IoT_{snd} denotes the IoT node that sends an *SAR*, IoT_{rcv} denotes the IoT node that receives an *SAR*, NAV_{data} denotes the duration of transmitting data for the IoT node sender, and $SLOT_{id}$ denotes the selected time slot in the contention period for the IoT node sender.
- *SAC* contains the following fields: DCC_{id}, $Head_{id}$, m, N_{avg}, N^{i}_{suc}, $Slot^{1}_{nodeID}$, $\cdots$, $Slot^{n}_{nodeID}$. DCC_{id} denotes the ID for the control channel, $Head_{id}$ denotes the ID of the cluster head, m denotes the frame length of the contention period, N_{avg} denotes the average number of contention nodes in one beacon interval, N^{i}_{suc} denotes the number of successful contention nodes in the contention period

i, and $Slot_{nodeID}^{i}$ denotes the time slot i selected successfully in the contention period for the IoT node sender *nodeID*.

5.3 Active and sleep

In [7], the authors proposed an S-MAC for WSNs. In S-MAC, the sensor node switches between the sleeping and active states. Therefore, the sensor node will remain in a sleeping state for a long time until an active message is detected. Furthermore, the sensor nodes in S-MAC will enter the sleeping state periodically, thereby reducing the energy consumption.

In the distributed queuing MAC protocol, each node sends a request control frame in the selected time slot. The successful node enters the DTQ and then transmits data. The collision node enters the CRQ. Finally, all the nodes successfully transmit their data according to the distributed queuing mechanism.

For a distributed queuing MAC protocol, the propagation delay will be longer when the traffic load is heavy. Moreover, collisions will be frequent under heavy traffic loads. Consequently, the energy consumption is increased when the traffic load is increased. Therefore, decreasing the collision is an important issue when using a distributed queuing MAC protocol for IoT networks.

In QL-based DQMAC, the specified number of IoT nodes will be in a sleeping state in each beacon interval while the distributed queuing mechanism is executed. This specified number of IoT nodes will go into sleep mode according to the Q-learning mechanism.

Then the number of contention IoT nodes in contention period i is calculated as follows:

$$N_{\text{con}}^{i} = \begin{cases} N_{\text{avg}} - \sum_{j=1}^{i-1} N_{\text{suc}}^{j} & 2 \leq i \leq n; \\ N_{\text{avg}} & i = 1. \end{cases} \tag{4}$$

The active rate of the number of contention nodes in contention period i is defined as follows:

$$\text{Rate}_{\text{active}} = 1 - \frac{N_{\text{sleep}}}{N_{\text{con}}^{i}} = 1 - \frac{N_{\text{con}}^{i} - X}{N_{\text{con}}^{i}} \tag{5}$$

Each contention node determines itself whether to be active or in sleeping state in the next contention period according to the active rate.

N_{sleep} denotes the number of sleeping nodes during the contention period, and X denotes the intended optimal number of nodes in the contention period.

For a distributed queuing MAC protocol, the IoT nodes will be successful and then transmit data sequentially according to the resolution mechanism. Therefore, N_{avg} and N_{sleep} are variable numbers with time according to the resolution mechanism.

In the proposed QL-based DQMAC protocol, the number of contention IoT nodes in the next contention period is determined. If such number is large, then the distributed queuing mechanism is preferred. If the number is small, then all the contention nodes will enter the same contention period. In general, the optimal contention nodes cannot increase the number of time slots in the contention period, that is, $X \leq m$.

All the sleeping nodes that want to transmit data will wake up in the next contention period. The new sleeping nodes are set again during the next contention period. All contention nodes in the next contention period include the fail contention node and the sleeping nodes in the previous contention period. The active rate is calculated by the IoT node itself according to Eq. (5) after one contention period. Accordingly, the IoT node will know whether it will be in active or sleep mode in the next contention period.

The executed condition of the distributed queuing mechanism is used to determine which condition is suitable to use the distributed queuing mechanism. Here, the executed condition of the distributed queuing mechanism is defined as follows:

$$\mathrm{DQ}_{\mathrm{condition}} = X \tag{6}$$

If the number of contention nodes is greater than $DQ_{condition}$, then the above distributed queuing mechanism is executed. Otherwise, all the contention nodes will enter the next contention period.

6 Detailed principles of, and steps in, QL-based DQMAC protocol

6.1 Action selection

The optimal number of contention nodes is adapted prior to each beacon interval by performing one of the following actions:

$$X_{t+1} \longleftarrow X_t, a \in (X_t - d, X_t, X_t + d), \tag{7}$$

where X_t is the optimal number of contention nodes at time step t, and d denotes the variation in the number of contention nodes.

RL is different from supervised and unsupervised learning, which are strategies currently being broadly investigated in the field of machine learning. One challenge of RL is the trade-off between exploration and exploitation. To obtain the reward, an RL agent must take one preferred action. This preferred action results from the fact it was effective in generating rewards in past situations.

However, to discover such preferred actions, the agent must not attempt previous actions that did not generate rewards, and must exploit experiences that can be used to obtain rewards. However, the agent must also explore potential outcomes to identify better actions for future situations. As such, neither exploration nor exploitation can be carried out exclusively, which results in a dilemma [25].

The simplest action selection rule is used to select the action with the greatest estimated value. If there is more than one “greedy” action (an action that presents the maximum estimated value), then the action is randomly selected. The greedy-action selection method can be expressed as [25]

$$\pi(s) \doteq \underset{a}{\operatorname{argmax}}\, Q(s, a), \tag{8}$$

where argmax_a denotes the action a for which the expression is maximized.

6.2 Convergence requirements

The convergence of the QL algorithm is based on all actions that are repeatedly sampled in all states. Furthermore, the action values are indicated discretely, with an optimal action-value probability of 1 when the QL algorithm converges [26].

The current state of knowledge is used to maximize the immediate reward through greedy-action selection. However, greedy action does not require additional sampling time to determine whether better options can be selected. Thus, greedy-action selection behaves in most cases with a small probability ε.

The ε-greedy method is defined such that the greedy action is not randomly selected from all actions with the same probability. Instead, it is independently selected according to the action-value estimates. As the number of steps increases, the number of samplings for each action approaches infinity. $Q_t(a)$ is guaranteed to converge to $q^*(a)$, which implies that the converged probability of selecting an optimal action will be greater than $1 - \varepsilon$.

The convergence speed of QL algorithms depends on the application and its associated environmental complexities [27]. When QL is applied in a new environment, the agent must explore and exploit the reward to gradually discover the optimal action A_t that maximizes the Q-value. Note that ε is defined as follows:

$$\varepsilon = e^{-\frac{T_{\text{run}}}{T_{\text{simu}}}}, \tag{9}$$

where T_{run} is the running time and T_{simu} is the system simulation time. Convergence to an optimal policy is guaranteed by the decay function in our proposed QL-based MAC for an IoT-enabled MANET.

6.3 A priori approximate controller

From the first transmission of the *SAR* control packet, the application of the strategy defined in Eq. (9) can result in instantaneous performance benefits.

An initial condition is required for an iterative algorithm such as QL. After the first update, the initial condition will be changed. For QL, the initial condition is always zero. The initial untrained Q-table is set as shown in Table 3: the rows represent the possible states, and the columns represent the action space. Here, the rows represent the optimal number of contention nodes, and X_t denotes the current number of contention nodes at time step t. $X_t - d$ denotes the number of contention nodes decreased by d, whereas $X_t + d$ denotes that the number of contention nodes increased by d. The initial value for the initial untrained Q-table with size $[m, 3]$ is zero, except for $Q[0, 0]$ and $Q[m, 2]$, which are set to extreme negative values (-100) to ensure that they are never visited by the IoT node.

Table 3 Initial Q-value table for our proposed QL-based MAC scheme

	$X_t - d$	X_t	$X_t + d$
1	− 100	0	0
2	0	0	0
:	:	:	:
:	:	:	:
$m - 1$	0	0	0
m	0	0	− 100

Each node uses a controller that depends on the traffic rate and the number of contention nodes for each contention period. In this study, all the nodes are in a one-hop environment, and the traffic destination is either the cluster head or other IoT nodes in the IoT network. The controller is trained a priori for $\gamma = 0.9$.

6.4 Implementation details

An *SAR* control packet must be sent in the *SAR* sub-period for each node that wants to transmit data in the IoT networks. Once a beacon interval period ends, the cluster head can know the number of all contention nodes. The cluster head collects the number of contention periods in the beacon interval. The *SAC* control frame contains the contention information in each contention period and broadcasts by the cluster head when each contention period ends. There are three possible statuses for each slot: successful, collision, and idle. Each node calculates the results of Eq. (10) by assigning suitable values for the parameters μ and ν.

$$\begin{aligned} R_t 7 \mu \frac{N_{\text{avg}}}{\text{CP}_{\text{num}} \times m} + \nu \frac{N_{\text{avg}}}{\text{Slot}_{\text{tol}}} \\ = \mu \frac{N_{avg}}{\text{CP}_{\text{num}} \times m} + \nu \frac{N_{\text{avg}}}{\sum_{i=1}^{N_{\text{avg}}} \text{Slot}_{\text{con}}(i)} \end{aligned} \tag{10}$$

where CP_{num} denotes the total number of contention periods in the beacon interval. $\text{Slot}_{\text{con}}(i)$ denotes the total number of contentions in the beacon interval for IoT node i until achieving successful contention.

Algorithm 1 shows the contention mechanism selection for the proposed QL-based DQMAC protocol in IoT networks.

```
Algorithm 1 : Contention mechanism selection for QL-based DQMAC protocol

01: After one beacon interval of the QL-DQMAC protocol, the cluster head
     calculates the average number of contention nodes, $N_{avg}$.
02: The contention node sends the $SAR$ in selected slots.
03: The cluster head broadcasts the $SAC$ after one contention period.
04: Each node obtains the optimal intended contention number $X$ from Algorithm 2.
05: Find $N_{con}^i$ in the contention period $i$ from Eq.(4).
06: Each IoT node before contention calculates the active rate according to Eq.(5).
07: Each node decides itself whether it is active or in sleeping mode in the next
     contention period according to the results of Eq.(5).
08: If the $N_{con}^i \geq DQ_{condition}$ then
09:    Distributed queuing (DQ) MAC executed.
10: else
11:    all the nodes contend in only one contention period.
12: endif
```

In the Algorithm 2, the action selection is based on P_ε. When $P_\varepsilon < \varepsilon$, the optimal number of contention nodes in the next period is randomly selected from $(X_t - d, X_t, X_t + d)$. Otherwise, the action is determined by the controller (Eq. (8)). The reward is defined by Eq. (10). The value of P_ε is randomly selected from the interval (0, 1).

```
Algorithm 2 : Q-Learning DQMAC protocol

01: Initialize $Q_0(X_t, A)$ at $t = 0$.
02: Set the values of $\mu$ and $\nu$.
03: Randomly select $P_\varepsilon \in$ random (0, 1).
04: if $P_\varepsilon < \varepsilon$ then
05:     $A_{t+1} \longleftarrow$ random $(X_t - d, X_t, X_t + d)$.
06: else if $P_\varepsilon \geq \varepsilon$ then
07:     $A_{t+1} \longleftarrow A_\pi$ .
08: end if
09: After one beacon interval, each node calculates the reward using Eqs.(10).
10: Update $Q(X_{t+1}, A_{t+1})$ from Algorithm 3.
```

Algorithm 3 shows the Q value and reward calculation for the proposed QL-based DQMAC protocol in IoT networks.

```
Algorithm 3 : Q value and reward calculation for QL-based DQMAC protocol

01: Initialize $Q_0 = 1$ at $t = 0$.
02: Set the learning rate $\alpha$, $0 < \alpha < 1$.
03: $Q_{t+1} = (1 - \alpha)Q_t + \alpha R_t$.
04: Update $Q_{t+1}$.
```

7 Results and discussion

7.1 Throughput

In [28], the author derived a performance analysis of IEEE 802.11. The saturation throughput of the licensed channels was analyzed in [29].

The throughput per contention period size for IoT networks owing to the simulation ending, ζ, is defined as follows:

$$\zeta = \frac{R_{\text{CH}} T_{\text{suc}}}{T_{\text{simu}}}. \tag{11}$$

where R_{CH} is the data transmission rate for the licensed channel, T_{suc} is the total successful data transmission time, and the T_{simu} is the system simulation time.

Figure 8 shows the system throughput in traditional DQMAC and QL-based DQMAC in IoT networks. The system throughput in QL-based DQMAC ranges from 0.455 to

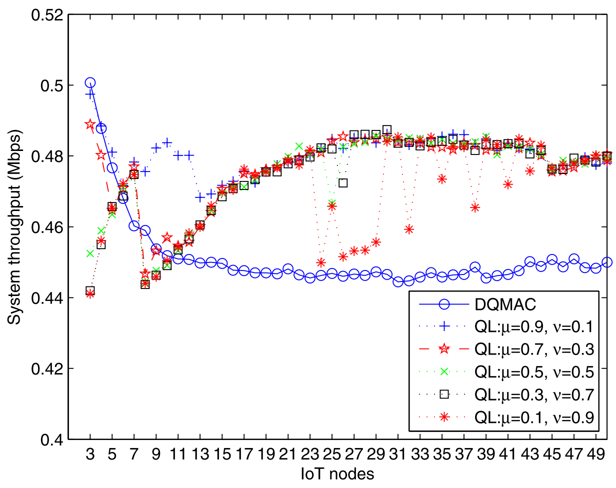

Fig. 8 Comparison of system throughput of IoT node in traditional DQMAC, QL-based DQMAC in a IoT network

0.485 Mbps for $\mu = 0.7$ and $\nu = 0.3$ while the number of IoT nodes is greater than 11. Under the same conditions, the number of contention for MAC contention in traditional DQMAC ranges from 0.445 to 0.450 Mbps. The largest improvement for system throughput in QL-based DQMAC compared with traditional DQMAC is 6.59%.

7.2 Number of contention for MAC

Each IoT node sends *SAR* in the contention period for the QL-based DQMAC protocol until it succeeds. Each node contends only once for each contention period until the successful transmission. Therefore, the number of contend periods for each IoT node equals the transmission number of the *SAR*.

The average number of contentions for one IoT node to achieve a successful transmission is defined as follows[30]:

$$n_{\mathrm{CP}} \simeq \log_m(N_{\mathrm{con}} - 1) + \left(\frac{1}{2} + \frac{\gamma}{\log m}\right) + \frac{1}{2N_{\mathrm{con}}\log m} \tag{12}$$

Here, $\gamma \approx 0.5772$ is the Euler's constant. The value of n_{CP} is finite, whereas m is very low. N_{con} is the number of contention nodes and is defined in Eq. (4); m is the number of contention slots for the contention period, and is the executed condition of Eq. (6). In addition, when N_{con} is low or high, the value of n_{CP} is similar regardless of the number of contention slots, m.

In the proposed QL-based DQMAC protocol, the IoT node has two states: sleeping and active. The number of MAC contentions is the sum of the number of contention failure and one successful contention. Therefore, the number of MAC contention in the proposed QL-based DQMAC for a node is n_{CP}:

$$n_{\mathrm{CP}} = N_{\mathrm{CRQ}} \tag{13}$$

where N_{CRQ} is the average number of collision resolution in MAC contention periods.

Figure 9 shows the level of MAC contention in traditional DQMAC and QL-based DQMAC in IoT networks. The level of MAC contention before successful transmission

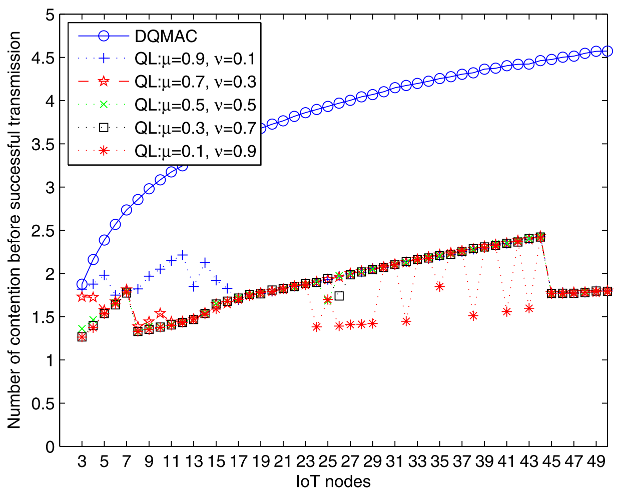

Fig. 9 Comparison of average contentions before successful transmission (n_{CP}) of IoT node for traditional DQMAC, QL-based DQMAC in a IoT network

in QL-based DQMAC ranges from 1.74 to 2.43 for $\mu = 0.7$ and $\nu = 0.3$. Under the same conditions, the level for MAC contention before successful transmission in traditional DQMAC ranges from 1.87 to 4.57. The largest improvement in the level for MAC contention before successful transmission in QL-based DQMAC compared with traditional DQMAC is 46.83%.

MAC collisions occur in traditional DQMAC with a greater number of active IoT nodes and are more relevant than in QL-based DQMAC because of the lack of a sleeping-mode mechanism in the contention period.

7.3 Delay for MAC contention

The arrival process of IoT nodes in IoT networks presents a Poisson distribution with arrival rate λ. In the proposed QL-based DQMAC, the arrival process is memoryless, and the MAC protocol uses a tree-splitting mechanism to perform the collision resolution. The probability of N^i_{con} for active IoT nodes contending in contention period i is denoted by $P(k = N^i_{\text{con}})$ and expressed as follows [31]:

$$P(k = N^i_{\text{con}}) = \frac{\lambda^{N^i_{\text{con}}}}{N^i_{\text{con}}!} e^{-\lambda} \tag{14}$$

Then, the probability of an active IoT node randomly and successfully selecting a free mini-slot when there are N^i_{con} active IoT nodes in a given frame is given by the following expression:.

$$P(\text{succ}|k = N^i_{\text{con}}) = m\left(\frac{1}{m}\right)\left(1 - \frac{1}{m}\right)^{N^i_{\text{con}}-1} \tag{15}$$

Let $P(\lambda)$ be the probability of an IoT node successfully selecting one idle mini-slot in contention period i. $P(\lambda)$ is defined as follows [32]:

$$\begin{aligned} P(\lambda) &= \sum_{N_{\text{con}}^i=0}^{\infty} P(\text{succ}|k = N_{\text{con}}^i)P(k = N_{\text{con}}^i) \\ &= e^{-\lambda} + \sum_{N_{\text{con}}^i=1}^{\infty} P(\text{succ}|k = N_{\text{con}}^i)P(k = N_{\text{con}}^i) \\ &= e^{-\lambda} + \sum_{N_{\text{con}}^i=1}^{\infty} \frac{\lambda^{N_{\text{con}}^i}}{N_{\text{con}}^i!} e^{-\lambda} m\left(\frac{1}{m}\right)\left(1-\frac{1}{m}\right)^{N_{\text{con}}^i-1} \\ &= \frac{me^{-\frac{\lambda}{m}} - e^{-\lambda}}{m-1} \end{aligned} \tag{16}$$

The service time of the collision resolution queue is a Poisson-distributed random variable with mean [31]

$$t_{\text{CRQ}} = \left(ln\left[\frac{1}{1-P(\lambda)}\right]\right)^{-1} \tag{17}$$

Therefore, if the model for the collision-resolution queue is M/M/1, the service rate is defined as follows:

$$\mu_{\text{CRQ}} = (t_{\text{CRQ}})^{-1} = ln\left[\frac{1}{1-P(\lambda)}\right] \tag{18}$$

Here, we define the average delay for CRQ in the proposed QL-based DQMAC as follows:

$$\overline{T}_{\text{CRQ}} = \frac{1}{\mu_{\text{CRQ}} - \lambda} = \left(ln\left[\frac{1}{1-P(\lambda)}\right] - \lambda\right)^{-1} \tag{19}$$

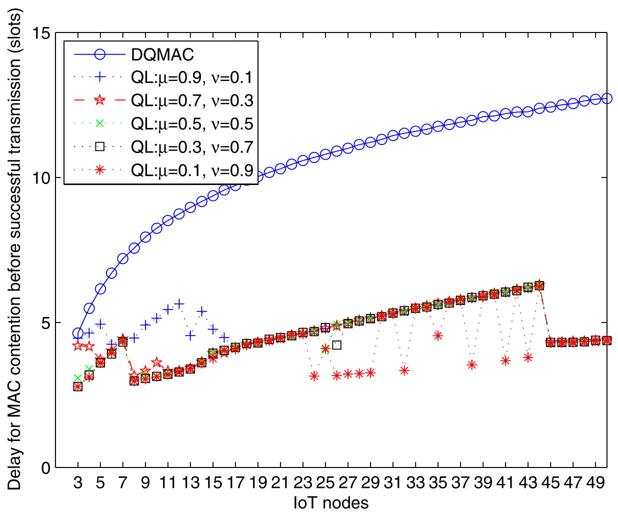

Fig. 10 Comparison of delay of MAC contention before success transmission of IoT node for traditional DQMAC, QL-based DQMAC in a IoT network

Figure 10 shows the delay of MAC contention before successful transmission in DQMAC and QL-based DQMAC in IoT networks. The delay of MAC contention before successful transmission in QL-based DQMAC ranges from 3.16 to 6.30 slots for $\mu = 0.7$ and $\nu = 0.3$. Under the same conditions, the delay of MAC contention in traditional DQMAC ranges from 4.62 to 12.72 slots. The largest improvement in the delay of MAC contention before successful transmission in QL-based DQMAC compared with traditional DQMAC is 50.49%.

7.4 Energy consumption for MAC contention

The average energy consumption for an IoT node for MAC contention in a QL-based DQMAC can be expressed as follows:

$$\overline{E} = \overline{E}_{\text{CRQ}} + \epsilon_{\text{sleep}} \overline{T}_{\text{sleep}} \tag{20}$$

where $\overline{E}_{\text{CRQ}}$ is the average consumption during the collision resolution in the contention period, ϵ_{sleep} is the power consumption in the sleeping mode, and $\overline{T}_{\text{sleep}}$ is the average time in the sleeping mode.

$\overline{E}_{\text{CRQ}}$ can be expressed as follows:

$$\overline{E}_{\text{CRQ}} = n_{\text{CP}} \overline{E}_{\text{SAR}} + n_{\text{listen}} \epsilon_{\text{listen}} \overline{T}_{\text{listen}} \tag{21}$$

where $\overline{E}_{\text{SAR}}$ is the average energy consumption when the IoT node sends one SAR control frame and contends with other IoT nodes in the contention period until it succeeds; $\overline{T}_{\text{listen}}$ is the average time in listening mode.

Each node sends the SAR control frame in a selected mini-slot in the contention period; this node in other mini-slots will be in listening mode. After the SAR period, each node receives one SAC control frame, which is broadcast by the cluster head.

Then, the E_{SAR} can be expressed as follows:

$$\overline{E}_{\text{SAR}} = (\epsilon_{tx} + (m-1)\epsilon_{\text{listen}}) T_{\text{SAR}} + \epsilon_{rx} T_{\text{SAC}} \tag{22}$$

where ϵ_{tx}, ϵ_{listen}, and ϵ_{rx} are the power consumption in the transmission, listening and reception modes, respectively; T_{SAR} is the average time required to send one SAR control frame to a contention slot; and T_{SAC} is the average time required to send one SAC control frame by a cluster head in a contention slot.

Therefore,

$$\begin{aligned} \overline{E} &= \overline{E}_{\text{CRQ}} + \epsilon_{\text{sleep}} \overline{T}_{\text{sleep}} \\ &= n_{\text{CP}} E_{\text{SAR}} + n_{\text{listen}} \epsilon_{\text{listen}} \overline{T}_{\text{slot}} + \epsilon_{\text{sleep}} \overline{T}_{\text{sleep}} \\ &= n_{\text{CP}}((\epsilon_{tx} + (m-1)\epsilon_{\text{listen}}) T_{SAR} + \epsilon_{rx} T_{\text{SAC}}) \\ &\quad + n_{\text{listen}} \epsilon_{\text{listen}} \overline{T}_{slot} + \epsilon_{\text{sleep}} \overline{T}_{\text{sleep}} \end{aligned} \tag{23}$$

Figure 11 shows the energy consumption for MAC contention in traditional DQMAC and QL-based DQMAC in IoT networks. The energy consumption for MAC contention in QL-based DQMAC ranges from 34.35 to 46.44 mW for $\mu = 0.7$ and $\nu = 0.3$. Under the same conditions, the energy consumption for MAC contention in traditional DQMAC ranges from 60.29 to 113.19 mW. The largest improvement in energy consumption for MAC contention in QL-based DQMAC compared with traditional DQMAC is 46.73%.

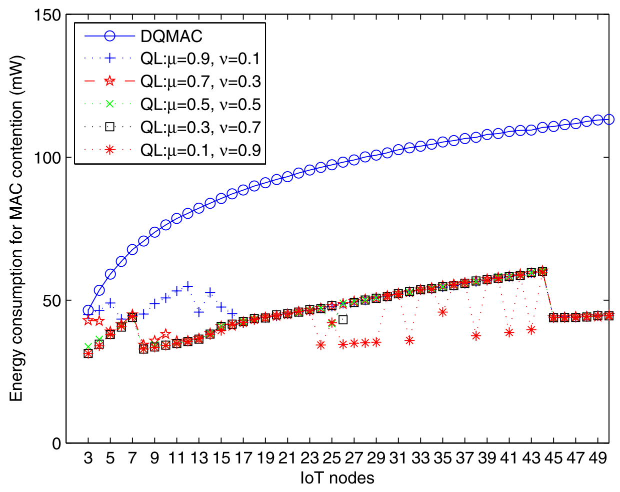

Fig. 11 Comparison of energy consumption for MAC contention of IoT node in traditional DQMAC, QL-based DQMAC in a IoT network

An IoT node will enter the sleeping mode after successful transmission for QL-based DQMAC. The IoT node, after successful transmission, will also enter the sleeping mode for traditional DQMAC. Figure 12 shows the total energy consumption per beacon interval in traditional DQMAC and QL-based DQMAC in an IoT network. The total energy consumption per beacon interval in QL-based DQMAC ranges from 90.49 to 158.76 mW for $\mu = 0.7$ and $\nu = 0.3$. Under the same conditions, the total energy consumption per beacon interval in traditional DQMAC ranges from 122.46 to 300.22 mW. The largest improvement in the total energy consumption per beacon interval in QL-based DQMAC compared with traditional DQMAC is 47.12%.

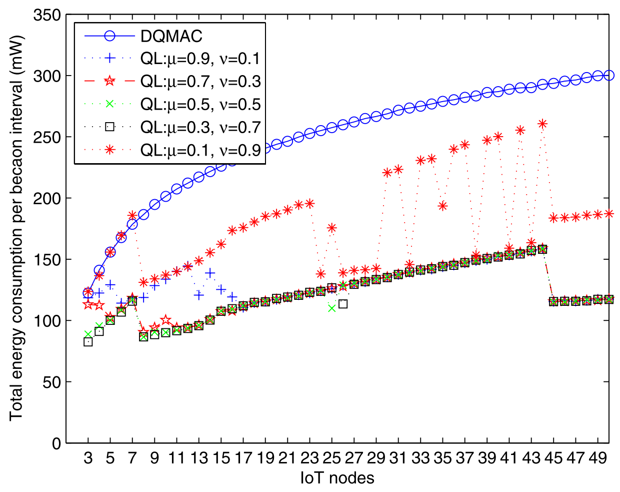

Fig. 12 Comparison of total energy consumption of IoT node in traditional DQMAC, QL-based DQMAC in a IoT network

To demonstrate the effectiveness of reinforcement learning sufficient training data is required. Therefore, when the amount of data is insufficient, Q-learning cannot be performed optimally. However, when the amount of data becomes excessively large and exceeds the saturation point of the system, Q-learning may diverge and degrade system performance [33].

In [5], the authors showed that the target system throughput was achieved when three contention slots were used in the DQ mechanism. Therefore, the number of *SAR* slots for one contention period was set to 3 in Table 1. In Figs. 8, 9, 10, 11 and 12, there is a degrades suddenly for system performance under 8 and 45 IoT nodes for $\mu = 0.7$ and $\nu = 0.3$. The system performance becomes unstable when the number of contention nodes is either excessively low (i.e. under 8) or excessively high (i.e. beyond 45) under 3 *SAR* slots for one contention period in our proposed QL-based DQMAC in an IoT network for $\mu = 0.7$ and $\nu = 0.3$. The simulation results in Figs. 8, 9, 10, 11 and 12 show that the total number of IoT contention nodes between 8 and 44 for our proposed QL-based DQMAC in an IoT network will have superior and more stable system performance. The average delay for CRQ, denoted by $\overline{T}_{CRQ}$ and defined in Eq. (19). In Fig. 10, and the line curve of $\overline{T}_{CRQ}$ versus the number of IoT nodes are consistent with the average delay in MAC contention before successful transmission for the proposed QL-based DQMAC protocol, whereas the number of contention nodes is between 8 and 44. Moreover, the proposed QL-based DQMAC protocol had a lower average delay in MAC contention before successful transmission than the DQMAC protocol.

8 Conclusion

This study proposes a QL-based DQMAC protocol to achieve an energy-efficient MAC contention with a low number of contentions and high system throughput. The contention reduction in QL-based DQMAC was found to be greater than in traditional DQMAC because of the sleeping mode mechanism and optimal number of contention IoT nodes in QL-based DQMAC. In addition, the QL-based DQMAC protocol reduces the number of MAC contention collisions with respect to traditional DQMAC. The proposed QL-based DQMAC scheme effectively achieves not only a lower number of contentions, but also lower energy consumption and lower end-to-end delay than traditional DQMAC. Simulation results show that the largest improvement in the number of contentions for MAC contention before successful transmission in QL-based DQMAC compared with traditional DQMAC was 46.83% (for $\alpha = 0.7$ and $\beta = 0.3$). The largest improvement in the delay of MAC contention before successful transmission in QL-based DQMAC compared with traditional DQMAC was 50.49% (for $\alpha = 0.7$ and $\beta = 0.3$). The largest improvement in the total energy consumption per beacon interval in QL-based DQMAC compared with traditional DQMAC was 47.12%. The largest improvements in energy consumption for MAC contention in QL-based DQMAC compared with traditional DQMAC was 46.73% (for $\alpha = 0.7$ and $\beta = 0.3$). Finally, the largest improvement in system throughput in QL-based DQMAC compared with traditional DQMAC was 6.59% when the number of IoT nodes was larger 11.

Abbreviations

MAC	Medium access control
DQMAC	Distributed queuing medium access control protocol
QL-based DQMAC	Q-learning-based distributed queuing medium access control protocol
IoT	Internet-of-Things
M2M	Machine-to-machine
TDMA	Time division multiple access
DQRAP	Distributed queuing random access protocol
DTQ	Data transmission queue
CRQ	Collision resolution queue
S-MAC	Synchronized sensor-MAC
WSN	Wireless sensor networks
X-MAC	Low-power MAC
MDP	Markov decision process
ML	Machine-learning
QL	Q-learning
RL	Reinforcement-learning
ARS	Access request sequence
DLMA	Deep-reinforcement learning multiple access
RL-MAC	RL-based MAC
SAC	Slot access confirm
SAR	Slot access request
CSMA	Carrier sense multiple access

Acknowledgements
The authors would like to thank the editor and reviewers for their valuable comments and suggestions.

Author contributions
CMWU author wrote all the sections. The other authors did the simulation programming. All authors read and approved the final manuscript.

Funding
This work is supported by Ministry of Science and Technology, Taiwan, R.O.C. under Grant MOST 108-2221-E-343-002 -.

Availability of data and materials
No availability of data and materials.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 14 April 2022 Accepted: 7 August 2023
Published online: 16 August 2023

References

1. A.A. Khan, M.H. Rehmani, A. Rachedi, Cognitive-radio-based internet of things: applications, architectures, spectrum related functionalities, and future research directions. IEEE Trans. Wirel. Commun. **24**(3), 17–25 (2017)
2. R.N.S.D.A. Vasavi, R. Priya, The internet of everything: a survey, in *2021 13th International Conference on Computational Intelligence and Communication Networks (CICN)*, 28 October 2021, Lima, Peru (2021)
3. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswam, Internet of things (iot): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1465–1660 (2013)
4. E.A. Shammar, A.T. Zahary, The internet of things (iot): a survey of techniques, operating systems, and trends. Library Hi Tech **38**(1), 5–66 (2020)
5. W. Xu, G. Campbell, A distributed queuing random access protocol for a broadcast channel, in *SIGCOMM '93: Conference Proceedings on Communications Architectures, Protocols and Applications*, 13–17 September 1993; California, San Francisco, USA (1993)
6. W. Wu, Y. Li, Y. Zhang, B. Wang, W. Wang, Distributed queueing-based random access protocol for loRa networks. IEEE Internet of Things J. **7**, 763–772 (2020)
7. W. Ye, J. Heidemann, D. Estrin, An energy-efficient mac protocol for wireless sensor networks, in *Proceedings Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 23–27 June 2002; New York, NY, USA (2002)
8. M. Buettner, G.V. Yee, E. Anderson, R. Han, X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks, in *SenSys06: ACM Conference on Embedded Network Sensor Systems*, 31 October 2006–3 November 2006; Boulder, Colorado, USA (2006)
9. J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 3–5 November 2004; MD, Baltimore, USA (2004)

10. R. Ali, Y.A. Qadri, Y.B. Zikria, T. Umer, B.S. Kim, S. Kim, Q-learning-enabled channel access in next-generation dense wireless networks for iot-based ehealth systems. EURASIP J. Wirel. Commun. Netw. **178**, 1–12 (2019)
11. P. Pierrick, B. Conche, G. Serge, Distributed ensembles of reinforcement learning agents for electricity control, in *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 18 August 2022; Barcelona, Spain (2022)
12. S. Amuru, Y. Xiao, M.V.D. Schaar, R.M. Buehrer, To send or not to send: learning mac contention, in *2015 IEEE Global Communications Conference (GLOBECOM)*, 6–10 December 2015; San Diego, CA, USA (2015)
13. Z. Liu, I. Elhanany, Rl-mac: a qos-aware reinforcement learning based mac protocol for wireless sensor networks, in *2006 IEEE International Conference on Networking, Sensing and Control*, 23–25 April 2006; Ft. Lauderdale, FL, USA (2006)
14. C. Wu, S. Ohzahata, Y. Ji, T. Kato, A mac protocol for delaysensitive vanet applications with self-learning contention scheme, in *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 10–13 January 2014; Las Vegas, Nevada, USA (2014)
15. F.C. Keras et al., https://keras.io (2015)
16. A.T. Ahmed, A.N. Rashid, S. Khalid, Energy model of wireless sensor network, in *2021 14th International Conference on Developments in eSystems Engineering (DeSE)*, 07–10 December 2021; Sharjah, United Arab Emirates (2021)
17. A. Laya, C. Kalalas, F. Vazquez-Gallego, L. Alonso, J. Alonso-Zarate, Goodbye, aloha. IEEE Access **4**, 2029–2044 (2016)
18. F. Gazi, N. Ahmed, S. Misra, W. Wei, Reinforcement learning-based mac protocol for underwater multimedia sensor networks. ACM Trans. Sens. Netw. **18**(3), 1–25 (2022)
19. V. Jorge, B. Javier, Q-learning for opportunistic networks. Future Internet **14**, 348 (2022)
20. H. Jiang, R. Gui, L. Chen, Z. Wu, J. Dang, J. Zhou, An improved sarsa λ reinforcement learning algorithm for wireless communication systems. IEEE Access **7**, 115418–115427 (2019)
21. Y. Yu, T. Wang, S.C. Liew, Deep-reinforcement learning multiple access for heterogeneous wireless networks. IEEE J. Sel. Areas Commun. **37**(6), 1277–1290 (2019)
22. L. Jianjun, L. Lu, W. Ying, Qos-oriented media access control using reinforcement learning for next-generation wlans. Comput. Netw. **24**, 1–12 (2022)
23. M.P. Mota, A. Valcarce, J.M. Gorce, J. Hoydis, The emergence of wireless mac protocols with multi-agent reinforcement learning, in *2021 IEEE Globecom Workshops (GC Wkshps)*, 24 January 2022; Madrid, Spain (2021)
24. H.Y. Chen, M.L. Ku, S.J. Jou, C.C. Huang, A q-learning approach with collective contention estimation for bandwidth-efficient and fair access control in IEEE 802.11p vehicular networks. IEEE Trans. Veh. Technol. **68**(9), 9136–9150 (2019)
25. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, 1988)
26. C.J.C.H. Watkins, P. Dayan, Q-learning. Mach. Learn. **8**, 279–292 (1992)
27. A. Pressas, Z. Sheng, D. Ali, F. Tian, M. Nekovee, Contention-based learning mac protocol for broadcast vehicle-to-vehicle communication, in *2017 IEEE Vehicular Networking Conference (VNC)*, 27–29 November 2017; Torino, Italy (2017)
28. G. Giuseppe, Performance analysis of the IEEE 802.11 distributed coordination function. IEEE J. Sel. Areas Commun. **18**(3), 535–547 (2000)
29. Z. Sadreddini, O. Makul, T. Çavdar, F.B. Günay, Performance analysis of licensed shared access based secondary users activity on cognitive radio networks, in *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, 18–19 April 2018; Istanbul, Turkey (2018)
30. K.H. Lee, D. Kim, Cfx: contention-free channel access for IEEE 802.11ax. Sens. Netw. **22**(33), 1–20 (2022)
31. L. Alonso, R. Agusti, O. Sallent, A near-optimum mac protocol based on the distributed queuing random access protocol (dqrap) for a cdma mobile communication system. IEEE J. Sel. Areas Commun. **18**(9), 1701–1718 (2000)
32. W. Wu, Y. Li, B. Zhang, Y. Wang, W. Wang, Distributed queuing based random access protocol for lora networks. IEEE Internet Things J. **7**(1), 763–772 (2020)
33. B. Jang, M. Kim, G. Harerimana, J.W. Kim, Q-learning algorithms: a comprehensive classification and applications. IEEE Access **7**, 133653–133667 (2019)

Publisher's Note