Hindawi
Security and Communication Networks
Volume 2021, Article ID 5530023, 11 pages
https://doi.org/10.1155/2021/5530023

Research Article

Adaptive Routing Strategy Based on Improved Double Q-Learning for Satellite Internet of Things

Jian Zhou,[1,2,3] **Xiaotian Gong**,[1,2] **Lijuan Sun**,[1,2] **Yong Xie**,[1,2] **and Xiaoyong Yan**[1,2]

[1]*College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[3]*Ministry of Education Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, Nanjing 211189, China*

Correspondence should be addressed to Lijuan Sun; lucifinil919@126.com

Received 17 February 2021; Revised 31 March 2021; Accepted 8 April 2021; Published 21 April 2021

Academic Editor: Hao Peng

Satellite Internet of Things (S-IoT), which integrates satellite networks with IoT, is a new mobile Internet to provide services for social networks. However, affected by the dynamic changes of topology structure and node status, the efficient and secure forwarding of data packets in S-IoT is challenging. In view of the abovementioned problem, this paper proposes an adaptive routing strategy based on improved double Q-learning for S-IoT. First, the whole S-IoT is regarded as a reinforcement learning environment, and satellite nodes and ground nodes in S-IoT are both regarded as intelligent agents. Each node in the S-IoT maintains two Q tables, which are used for selecting the forwarding node and for evaluating the forwarding value, respectively. In addition, the next hop node of data packets is determined depending on the mixed Q value. Second, in order to optimize the Q value, this paper makes improvements on the mixed Q value, the reward value, and the discount factor, respectively, based on the congestion degree, the hop count, and the node status. Finally, we perform extensive simulations to evaluate the performance of this adaptive routing strategy in terms of delivery rate, average delay, and overhead ratio. Evaluation results demonstrate that the proposed strategy can achieve more efficient and secure routing in the highly dynamic environment compared with the state-of-the-art strategies.

1. Introduction

Satellite Internet of Things (S-IoT) is an integration of satellite networks [1] and IoT [2]. S-IoT not only strengthens communication by using relay satellites, but also forms a new mobile Internet [3] oriented toward the integrated satellite-terrestrial information network architecture [4]. S-IoT has the advantages of wide coverage and high robustness and can provide ubiquitous services for social networks, so it has attracted considerable attention [5, 6].

As the fundamental of communication protocol for S-IoT, the routing strategy is responsible for data packet forwarding and is of great significance to the communication security [7–9]. Compared with terrestrial networks, S-IoT has the following characteristics. (1) The high-speed movement of satellite nodes and the frequent failure of sensor nodes result in the dynamic topology structure, causing unstable end-to-end path in S-IoT. (2) The complex space environment and the uneven amount of terrestrial access data lead to the dynamic node status. (3) Due to the limited energy of satellite nodes and sensor nodes, energy efficiency must be taken into account in the routing strategy to reduce the overhead ratio. (4) The large number of nodes and the heterogeneity among nodes impose specific requirements upon the efficiency and security during data packet forwarding. With all these characteristics in mind, we conclude that the routing strategy for the terrestrial network is not applicable to S-IoT.

We study S-IoT as a delay tolerant network (DTN) without intersatellite links. Since S-IoT involves heavy data

service workloads, which are generally not in requirement of very low delay, the store-carry-forward mechanism of DTN, which can cope with the dynamic topology structure in S-IoT, is used by the satellite nodes to forward data packets. In recent years, DTN has attracted extensive attention of researchers, and many routing strategies for DTN have been proposed. Existing routing strategies usually can be classified into three categories including the flood-based, the utility-based, and the mobility model-based routing strategies. To be specific, we select several representative routing strategies falling within individual categories and discuss them in brief. The Epidemic routing strategy proposed by Vahdat et al. [10] is one of the flood-based routing strategies, in which one node forwards data packets to every node it encounters. This virus-like propagation mode results in excessive overhead ratio. In order to improve Epidemic, Spyropoulos et al. [11] proposed the Spray-and-Wait routing strategy. The process of data packet forwarding consists of two phases, i.e., spraying and waiting. The data packets are diffused into some copies in the spraying phase. These copies are directly forwarded to the destination node in the waiting phase. This strategy reduces the overhead ratio and achieves the similar performance in transmission with Epidemic. As one of the utility-based routing strategies, the Prophet routing strategy was proposed by Lindgren and Doria [12]. In this strategy, each data packet makes a copy to the node only in case of a high encountering probability, for the purpose of reducing the amount of replication and the overhead ratio. Sharma et al. [13] proposed the machine learning routing strategy based on Prophet (MLRSP). This strategy takes the speed and location of nodes into account and uses the decision tree as well as the neural network to calculate the encountering probability, achieving better performance than Prophet dose. Among mobile model-based routing strategies, the contact graph routing strategy, which was proposed by Araniti et al. [14], is capable of reducing the average delay by selecting the next hop node based on the minimum hop count and the shortest path.

However, the aforementioned routing strategies for DTN cannot quickly adapt to the frequent changes of node status, and the copies in these strategies bring in the challenge of communication security in S-IoTs. To tackle these challenges, we propose employing reinforcement learning on the basis of our previous work [15] to develop a novel adaptive routing strategy for S-IoT. Since the reinforcement learning can obtain optimal results even if the system environment changes frequently, it has been successfully applied in a variety of fields such as industrial manufacturing, analogue simulation, game competition, and scheduling management. As a reinforcement learning algorithm, double Q-learning [16] chooses the next better hop node by self-learning to cope with the dynamic changes of topology structure and node status in S-IoT while satisfying the communication security requirement.

In view of the dynamic topology structure and the dynamic node status, this paper presents an adaptive routing strategy based on improved double Q-learning for S-IoT. The main contributions of this paper are as follows:

(1) We apply the reinforcement learning to the S-IoT routing strategy to make it adapt to the dynamic changes of topology structure and node status in S-IoT.

(2) We improve the forwarding performance by means of optimizing the mixed Q value, the reward value, and the discount factor, respectively, based on the congestion degree, the hop count, and the node status.

(3) We establish the S-IoT model, which consists of a ground layer, a LEO layer, and a MEO layer, to perform simulation experiments. Simulation results demonstrate that the proposed strategy improves the performance of data packet forwarding, in terms of delivery rate, average delay, and overhead ratio, compared with the state-of-the-art strategies.

The rest of this paper is organized as follows. Section 2 introduces the related work. The description of the proposed adaptive routing strategy is detailed in Section 3. Section 4 discusses how to improve the Q value in double Q-learning. Simulation results and the associated analysis are given in Section 5. Section 6 concludes this paper.

2. Related Work

2.1. Routing Strategy for Satellite Networks. Satellite networks not only provide remote transmission capability for IoT, but also provide cloud computing capability [17–19], so satellite networks have direct impact on the overall performance of S-IoTs. The routing strategy for satellite networks is responsible for data transmission and distribution between satellites under various security requirements. In recent years, routing strategies for satellite networks are extensively studied in the literature.

Some researchers paid attention to the dynamic changes of the topology structure caused by the high-speed movement of satellites. Gounder et al. [20] proposed a routing strategy based on snapshot sequence. Mauger and Rosenburg [21] proposed a routing strategy based on virtual nodes. Hashimoto and Sarikaya [22] proposed a routing strategy based on division of the coverage area. Wang et al. [23] proposed a routing strategy based on position and velocity of the nodes. Though simple and easy-to-implement for routing computation, they often need high storage capacities. Some researchers focused on the limited energy caused by the lack of continuous energy supply. Ekici et al. [24] proposed a routing strategy for saving the energy cost. Yang et al. [25] proposed an energy-efficient routing strategy. Marchese and Patrone [26] proposed an energy-aware routing strategy. These strategies can reduce energy consumption, but they induce high computational burden. Some other researchers are concerned about the poor QoS caused by long distance between nodes and unstable links. Mao et al. [27] proposed a routing strategy separating the collection and calculation of QoS. Huang et al. [28] proposed a routing strategy under guaranteed delay constraints. Xu et al. [29] proposed a routing strategy based on asynchronous transfer mode. However, these strategies focused on

improving the QoS of voice and multimedia services and failed to consider data services.

It is worth emphasizing that all the routing strategies mentioned above use the intersatellite links. In existing low earth orbit (LEO) and medium earth orbit (MEO) constellation systems, only Iridinm is equipped with intersatellite links due to the high cost and complex system. Other constellation systems, such as Ocbcomm, Globalstar, and O3b, have no intersatellite links [30]. For this reason, it is more reasonable to construct the S-IoT based on the constellation systems without intersatellite links, which is the main purpose of this work.

2.2. Routing Strategy Based on Reinforcement Learning. In recent years, reinforcement learning has attracted widespread attention. As a classic reinforcement learning algorithm, Q-learning [31] obtains the sample data sequence (state, action, and reward value) through interacting with the environment and uses the state-action function value (Q value) to find the best action for the current state. In addition, Q-learning ensures communication security by the self-learning mechanism. Q-learning has been applied in many fields. Deng et al. [32] applied Q-learning to the task allocation of edge computing. Zhao et al. [33] applied Q-learning to the DoS attack of many core systems.

In the routing field, Elwhishi et al. [34] proposed a Q-learning routing strategy for DTN. In this strategy, nodes collaborate with each other and make forwarding decisions based on connections. However, node status is not considered in this work. Plate and Wakayama [35] proposed a Q-learning routing strategy based on kinematics and sweep features. This strategy can adapt to the constantly dynamic changes of the topology structure caused by the node mobility and energy consumption. Rolla and Curado [36] proposed an enhanced Q-learning routing strategy for DTN. This strategy calculates the reward value based on the distance between nodes such that more data packets in densely populated areas can be delivered. Wu et al. [37] proposed an adaptive Q-learning routing strategy based on anycast (ARSA). This strategy focuses on anycast communication from a node to multiple destination nodes, while considering the encountering probability and the relative speed of nodes.

However, the abovementioned Q-learning routing strategies suffer from the overestimation issue in certain cases. The reason is that Q-learning algorithm uses the same Q value for the action selection with the action evaluation and uses the maximum Q value as an approximation to the maximum expected Q value. Q-learning tends to produce a positive estimate deviation, since the overestimated Q value has the higher chance to be selected.

The double Q-learning algorithm, which was proposed by Hasselt [16], uses two Q values to separate action selection and action evaluation. Double Q-learning has been applied in many fields. Zhang et al. [38] applied double Q-learning to the speed control of autonomous vehicle. Vimal et al. [39] applied double Q-learning to improve energy efficiency of cognitive radio networks. Zhang et al. [40] applied double Q-learning to the energy-saving scheduling of edge computing. So far, double Q-learning has been rarely used in the routing field.

The kernel idea of the double Q-learning algorithm is that the action is selected based on the greedy algorithm in each step and the two Q values are adaptively updated with the changes of environment. One Q value selects the action, and the other one evaluates the selected action. The selection is decoupled from the evaluation for reducing the positive deviation. Furthermore, double Q-learning algorithm has a similar computational efficiency compared with Q-learning algorithm. Therefore, we use double Q-learning to avoid selecting neighbor nodes with overestimation.

3. Proposed Strategy

The whole S-IoT is regarded as a reinforcement learning environment in this paper. Satellites in S-IoT are regarded as satellite nodes, whereas sensors and data centers are regarded as ground nodes. For each individual node, all other nodes it can encounter constitute its neighbor node set. In particular, ground nodes generate and receive data packets, and satellite nodes use the store-carry-forward mechanism to forward data packets.

Both satellite nodes and ground nodes are considered as intelligent agents. Each node learns the network environment of the whole S-IoT through interacting with other nodes it encounters. Furthermore, all nodes are included to form the state set of reinforcement learning. A ground node or satellite node selects one node from its neighbor node set to forward data packets. This procedure is considered as an action selection of reinforcement learning. In this manner, the neighbor node set for this node can be regarded as the possible action set. The state transitions are equivalent to forwarding data packets from one node to a neighbor node.

In the proposed strategy, each node is assigned with two Q tables (Q^A and Q^B) to store the Q value of the action which is referred to as selecting a neighbor node to forward data packets to the destination node. Each node only updates its own two Q tables and shares its local information only with its neighbor nodes. The two Q values stored in the corresponding Q tables are used to determine and evaluate the greedy strategy, respectively. More importantly, the two Q values are decoupled to address the issue of overestimation which may cause the local optima of routing. The two Q values change with the topology structure and node status such that the proposed strategy can be adaptive to the highly dynamic environment.

Initially, a new node has no knowledge of the whole S-IoT environment with two empty Q tables. When this node encounters other nodes, it records the identities of other nodes and initializes the corresponding Q values to 0 in two Q tables.

The selection of neighbor node for each data packet would update the two Q values. Each data packet has a destination node. When the data packet reaches its destination node, the Q values of all nodes on this forwarding path will be updated by a rewarding procedure. In the proposed strategy, the two Q values are intensively learned

from two different experience sets of the S-IoT. The mixed Q value depending on the two Q values decides which node should be selected to forward data packets.

Figure 1 illustrates the general routing process of a specific node. If the destination node is in its neighbor node set, this node forwards data packets to the destination node to complete data transmission. Otherwise, depending on the largest mixed Q value, this node selects a neighbor node to forward data packets. It stores and carries these data packets until it encounters the selected node. Such operations are repeated until the simulation is terminated. The greedy algorithm ensures the largest cumulative future rewards. Take node c, for example, the node selected from its neighbor node set, can be expressed as

$$x^* = \arg\max_{x \in N_c} \hat{Q}_c(d, x), \tag{1}$$

where N_c is the neighbor node set of node c and node x is one of the neighbor nodes in N_c. $\hat{Q}_c(d, x)$ is the mixed Q value of the node selection action, and node d is the destination node of the data packets. The improved method for calculating $\hat{Q}_c(d, x)$ will be given in the next section. If two nodes have identical mixed Q value, we select one of them at random.

As the learning task is assigned to each node, the learning process is accordingly the updating process of Q tables. If the topology of node c changes, the Q values in Q_c^A will be updated. If the status of node c changes, the Q values in Q_c^B will be updated. In this sense, Q_c^A and Q_c^B represent an experience set of topology change and an experience set of status change, respectively. Q_c^A and Q_c^B learn from each other. The updates of Q_c^A and Q_c^B are given by

$$Q_c^A(d, x) = (1 - \alpha) Q_c^A(d, x) + \alpha\left(R_c(d, x) + \gamma_c(d, x) Q_x^B(d, y^*)\right), \tag{2}$$

$$Q_c^B(d, x) = (1 - \alpha) Q_c^B(d, x) + \alpha\left(R_c(d, x) + \gamma_c(d, x) Q_x^A(d, z^*)\right), \tag{3}$$

where N_x is the neighbor node set of node x and α is the learning rate manipulating the updating speed of Q values. $R_c(d, x)$ is the instant reward value (R value) and $\gamma_c(d, x)$ is the discount factor of the node selection action. y^* and z^* are the nodes with the largest Q value in Q_x^B and Q_x^A, respectively. The improved method for calculating $R_c(d, x)$ and $\gamma_c(d, x)$ will be given in the next section.

4. Improvement of Q Value

4.1. Mixed Q Value Based on the Congestion Degree. The next hop node of data packets is determined according to the mixed Q value. Because network congestion has an important impact on routing, we consider the congestion degree to give the corresponding weights of two Q values to calculate the mixed Q value.

Take node c for example; if node c selects neighbor node x to forward data packets, the mixed Q value is calculated by

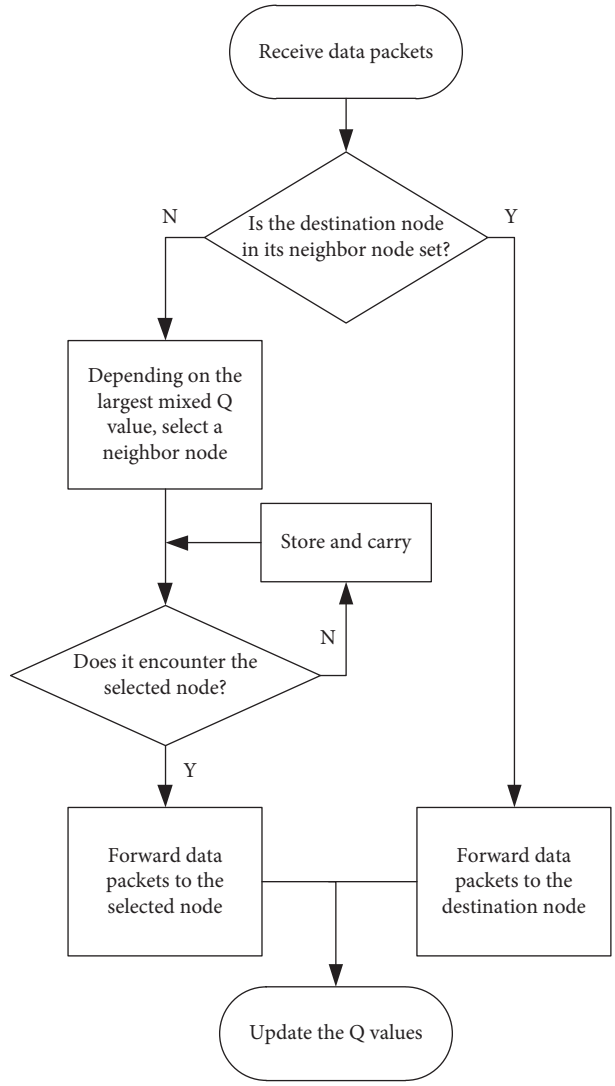

Figure 1: Routing process of each node.

$$\hat{Q}_c(d, x) = \beta(x) Q_c^A(d, x) + (1 - \beta(x)) Q_c^B(d, x), \tag{4}$$

where node d is the destination node of the data packets and $Q_c^A(d, x)$ and $Q_c^B(d, x)$ are the Q values provided by Q^A and Q^B, respectively, indicting the Q values of the action in which node c selects node x to forward data packets. $\beta(x)$ is the congestion factor of node x, and it is calculated by

$$\beta(x) = \begin{cases} 0.6, & \text{if } 0 < = con\_d(x) < 0.5, \\ 0.3, & \text{if } 0.5 < = con\_d(x) < 0.75, \\ 0.1, & \text{if } 0.75 < = con\_d(x)) < 1. \end{cases} \tag{5}$$

In particular, the smaller $con\_d(x)$ value is, the larger $\beta(x)$ value is, so that the influence of topology change is greater. Under the reverse situation, the influence of status change is greater. $con\_d(x)$ can be calculated by

$$con\_d(x) = \frac{\sum_{y \in N_x} S(y)/B_y}{C(x)}, \tag{6}$$

where $S(y)$ is the size of all data packets currently in the buffer of neighbor node y and B_y is the buffer size of neighbor node y. In addition, N_x is the neighbor node set of node x, and $C(x)$ is the number of neighbor nodes of node x.

4.2. Reward Value Based on the Hop Count. An important component in the Q value updating rule (refer to equations (2) and (3)) is the calculation of R value defining the instant reward value after forwarding data packets. R value reflects the advantages and disadvantages of one-time forwarding. Limited by the energy capacity of the S-IoT, the hop count is taken into account in the calculation of reward value to control energy consumption and to reduce the overhead ratio.

Take node c, for example; if node c has forwarded the data packets to neighbor node x, the reward value for the node selection action can be calculated by

$$R_c(d,x) = \begin{cases} 0, & \text{otherwise,} \\ e^{-\left(w_1h_1+w_2h_2+\ldots+w_ih_i+\ldots+w_kh_k\right)}, & \text{if } c == d, \end{cases} \tag{7}$$

where node d is the destination node of the data packets, $h_1, h_2, \ldots, h_i, \ldots, h_k$ are the hop counts on different satellite orbits, and $w_1, w_2, \ldots, w_i, \ldots, w_k$ are the weights of different satellite orbits satisfying $\sum_{i=1}^{k} w_i = 1$. A higher satellite orbit height stands for a greater amount of energy consumption for data transmission between the ground node and the satellite node. Hence, we set a relatively higher w_i value for a satellite node with a higher height orbit. As a result, the reward value for forwarding data packets to a satellite with a higher height orbit is lower.

4.3. Discount Factor Based on the Node Status. The discount factor is a multiplicative coefficient for the sum of subsequent reward values, which affects the possibility of reselecting a previously selected neighbor node to forward data packets. In order to adapt to the node status, the distance, direction, and buffer occupancy are considered in the calculation of discount factor.

Take node c, for example; if node c has forwarded the data packets to neighbor node x, the discount factor for the node selection action is calculated by

$$\gamma_c(d,x) = \gamma \times Dir\_F(d,x) \times Dis\_F(d,x) \times Buf\_F(d,x), \tag{8}$$

where node d is the destination node of the data packets and γ is the setting value subject to $0<\gamma<1$. $Dir\_F(d,x)$, $Dis\_F(d,x)$, and $Buf\_F(d,x)$ denote the direction factor, the distance factor, and the buffer factor, respectively. The larger these factors are, the larger the discount factor is and accordingly the larger the updated Q value is. As such, the possibility of reusing this node to forward data packets in the next time will be larger.

The direction factor is calculated by

$$Dir\_F(d,x) = 1 - \frac{\theta(d,x)}{180}, \tag{9}$$

where $\theta(x,d)$ stands for the angle between neighbor node x and destination node d. The smaller $\theta(x,d)$ value is, the larger $Dir\_F(d,x)$ value is.

The distance factor is calculated by

$$Dis\_F(d,x) = 1 - \frac{D(d,x)}{D_{\max}}, \tag{10}$$

where $D(x,d)$ is the distance from node x to destination node d and $D_{\max}$ is the maximum distance between the nodes in the network. The smaller $D(x,d)$ value is, the larger $Dis\_F(d,x)$ value is.

The buffer factor is calculated by

$$Buf\_F(d,x) = 1 - \frac{S(x)}{B_x}, \tag{11}$$

where $S(x)$ is the size of all data packets currently in the buffer of neighbor node x and B_x is the buffer size of neighbor node x. The smaller $S(x)$ value is, the larger $Buf\_F(d,x)$ value is.

5. Simulation Analysis

5.1. Simulation Environment. We use the ONE simulator to analyze and evaluate the proposed routing strategy. The S-IoT model in simulation experiments is shown in Figure 2. The ground layer is composed of 110 ground nodes, which are uniformly distributed over the Earth's surface. The LEO layer consists of 48 satellite nodes as the Globalstar constellation system. The MEO layer consists of 24 satellite nodes as the GPS constellation system. Table 1 lists the node parameters in each layer. Ground nodes generate and receive data packets, and both the source node and the destination node are randomly generated among ground nodes. Since we assume no intersatellite links in this S-IoT model, data packets cannot be forwarded between any two satellite nodes moving through their orbits periodically.

The network environment parameters in simulation experiments are shown in Table 2. Regarding the double Q-learning procedure, the learning rate is set to 0.8, and γ in the discount factor is set to 0.9. The weights of hop count on LEO and MEO satellite orbits are set to 0.3 and 0.7, respectively. The delivery rate, average delay, and overhead ratio are used to evaluate the routing strategies at different data packet generation intervals with different failure probabilities.

5.2. Simulation Results. We compare the proposed adaptive routing strategy based on improved double Q-learning for S-IoT (ARSIDQL) with the adaptive routing strategy based on original double Q-learning (ARSDQL), the adaptive routing strategy based on original Q-learning (ARSQL), the Spray-and-Wait routing strategy [11], MLRSP [13], and ARSA [37] in terms of delivery rate, average delay, and overhead ratio with different failure probabilities.

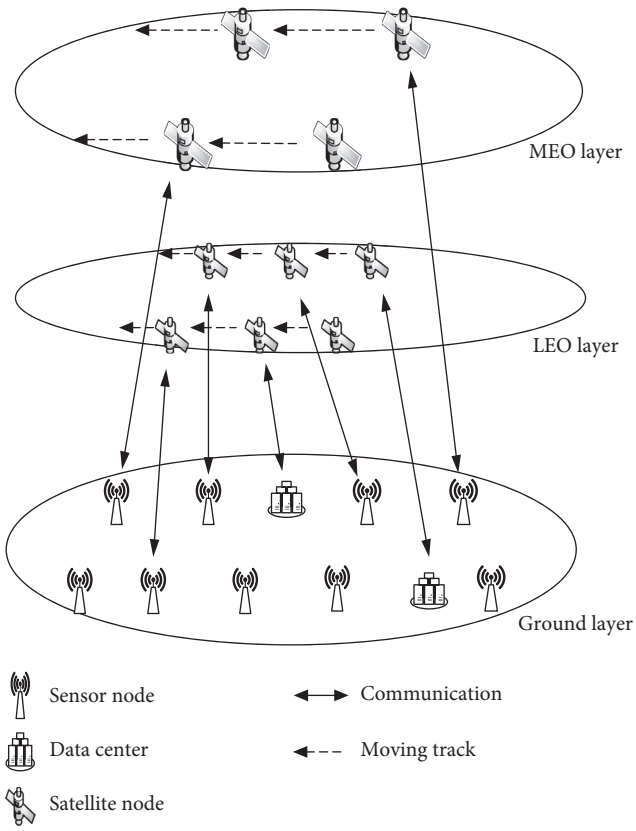

Figure 2: S-IoT model.

Table 1: Node parameters.

Layer	LEO layer	MEO layer	Ground layer
Constellation	Globalstar	GPS	Distributed evenly
Orbit numbers	8	6	/
Node numbers	48	24	110
Height	189 Km	20200 Km	0 Km

Table 2: Network environment parameters.

Parameters	Values
Buffer size	35 Mb
Transmission speed	250 Kb
Data packet generation intervals	10–50 s
Data packet size	500 Kb–1 Mb
Data packet TTL	3600 s

5.2.1. Delivery Rate. Figure 3 shows the comparison of delivery rates achieved by all routing strategies at different data packet generation intervals with different failure probabilities. On the whole, MLRSP achieves the lowest delivery rate, since MLRSP calculates the encountering probability of each node and copies data packets only to the node with the largest encountering probability. However, MLRSP fails to take into account the data packet loss caused by the high buffer occupancy of nodes. The delivery rate of Spray-and-Wait is higher than that of MLRSP by taking the advantage of flood. To be specific, the data packets are diffused into several copies to increase the probability of data packets arriving at the destination node. The delivery rates of ARSA and ARSQL are higher than that of Spray-and-Wait; since the Q-learning algorithm is self-learning and self-adaptive, ARSA and ARSQL can explore a suitable path in a highly dynamic environment. However, the encountering probabilities of nodes in S-IoT are fixed. ARSA considers the encountering probability, resulting in lower delivery rate than ARSQL. The delivery rate of ARSDQL is higher than that of ARSQL. The reason is that ARSDQL decouples data packet forwarding from the Q value evaluation of this forwarding, and the node used for forwarding is determined depending on the mixed Q value without positive deviation. Built upon ARSDQL, ARSIDQL incorporates the congestion degree and node status. Hence, data packets are more likely to arrive at the destination node before arriving at the end of their TTLs, so ARSIDQL achieves the highest delivery rate.

With the increase of the data packet generation interval, the delivery rate of MLRSP improves significantly. Since there are a large number of data packets in the network at low generation interval, the buffer size of each node is limited, causing many data packet losses. The delivery rate of Spray-and-Wait remains unchanged, since Spray-and-Wait limits the number of data packet replicas to reduce the buffer occupancy rate and further the number of data packet losses. The delivery rates of ARSA, ARSQL, and ARSDQL are relatively stable, because they can find the best action in the current state depending on the Q value through interacting with the environment. The delivery rates of ARSIDQL are relatively stable and high at low generation interval. This strategy can adapt to the buffer occupancy and forward data packets to nodes with low buffer occupancy rates to reduce the number of data packet losses and to achieve good performance.

With the increase of the failure probability, the delivery rate of MLRSP decreases. Since MLRSP forwards data packets depending on the encountering probability even if node failures have taken place, MLRSP cannot adapt to the changes of topology structure. The delivery rate of Spray-and-Wait decreases slightly. Because the data packets are diffused into some copies, the delivery rate can be guaranteed with insignificant degradation. The delivery rates of ARSA, ARSQL, ARSDQL, and ARSIDQL are relatively stable and high even with high failure probabilities owing to their abilities of self-learning. Since the Q value of forwarding data packets to the failed node would be smaller, these strategies can avoid forwarding data packets to the failed node and thus can adapt to the dynamic topology structure.

5.2.2. Average Delay. Figure 4 shows the comparison of average delays of routing strategies at different data packet generation intervals with different failure probabilities. On the whole, the average delay of Spray-and-Wait is the highest, due to the fact that in this strategy each node can only move and cannot forward data packets until it encounters the destination node in the waiting phase. The

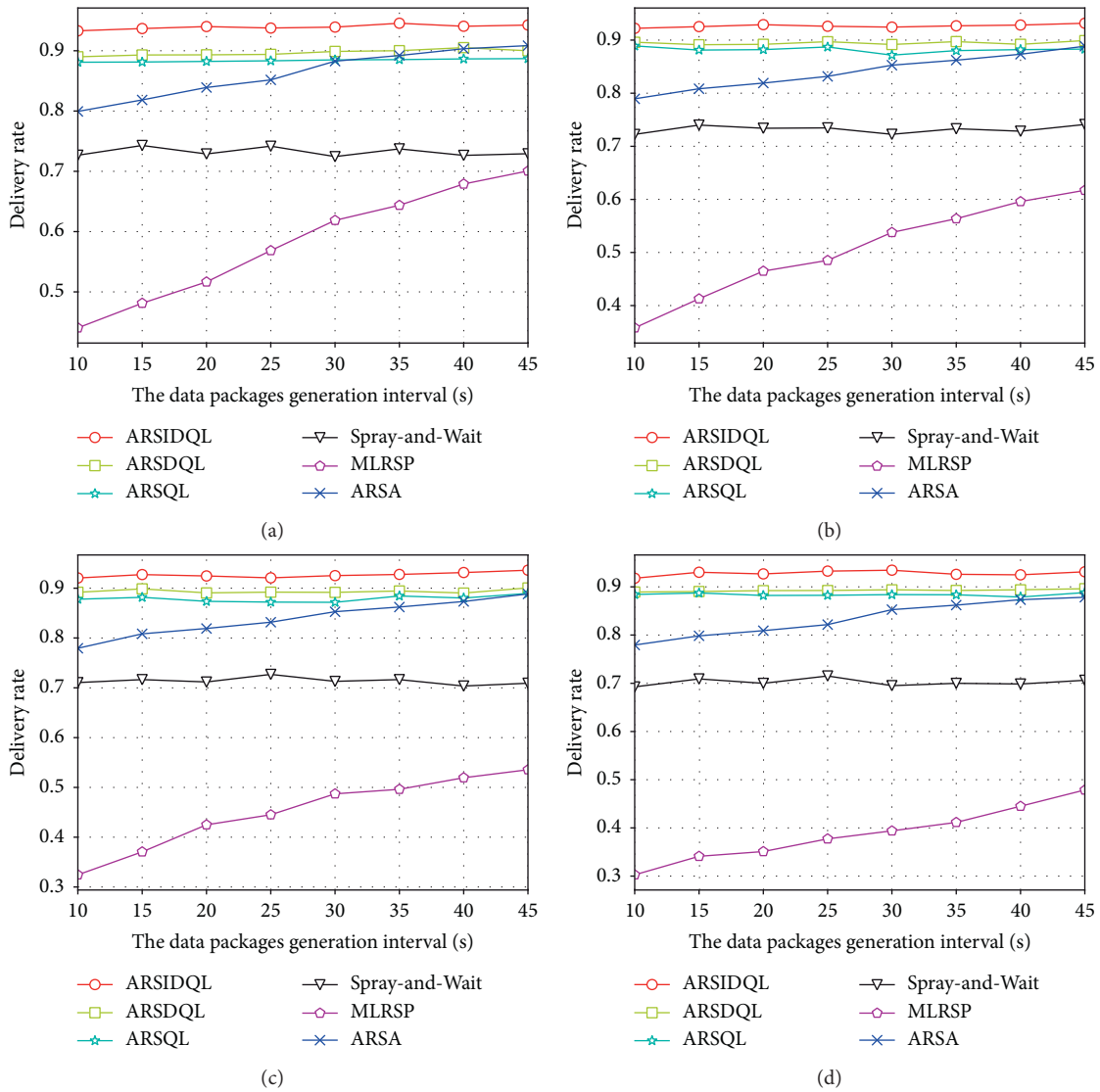

Figure 3: Delivery rates with different failure probabilities. (a) 0% failure probability. (b) 10% failure probability. (c) 20% failure probability. (d) 30% failure probability.

average delay of MLRSP is also high, since MLRSP only takes into account the encountering probability when each node forwards data packets. However, MLRSP cannot find an appropriate path as the encountering probability cannot reflect the node status. ARSQL can learn by itself to find the next hop node with a relatively low average delay. The average delay of ARSA is lower than that of ARSQL, since ARSA considers the relative speed of nodes. The average delay of ARSDQL is low, since ARSDQL solves the overestimation problem through two Q values and can find the global optimal path to reduce the average delay. Built upon ARSDQL, ARSIDQL can adapt to the congestion degree and hop count to achieve the lowest average delay.

With the increase of the data packet generation interval, the total number of data packets in S-IoT decreases such that the waiting time in the buffer and the average delay of Spray-and-Wait can be reduced. The average delay of MLRSP is reduced to a greater extent. However, the large number of data packets and copies made by MLRSP in S-IoT at low packet generation interval would lead to node congestion and long waiting times in the buffer. The average delays by suing ARSA, ARSQL, ARSDQL, and ARSIDQL decrease slightly with low failure probabilities, because the total number of data packets in S-IoT decreases with the increase of the data packet generation interval. In the cases of high failure probabilities, the average delays remain stable, since these strategies have found a suitable path at low packet generation interval. In addition, the high failure probability leads to fewer nodes in the network. The change of generation interval no longer affects the average delay.

With the increase of the failure probability, the average delays of Spray-and-Wait and MLRSP get worsen accordingly, due to the fact that these strategies cannot make adjustments to failed nodes in a timely fashion. The average delays of ARSA, ARSQL, ARSDQL, and ARSIDQL also degrade slightly. The good thing is that, because the update

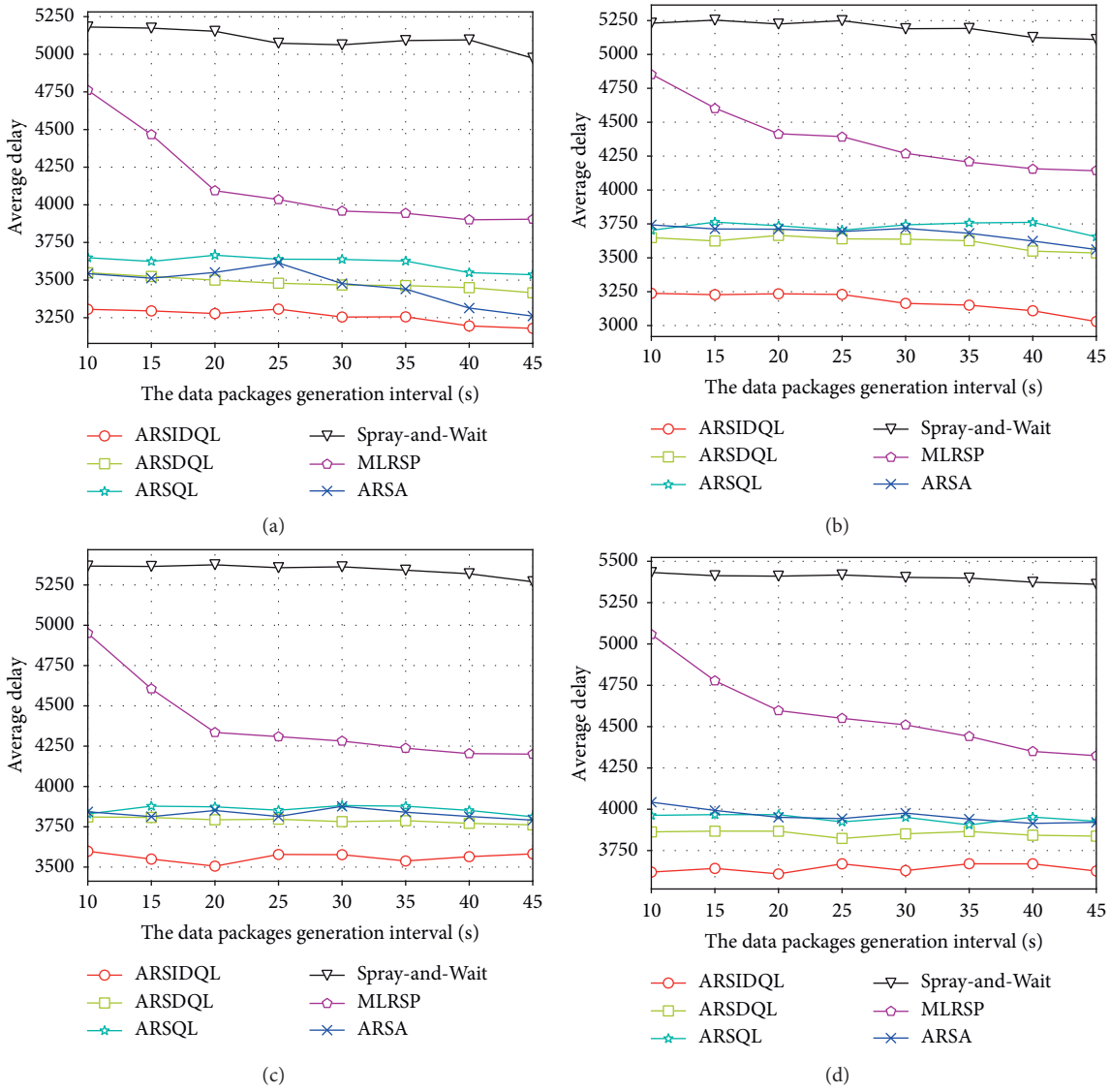

Figure 4: Average delays with different failure probabilities. (a) 0% failure probability. (b) 10% failure probability. (c) 20% failure probability. (d) 30% failure probability.

of the Q value reflects the changes of topology structure, the routes by using these strategies can bypass failed nodes and the degradation of average delay is not significant.

5.2.3. Overhead Ratio. Figure 5 shows the comparison of overhead ratios of various routing strategies at different data packet generation intervals with different failure probabilities. The overhead ratio depending on the forwarding time reflects the energy efficiency. On the whole, the overhead ratio of Spray-and-Wait is the highest. As a flood-based routing strategy, Spray-and-Wait increases the forwarding time in case of a large number of copies of data packets in the network. Compared with Spray-and-Wait, MLRSP achieves a lower overhead ratio, since MLRSP copies data packets only to the node with the largest encountering probability to restrict the forwarding time. ARSQL and ARSDQL, which are not flood-based routing strategies, result in less forwarding time due to fewer data packets in the network. The overhead ratio of ARSA is lower than that of ARSDQL. The reason is that ARSA reduces the forwarding time since it considers multiple destination nodes as the same virtual destination. Built upon ARSDQL, ARSIDQL takes the hop count and node status into consideration, thus achieving the lowest overhead ratio.

With the increase of the data packet generation interval, the overhead ratios of all strategies decrease slightly. As the total number of data packets in S-IoT decreases as data packet generation interval increases, the forwarding time is reduced. As a consequence, lower energy consumption and overhead ratio are achieved.

With the increase of the failure probability, the overhead ratios of Spray-and-Wait and MLRSP increase. The reason is that, under the circumstance of node failures, Spray-and-Wait retransmits data packets in order to maintain a fixed number of copies, whereas MLRSP still forwards data

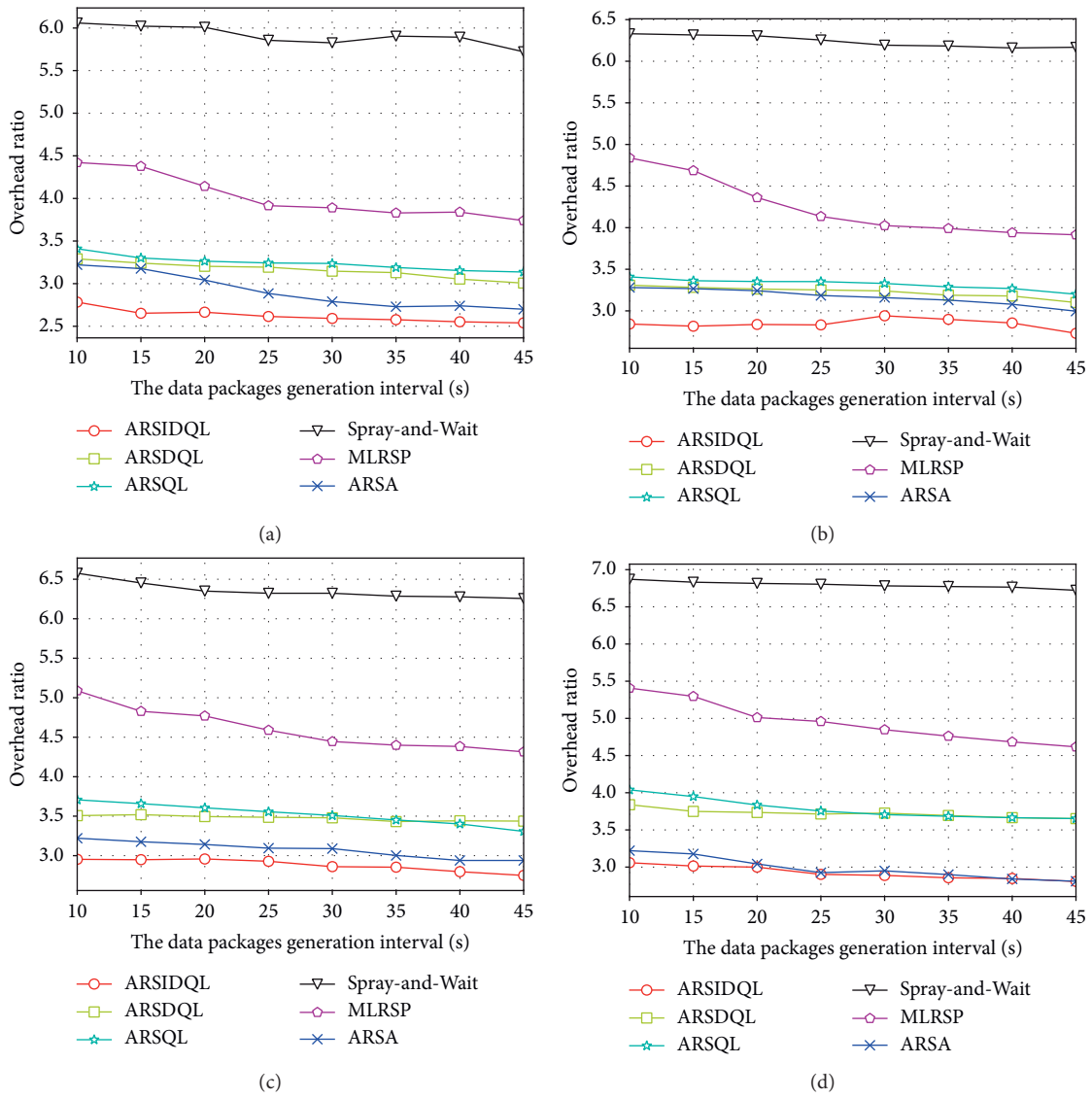

Figure 5: Overhead ratios with different failure probabilities. (a) 0% failure probability. (b) 10% failure probability. (c) 20% failure probability. (d) 30% failure probability.

packets depending on the encountering probability. The overhead ratios of ARSA, ARSQL, ARSDQL, and ARSIDQL increase slightly. These strategies are capable of bypassing failed nodes. The bypassing procedure would inevitably lead to the increase of forwarding time, energy consumption, and overhead ratio.

In summary, compared with ARSDQL, ARSIDQL can improve the delivery rate, average delay, and overhead ratio by taking into account the congestion degree, hop count, and node status in the S-IoT model. Also, compared with ARSQL and ARSA, ARSIDQL can find the best next hop node of data packets due to the decoupling of selection and evaluation. Compared with traditional routing strategies, such as the flood-based routing strategy and the utility-based routing strategy, ARSIDQL can significantly improve the delivery rate, average delay, and overhead ratio with the integration of reinforcement learning.

6. Conclusions

S-IoT is a new mobile Internet to provide services for social networks. The routing strategy determines the communication performance of S-IoT. Traditional routing strategies cannot cope with frequent changes of topology structure and node status and cannot meet the requirement of communication security in S-IoTs. This paper proposes an adaptive routing strategy based on improved double Q-learning for S-IoT. The proposed strategy selects the next hop node of data packets relying on the mixed Q value. Moreover, in order to optimize the Q value, this paper makes improvements on the mixed Q value, the reward value, and the discount factor, respectively, based on the congestion degree, the hop count, and the node status. Simulation experiments show that the proposed strategy not only can operate efficiently and securely in complex environments but also can increase the delivery ratio and reduce the average delay and

overhead ratio. Considering the large sizes of the two Q tables due to the increasing number of nodes in S-IoT, future work can be directed toward replacing the two Q tables with two neural networks.

Data Availability

The simulated evaluation data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61972210, 61873131, 61872191, and 61803212) and the 1311 Talent Program of Nanjing University of Posts and Telecommunications.

References

[1] M. Liu, N. Qu, and J. Tang, "Signal estimation in cognitive satellite networks for satellite-based industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2062–2071, 2020.

[2] T. Wang, Y. Mei, X. Liu, J. Wang, H.-N. Dai, and Z. Wang, "Edge-based auditing method for data security in resource-constrained internet of things," *Journal of Systems Architecture*, vol. 114, p. 101971, 2021.

[3] X. Liu, M. S. Obaidat, C. Lin, T. Wang, and A. Liu, "Movement-based solutions to energy limitation in wireless sensor networks: state of the art and future trends," *IEEE Network*, vol. 35, no. 2, pp. 188–193, 2021.

[4] G. X. Zhang, X. Jie, and C. Z. Qu, "Development status and challenges of IoT for LEO satellites," *Journal of IoT*, vol. 1, no. 3, pp. 6–9, 2017.

[5] Z. Zhang, W. Zhang, and F. H. Tseng, "Satellite mobile edge computing: improving QoS of high-speed satellite-terrestrial networks using edge computing techniques," *IEEE Network*, vol. 33, no. 1, pp. 70–76, 2018.

[6] F. Wang, D. Jiang, S. Qi, C. Qiao, and L. Shi, "A dynamic resource scheduling scheme in edge computing satellite networks," *Mobile Networks and Applications*, 2020.

[7] A. B. Gabis and M. Koudil, ""NoC routing protocols–objective-based classification," *Journal of System Architecture*, vol. 66, pp. 14–32, 2016.

[8] W. Zhang, G. Han, Y. Feng, and J. Lloret, "IRPL: an energy efficient routing protocol for wireless sensor networks," *Journal of Systems Architecture*, vol. 75, pp. 35–49, 2017.

[9] Q. Li, A. Liu, T. Wang, M. Xie, and N. N. Xiong, "Pipeline slot based fast rerouting scheme for delay optimization in duty cycle based M2M communications," *Peer-to-Peer Networking and Applications*, vol. 12, no. 6, pp. 1673–1704, 2019.

[10] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," *Technical report*, Duke University, Durham, NC, USA, 2000.

[11] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking*, Philadelphia, PA, USA, August 2005.

[12] A. Lindgren and A. Doria, "Probabilistic routing in intermittently connected networks," in *Proceedings of International Workshop on Service Assurance with Partial and Intermittent Resources*, Fortaleza, Brazil, August 2004.

[13] D. K. Sharma, S. K. Dhurandher, I. Woungang, R. K. Srivastava, A. Mohananey, and J. J. P. C. Rodrigues, "A machine learning-based protocol for efficient routing in opportunistic networks," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2207–2213, 2018.

[14] G. Araniti, N. Bezirgiannidis, E. Birrane et al., "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, 2015.

[15] X. T. Gong, L. J. Sun, J. Zhou et al., "Adaptive routing strategy based on improved Q-learning for satellite internet of things," in *Proceedings of International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, Nanjing, China, December 2020.

[16] H. V. Hasselt, "Double Q-learning," in *Proceedings of Advances in Neural Information Processing Systems*, Vancouver, BC, USA, December 2010.

[17] J. Sun, Y. Zhang, Z. Wu et al., "An efficient and scalable framework for processing remotely sensed big data in cloud computing environments," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 7, pp. 4294–4308, 2019.

[18] Z. Wu, J. Sun, Y. Zhang et al., "Scheduling-guided automatic processing of massive hyperspectral image classification on cloud computing architectures," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[19] J. Zhou, J. Sun, M. Zhang, and Y. Ma, "Dependable scheduling for real-time workflows on cyber-physical cloud systems," *IEEE Transactions on Industrial Informatics*, p. 1, 2020.

[20] V. V. Gounder, R. Prakash, and H. Abu-Amara, "Routing in LEO-based satellite networks," in *Proceedings of IEEE Emerging Technologies Symposium, Wireless Communications and Systems*, Richardson, TX, USA, April 1999.

[21] R. Mauger and C. Rosenberg, "QoS guarantees for multimedia services on a TDMA-based satellite network," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 56–65, 1997.

[22] Y. Hashimoto and B. Sarikaya, "Design of ip-based routing in a LEO satellite network," in *Proceedings of International Workshop on Satellite-Based Information Servies*, Dallas, TX, USA, 1998.

[23] S. Wang, C. Fan, C. Deng, W. Gu, Q. Sun, and F. Yang, "A-GR: a novel geographical routing protocol for AANETs," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 931–937, 2013.

[24] E. Ekici, I. F. Akyildiz, and M. D. Bender, "A distributed routing algorithm for datagram traffic in LEO satellite networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 137–147, 2001.

[25] Y. Yang, M. Xu, D. Wang, and Y. Wang, "Towards energy-efficient routing in satellite networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3869–3886, 2016.

[26] M. Marchese and F. Patrone, "Energy-aware routing algorithm for DTN-Nanosatellite networks," in *Proceedings of IEEE Global Communications Conference*, Abu Dhabi, UAE, December 2018.

[27] T. Mao, B. Zhou, and Z. Xu, "A multi-QoS optimization routing for LEO/MEO satellite IP networks," *Journal of Multiple Sclerosis*, vol. 9, no. 4, Article ID 576, 2014.

[28] Q. Huang, B. S. Yeo, and P. Y. Kong, "A routing algorithm to provide end-to-end delay guarantee in low earth orbit satellite networks," in *Proceedings of IEEE Vehicular Technology Conference*, Los Angeles, CA, USA, September 2004.

[29] H. Xu, F. Huang, and S. Wu, "A distributed QoS routing based on ant algorithm for LEO satellite network," *Journal of Electronics (China)*, vol. 24, no. 6, pp. 765–771, 2007.

[30] Y. Z. Qin, S. G. Shu, and W. Ye, "Distributed data storage and transmission technology of the space internet of things," *Journal of IoT*, vol. 2, no. 4, pp. 26–34, 2018.

[31] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[32] X. H. Deng, J. Li, and E. L. Liu, "Task allocation algorithm and optimization model on edge collaboration," *Journal of System Architecture*, vol. 110, pp. 1–14, 2020.

[33] Y. M. Zhao, X. H. Wang, and Y. T. Jiang, "On hardware-trojan-assisted power budgeting system attack targeting many core systems," *Journal of System Architecture*, vol. 109, pp. 1–11, 2020.

[34] A. Elwhishi, P. H. Ho, and K. Naik, "ARBR: Adaptive reinforcement-based routing for DTN," in *Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Ontario, Canada, 2010.

[35] R. Plate and C. Wakayama, "Utilizing kinematics and selective sweeping in reinforcement learning-based routing algorithms for underwater networks," *Ad Hoc Networks*, vol. 34, pp. 105–120, 2015.

[36] V. G. Rolla and M. Curado, "A reinforcement learning-based routing for delay tolerant networks," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2243–2250, 2013.

[37] C. Wu, T. Yoshinaga, D. Bayar, and Y. Ji, "Learning for adaptive anycast in vehicular delay tolerant networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 4, pp. 1379–1388, 2019.

[38] Y. Zhang, P. Sun, and Y. Yin, "Human-like autonomous vehicle speed control by deep reinforcement learning with double Q-learning," in *Proceedings of IEEE Intelligent Vehicles Symposium*, Changshu, China, June 2018.

[39] S. Vimal, M. Khari, R. G. Crespo, L. Kalaivani, N. Dey, and M. Kaliappan, "Energy enhancement using Multiobjective Ant colony optimization with Double *Q* learning algorithm for IoT based cognitive radio networks," *Computer Communications*, vol. 154, no. 1, pp. 481–490, 2020.

[40] Q. Zhang, M. Lin, and L. T. Yang, "A double deep Q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 739–749, 2018.