Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6685722, 21 pages
https://doi.org/10.1155/2021/6685722

Research Article

Data Transmission Evaluation and Allocation Mechanism of the Optimal Routing Path: An Asynchronous Advantage Actor-Critic (A3C) Approach

Yahui Ding[1,2] **Jianli Guo**[2] **Xu Li**[2] **Xiujuan Shi**[2] **and Peng Yu**[1,2]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China*
[2]*Science and Technology on Communication Network Laboratory, The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, China*

Correspondence should be addressed to Jianli Guo; 15232132720@163.com

Received 16 November 2020; Accepted 29 July 2021; Published 3 September 2021

Academic Editor: Chunpeng Ge

The delay tolerant networks (DTN), which have special features, differ from the traditional networks and always encounter frequent disruptions in the process of transmission. In order to transmit data in DTN, lots of routing algorithms have been proposed, like "Minimum Expected Delay," "Earliest Delivery," and "Epidemic," but all the above algorithms have not taken into account the buffer management and memory usage. With the development of intelligent algorithms, Deep Reinforcement Learning (DRL) algorithm can better adapt to the above network transmission. In this paper, we firstly build optimal models based on different scenarios so as to jointly consider the behaviors and the buffer of the communication nodes, aiming to ameliorate the process of the data transmission; then, we applied the Deep Q-learning Network (DQN) and Advantage Actor-Critic (A3C) approaches in different scenarios, intending to obtain end-to-end optimal paths of services and improve the transmission performance. In the end, we compared algorithms over different parameters and find that the models build in different scenarios can achieve 30% end-to-end delay decline and 80% throughput improvement, which show that our algorithms applied in are effective and the results are reliable.

1. Introduction

Delay tolerant network (DTN) which has high delay and lower delivery rate is a newly developing network framework, aiming to realize the interconnection and asynchronous data stable transmission in hybrid environment. DTN has a wide range of application, like the sensor networks and the mobile networks, which have obtained the attention and deep research of the academic and industries.

Although DTN can be applied in many challenged scenarios, the reliability cannot be guaranteed for the characteristics of discontinuity and randomness. So many scholars have proposed plenty of routing algorithms based on "carry-store-forward" to improve the quality of transmission. The algorithms can be classified into two types of strategies; the first kind of algorithms gets better delivery rate through message copies, like the "Epidemic" algorithm forwards the data in the manner of flooding, but too many copies of the message occupy much memory and increase the overhead of network. The second kind of algorithms forwards the data through classifying, like "First Contact" chooses end-to-end paths randomly to forward data and takes no account of the priori data; "Minimum Expected Delay" uses Dijkstra algorithm to find the path if minimum delay, but which only considers the limited prior knowledge not necessarily global optimal. Although the above algorithms can provide great convenience for us, they also increase the risk if the security of an SDN (Software Defined Network (SDN)) network is compromised. So a new authentication scheme called the hidden pattern (THP) was proposed, which combines graphics password and digital challenge value to prevent multiple types of authentication attacks at the same time [1].

DTN can complete the delivery of the message in the complex environments of frequent interruption just because the nodes can store messages. But the above routing algorithms have not considered the memory management, so it is important to determine the optimal end-to-end paths considering the effective management and usage of node capacity.

So in our paper, we have did research on the DTN and forming three different scenarios when the communication links break down firstly, and then, we applied the DQN algorithm and A3C algorithm in our proposed optimal models; finally, we compared algorithms over different parameters and find that the models build in different scenarios can achieve 30% end-to-end delay decline and 80% throughput improvement.

The main innovation of this paper lies in the following:

(i) We have researched on various scenes and different actions of the nodes, build optimal models in different scenarios

(ii) We adopt the DQN algorithm and A3C algorithm in the optimal models, with the aim of optimizing the throughput of the service data

(iii) We compare the DRL algorithm with other DTN routing algorithms over different parameters

The composition of this paper is as follows. Section 1 outlines the characters of DTN and the related works. The optimal models which built over different scenarios can be found in Section 2.

Section 3 states the procedure and structure of the DQN algorithm and A3C algorithm. Section 4 gives the simulation topology and parameters. Section 5 shows the performances of the algorithms over different simulation parameters and gives the analysis results. At last, Section 6 states the conclusions and future improvements.

2. The Outline of DTN

For the existence of the Bundle Layer in DTN, it can implement store-and-forward message switching and the service of custody transfer. These two functions are described in details below.

When forwarding messages from the source node to the destination node in TCP/IP network, the messages can query route to find the path in the relay nodes and cannot be stored permanently, for this network has continuous connection to complete the transmission. But in DTN networks, the nodes can store the messages for a period of time and move by carrying the messages when meet the appropriate node to forward the message in the manner of message bundle like the Figure 1 shown.

After the nodes sending the messages to the next rely node in the form of bundles, if the node has not received the receipt confirmation information from the next node, then the node will choose an appropriate time to forward the bundle again. As Figure 2 shown, the relay node will return a receipt to the previous node, and when the relay node forwarding messages to the next hop, the next relay

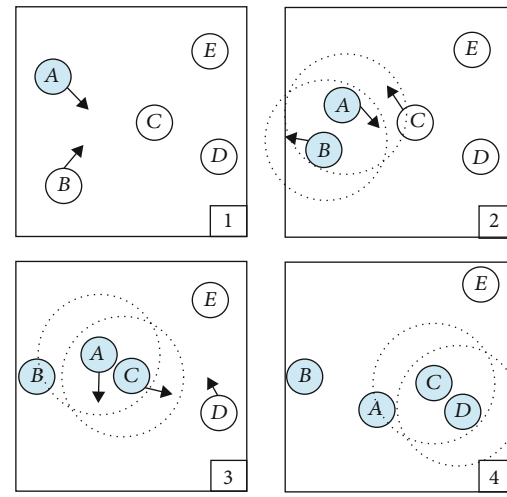

Figure 1: Carry-store-forward.

node will also send a receipt to the relay node; this procedure is carried on until the destination node receive the bundle [2]. The purpose of the custody transfer is to increase the reliability of the data transmission; only the node receives the receipt from the next hop, or the message is overdue, or the memory is full; the node deletes the message.

Ensuring DTN completes the service data transmission is important, so the scholars have studied and improved the routing algorithm in specific scenarios and proposed many routing algorithms [3].

In view of whether the infrastructure is required in the process of data forwarding, the DTN routing algorithms are composed of infrastructure-aided algorithms and non-infrastructure-aided algorithms as Figure 3 shown.

For the above problems, some recent studies [4–6] have proposed efficient cooperative caching schemes, in which data is cached at proper nodes or router nodes with limited sizes. But these papers need long time and large memory to broadcast the services. In [7], a joint optimization framework about caching, computation, and security for delay-tolerant data in M2M communication networks was proposed and adopted deep Q-network (DQN) in the model. In [8], a shortest weighted path-finding problem is formulated to identify the optimal route for secure data delivery between the source–destination pair, which can be solved by employing the Dijkstra's or Bellman–Ford algorithm.

In [9], this paper uses the reliability of travel time as the weight of path selection, and solving by Dijkstra algorithm can reflect the actual vehicle path selection more accurately. This method is a beneficial improvement to the problem of static path selection. A dynamic routing algorithm based on energy-efficient relay selection (RS), referred to as DRA-EERS, is proposed in [10] to adapt to the higher dynamics in time-varying software-defined wireless sensor networks. In [11], a solution to the data advertising problem that is based upon random linear network coding was provided; the simulation results show that the proposed approach is both highly scalable and can significantly decrease the time for advertisement message delivery. A routing architecture and algorithm based on deep neural networks was proposed in [12], which can help routers

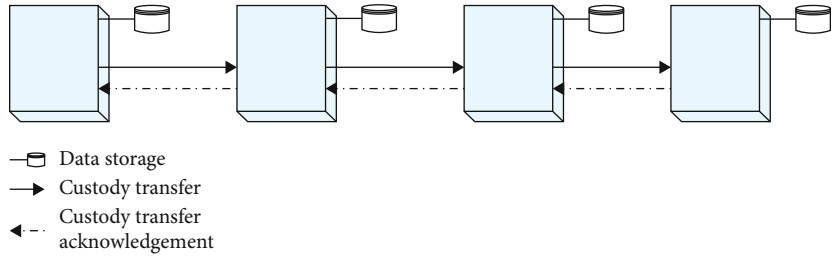

Figure 2: Custody transfer.

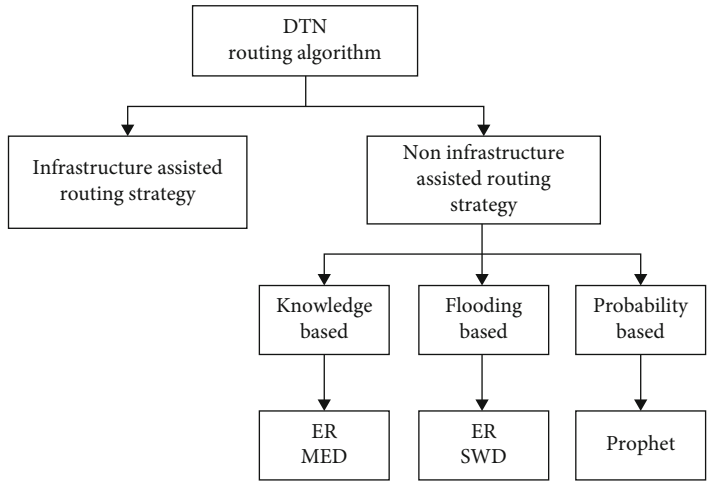

Figure 3: DTN routing algorithms.

make packet forwarding decisions based on the current conditions of its surroundings. A limited copy algorithm MPWLC based on service probability was provided in [13]; not only is the number of copies limited but the storage resources of the satellite are also taken into account to ensure reliable data transmission. The simulation results show that the proposed algorithm can effectively improve the efficiency of the network and ensure the reliable data transmission. In [14], a mathematical framework for DTN is introduced and suggested and applies it to a space network that is simulated using an orbital analysis toolkit. In [15], the problem of autonomously avoiding memory overflows in a delay tolerant node was considered, and reinforcement learning was proposed to automate buffer management given that this paper can easily measure the relative rates of data coming in and out of the DTN node.

3. Scenarios and System Model

Definition 1 (connected directed graph). We use $G = (V, E)$ to denote the connected graph if

(i) G is a directed graph

(ii) If the connections exist between node $v_i \in V$ and node $v_j \in V$, there will have $e_{i,j} \in E$

The connected directed graph can be seen in Figure 4, the communication nodes responsible for forwarding the messages, the schedule nodes responsible for scheduling of service data, the connections among the nodes are affected by the actual environments.

Assume in our graph that there exist N nodes and M links and K services; the communication nodes and schedule nodes can assemble together in $V = \{v_i, i = 1, 2, \cdots, N\}$, and the broadband links and narrow-band links can assemble together in $E = \{e_j, j = 1, 2, \cdots, M\}$; the service data $S = \{s_k, k = 1, 2, \cdots, n\}$ transmitting from the initial node v_s to the end node v_d and the end-to-end paths are expressed by $P = \{p_k, k = 1, 2, .., n\} = \{V_{pk}, E_{pk}\}$, and the time slots are expressed by $T = \{t, t = 1, 2, .., n\}$.

Due to the exceptional application environments of DQN, which always have long transmission delay and uncertain end-to-end paths, we have researched on the "carry-store-forward" and "custody transfer" mechanism and build optimal models in the following scenarios.

During service data communication, assume all service data fragments start with the shortest end-to-end path p_k. Each source node of service can send s_η fragments altogether but can only delivery one fragment $f_k(t)$ in time slot t and ascertain the transmission situations of the fragments which has been delivered before simultaneously. $I_{fk}(t)$ indicates the connection status from the node to the next-hop node in time slot t.

$$I_f^k(t) = \begin{cases} 1, & \text{the conection is interrupted,} \\ 0, & \text{the connection is connected.} \end{cases} \quad (1)$$

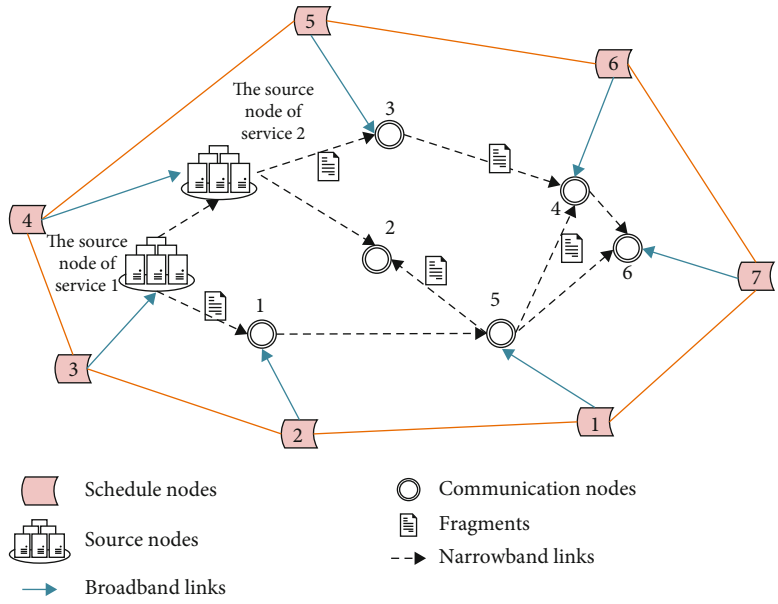

Figure 4: Connected directed graph.

Definition 2 (only consider cache scenario). We define a scenario only consider cache as Figure 5 shown, if

(i) All fragments choose the shortest paths when sending from the source node

(ii) If the fragments encounter interruption, the fragments only choose to store at the interrupted nodes and wait the nodes return to normal

In this scenario, we believe that every fragment $f_k(t)$ just chooses to store at the interrupted nodes, but at the same time, the cache data will increase the node cache processing delay $\boldsymbol{\alpha}_f(t)$ and the link interrupt waiting delay $\boldsymbol{\beta}_f(t)$; otherwise, there are no communication links interrupted $(I_{fk}(t) = 0)$; the fragments need to consider the delivery delay on the link γ_e. This process is iterated until all the fragments reached the destination and calculate the throughput. Suppose the total delay used to complete the service transmission is λ_1:

$$\lambda_1 = \sum_{k=1}^{K}\left(\sum_{l=1}^{\eta}((m_l - 1)\cdot\gamma_e) + \sum_{t=1}^{T}\left(I_{fk}(t)\cdot\left(\alpha_f(t) + \beta_f(t)\right)\right)\right). \tag{2}$$

m_k is assumed to be the interrupted node in the shortest path p_k, and when all the services accomplish the transmission, the total length of fragments that reached the terminal nodes is ω_1, and the throughput is ε_1:

$$\omega_1 = \sum_{k=1}^{K}(\eta f_k(t)), \tag{3}$$

So the optimal model is:

$$\max \varepsilon_1 = \sum_{k=1}^{K}\frac{\omega_{1\{v_1\},\{e_1\}}}{\lambda_{1\{v_1\},\{e_1\}}} \begin{cases} s.t.\forall v_i \in V_{p_k}, D_{v_i} = \left(I_{f^k}(t) * \left(\alpha_{f^k}(t) + \beta_{f^k}(t)\right)\right), & (1) \\ \forall v_i \in V_{p_k}, D_{e_j} = \gamma_{e_j}, & (2) \\ \forall v_i \in V_{p_s}, \forall e_j \in E_{p_s}, D_R^{\eta} = \sum_{l=1}^{\eta}\left(\sum_{i=1}^{m} D_{v_i} + \sum_{j=1}^{m-1} D_{e_j}\right), & (3) \\ \forall v_i \in V_{p_s}, B \geq B_R^k, & (4) \\ \forall v_i \in V_{p_s}, \sum_{n=1}^{k}\sum_{t=1}^{T} I_{f^k}(t) f^k(t). & (5) \end{cases} \tag{4}$$

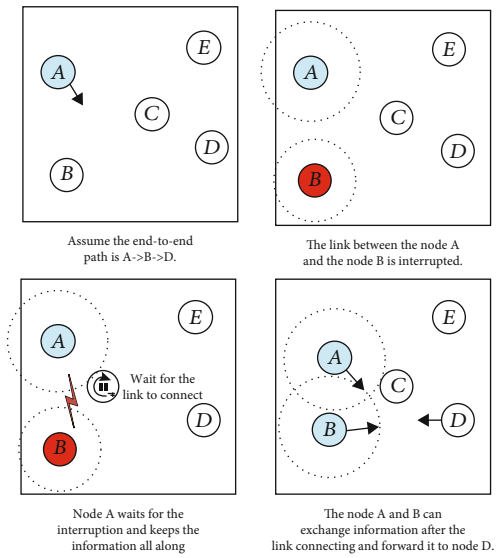

Figure 5: Only consider cache scenario.

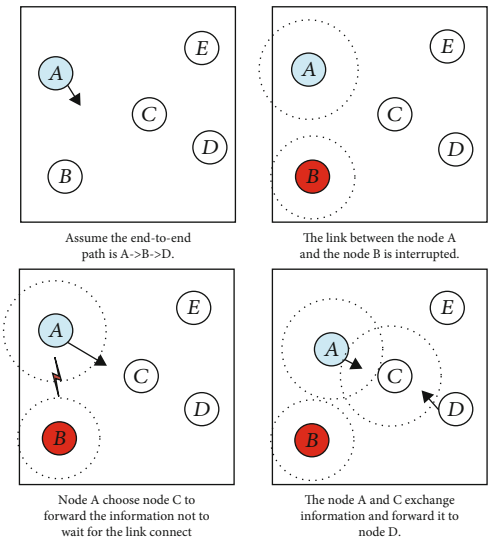

Figure 6: Only consider detour scenario.

Here, D_{v_i} denotes the total delay when the fragments store in the nodes in (1), D_{e_j} denotes the transmission delay in the path in (2), (3) states that the sum of processing delay D_{v_i} and transmission delay D_{e_j} cannot exceed the value bound s_R^k, (4) states that the bandwidth should be enough for the data transmission, and (5) denotes the maximum caching space of every node, so the total cache fragments cannot exceed the target value.

Definition 3 (only consider detour scenario). We define a scenario only considered choosing the detour path as Figure 6 shown, if

(i) All fragments choose the shortest paths when sending from the source node

(ii) If the fragments encounter interruption, the fragments only choose other available paths which have more identical nodes with the initial shortest path, so the fragments choose not to store and wait the nodes return to normal

In this scenario, we believe that every fragment $f_k(t)$ just chooses other detour paths p_s in interrupted nodes $(I_{fk}(t) = 1)$; the shortest path is p_k; otherwise, the links are connected $(I_{fk}(t) = 0)$; the fragment just transmitted along the initial shortest path and only takes into account the delivery delay γ_{e_j}. The source nodes need to continuously pay attention to the transmission till all services accomplish data delivery. We assume that the fragments run into interruption in the u_k-th node; the transmission delay of every service in the alternate path is $Ds(f_R^k(t))$. Assume the total delay of all services is λ_2.

$$\lambda = \sum_{k=1}^{K}\sum_{l=1}^{\eta}\sum_{j=1}^{m}\left(I_{f^k}(t)(j_k - 1)\gamma_e + D_R^d\left(f^k(t)\right)\right), \tag{5}$$

when all the services accomplish the transmission, the total length of fragments that reached the terminal nodes is ω_2 as formula (3), and the throughput is ε_2, so the optimal model is

$$\max \varepsilon_2 = \sum_{k=1}^{K}\left(\frac{\omega_{2\{v_i\},\{e_j\}}}{\lambda_{2\{v_i\},\{e_j\}}}\right) \begin{cases} s.t.\forall v_i \in V_{p_s}, \forall e_j \in E_{p_s}, D_R^s = \sum_{i=1}^{u}\left(1 - I_{f^k}(t)\right)\gamma_{e_i}, & (1) \\ \forall v_i \in V_{p_s}, \forall e_j \in E_{p_s}, \forall v_i \in V_{p_d}, \forall e_j \in E_{p_d}, D_R^s + D_R^d \le s_R^k, & (2) \\ \forall v_i \in V_{p_s}, B \ge B_R^s, & (3) \\ \forall v_i \in V_{p_d}, B \ge B_R^d. & (4) \end{cases} \tag{6}$$

Here, (1) states the transmission delay D_R^k in the initial shortest path, D_R^s denotes the transmission delay of every service in the alternate path, (2) states that the sum of transmission of the entire path cannot be exceed the target value s_R^k, and (3) and (4) state that the bandwidth of the entire path should be enough for the data transmission.

Definition 4 (comprehensive scenario). We define a scenario comprehensive the end-to-end path as Figure 7 shown, if

(i) All fragments choose the shortest paths when sending from the source node

(ii) If the fragments encounter interruption, the fragments jointly consider choosing other available paths which have more identical nodes with the initial shortest path or storing at the interrupted nodes, at last choose a path has the minimum end-to-end delay after comparing the above circumstances

In this scenario, we believe that every fragment $f_k(t)$ takes both the storage and the detour paths into account; $U_{f(t)}$ indicates the choice of the fragment. If the fragments chooses to wait at the nodes $(U_{f(t)} = 1)$, which generates the waiting and transmitting delay λ_1, which denotes by $\lambda_1 = \sum_{k=1}^{K}\sum_{j=1}^{\eta}(m_j - 1)\cdot\gamma_e) + \sum_{j=1}^{m^k}(I_{f(t)})(\alpha_{f(t)} + \beta_{f(t)}))$; if the fragment chooses a detour path $(U_{fk}(t) = 0)$ and tries to ensure that the available paths have more identical nodes with the initial shortest path and the total delivery delay λ_2 is $\lambda_2 = \sum_{k=1}^{K}(\sum_{l=1}^{\eta}(\sum_{j=1}^{m}(I_{f(t)}(j_k - 1)\cdot\gamma_{e_j} + D_R^s f(t))))$, the fragment will compare the above delays and choose the path that has minimum delay. The source nodes need to continuously pay attention to the transmission till all services accomplish data delivery. Assume the total delay of all services is λ_3.

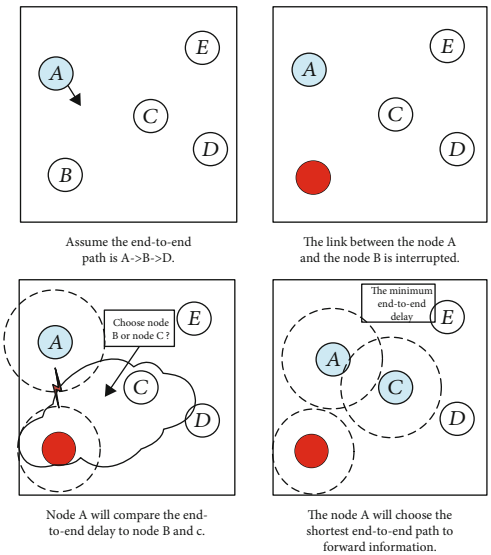

Figure 7: Comprehensive scenario.

$$\lambda_3 = \sum_{n=1}^{k}\sum_{l=1}^{\eta} J_{v_i}\left(I_{f^k}(t)T_1 + \left(1 - I_{f^k}(t)\right)T_2 + \left(1 - J_{v_i}\right)(m_k - 1)\gamma_e\right), \tag{7}$$

when all the services accomplish the transmission, the total length of fragments that reached the terminal nodes is as formula (3), so the optimal model is as follows:

$$\max \varepsilon_3 = \sum_{k=1}^{K}\frac{\omega_{3\{v_i\},\{e_j\}}}{\lambda_{3\{v_i\},\{e_j\}}} \begin{cases} s.t.\forall v_i \in V_p, D_{v_i} = \alpha_{f^k}(t) + \beta_{f^k}(t), & (1) \\ \forall e_j \in E_{p_s}, D_{e_i} = \sum_{u=1}^{u}\gamma_e, & (2) \\ \forall v_i \in V_{p_s}, \forall e_j \in E_{p_s}, D_R^\eta = \sum_{l=1}^{\eta}\left(\sum_{i=1}^{m} D_{v_i} + \sum_{j=1}^{m-1} D_{e_j}\right) \le s_R^k, & (3) \\ \forall v_i \in V_{p_s}, \forall e_j \in E_{p_s}, \forall v_i \in V_{p_d}, \forall e_j \in E_{p_d}, D_R^s + D_R^d \le s_R^k, & (4) \\ \forall v_i \in V_{p_s}, B \ge B_R^s, & (5) \\ \forall v_i \in V_{p_d}, B \ge B_R^d, & (6) \\ \forall v_i \in V_{p_s}, \sum_{n=1}^{k}\sum_{l=1}^{\eta} U_{f^k}(t)\cdot f_k(t) \le C_{v_i}. & (7) \end{cases} \tag{8}$$

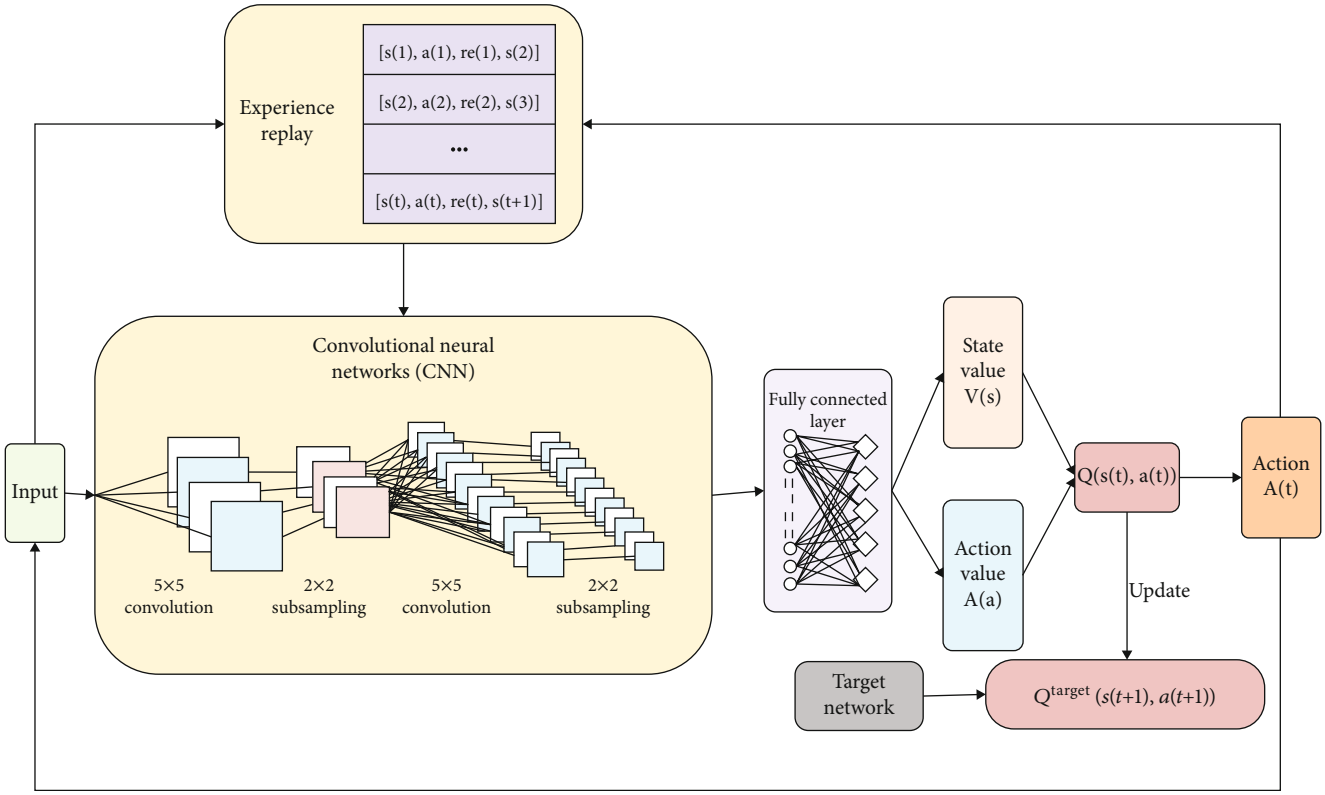

Figure 8: The work process of DQN algorithm.

Input: The initial locations of the service nodes
Output: The optimal path and throughput of every service

Initialize the topology of all the nodes and the start and end nodes of every service;
Initialize thread step counter $t \longleftarrow 1$;
while $T <= T_{\max}$ **do**
  Reset gradients: $d\theta \longleftarrow 0$ and $d\theta_v \longleftarrow 0$;
  Synchronize thread-specific parameters $\theta^1 = \theta$ and $t_{start} = t$;
  Get state s_t, that is the start node of everyservice;
  while *terminal* s_t *not is the end node of every service or* $t - t_{start} != t_{\max}$ **do**
    Perform a_t that is the next-hop according to policy $\pi(a_t \mid s_t; \theta^1)$;
    If all the constraints in models are satisfied, in consideration of the sequence of the fragments stay at the interrupted node and according to the choice of every fragment, like continue to store at the node or choose other detour paths, which can build different scenarios and have various the next-hop s_{t+1} and reward r_t;
    $t \longleftarrow t + 1$;
    $T \longleftarrow T + 1$;
  end
  Set

$$R = \begin{cases} \mathbf{f}_0 & \text{for terminal } s_t \\ V(s_t, \theta_v^1) & \text{for non} - \text{terminal } s_t \text{ or Bootstrap from last state} \end{cases}$$

  for $i = t - 1$ *to* t_{start} **do**
    $R \longleftarrow r_i + \gamma R$;
    Accumulate gradients wrt $\theta^1 : d\theta \longleftarrow d\theta + R - V(s_t; \theta_v) 6_{\theta 1} \log \pi(Ai \mid s_t; \theta)$;
    Accumulate gradients wrt $\theta_v^1 : d\theta_v \longleftarrow d\theta_v + \partial(R - V(s_t; \theta_v)^2)/\partial\theta$;
  end
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$;
end

Algorithm 1: Data transmission evaluation of the optimal routing path with A3C algorithm.

Input: The initial locations of the service nodes
Output: The optimal path and throughput of every service
Initialize replay memory D to capacity N;
Initialize action-value function Q with random weights θ;
Initialize target action-value function $\hat{Q}$ with weights $\theta - = \theta$;
for $episode = 1$ *to* N **do**
  Initialize the topology of all the nodes;
  Get the initial state X_k of all nodes and the distance between nodes;
  Set sequence $s_1 \longleftarrow e_1$,preprocess $\varphi_1 \longleftarrow \varphi(s_1)$;
  for $t = 1$ *to* T **do**
    Select a random action a_t for every node i with probability ϵ otherwise select $a_t = \max aQ(\varphi(st), a; Q)$;
    Execute action a_t in emulator and observe reward r_t;
    If all the constraints in models are satisfied, in consideration of the sequence of the fragments stay at the interrupted node and according to the choice of every fragment, like continue to store at the node or choose other detour paths, which can build different scenarios and have various the next-hop, Set $st + 1 \longleftarrow (st, at, et + 1)$ and preprocess $\varphi_{t+1} = \varphi(s_t + 1)$, otherwise go back to step 7; 13 Store transition $(\varphi_t, a_t, r_t, \varphi_{t+1})$ in D;
    Sample random mini batch of transitions $(\varphi_t, a_t, r_t, \varphi_{t+1})$ from D;
    Set

$$y_j = \begin{cases} \mathbf{fr}_j & \text{stops at step j + 1} \\ r_j + \gamma \max_a^1 hat(Q)(\varphi_{j+1}, a^1; \theta -) & \text{Otherwise} \end{cases}$$

    Perform a gradient descent step on $y_j - Q(\varphi_j, a_j; \theta)^2$ with respect to the network parameters θ;
    Every C step do reset $\widehat{Q} = Q$;
  end
end

Algorithm 2: Data transmission evaluation of the optimal routing path with DQN algorithm.

Here, D_{v_i} denotes the total delay when the fragments store in the nodes in (1), D_{e_j} denotes the transmission delay in the path in (2), D^s indicates the transmission delay of the alternate path in (3), the sum of transmission of the entire path cannot be exceed the target value s_R^k, (4) states that the sum of processing delay D_{v_i} and transmission delay D_{e_j} cannot exceed the value bound s_R^k, (5) and (6) state that the bandwidth of the entire path should be enough for the data transmission and $B^k = \min B_{v_i}$, $\forall v_i \in V_{p_k}$, and in (7), C_{v_i} denotes the maximum caching space of every node, so the total cache fragments cannot exceed the target value.

4. DRL Algorithm Procedure and Structure

Deep Reinforcement Learning (DRL) is a special and environment-friendly machine learning method, which uses the environment feedback as input and is the learning from environment state to behavior mapping; RL can maximize the cumulative return of system behavior from the environment, mainly consisting of agents and the external environment. But traditional reinforcement learning has bottlenecks; it uses table to save every state and the Q value of every action based on state [16]. And deep Q-learning network (DQN) can adopt neural networks to solve the above problems and make the state and action as the input of neural network; it can obtain the Q value through the neural network not the table, reducing the memory consumption. In DQN, they use experience reply to avoid correlation of data samples, but every time the agent interacts with the environments needs huge memory and computing power, and experience reply can only generate data by the old policy. So, A3C uses multithread of CPU to realize the parallel actor learner for multiagent instance; each thread corresponds to different exploration strategies. This parallelization can be used to decorrelate data and replace experience replay to save storage cost.

4.1. DQN Algorithm. Deep Learning has been proved to be a powerful tool to solve nonconvex and high complexity problems and has been widely used in many ways. Reinforcement learning pays more attention to the maximal rewards over a long period time which obtained by interacting with environment and carry out the optimal action. The deep Q-learning adopts deep neural network to develop an action plan and behave well when deal with dynamic time-varying environments. So, DTN provides a promising technology for the data transmission in delay tolerant network.

With regard to the reinforcement learning, it interacts with the environment through the agent, which can inspect the environment and obtain the states (t) and then take action $a(t)$ based on the state at the time slot t. Next, the external environment observes the action taken by agent and deliveries the latest state $s(t+1)$ and the reward $r(t)$ to the agent. The above process is aimed at finding the maximal reward value by the optimal policy π^*. DQN uses neural networks to approximate the value function $Q(s, a)$, and

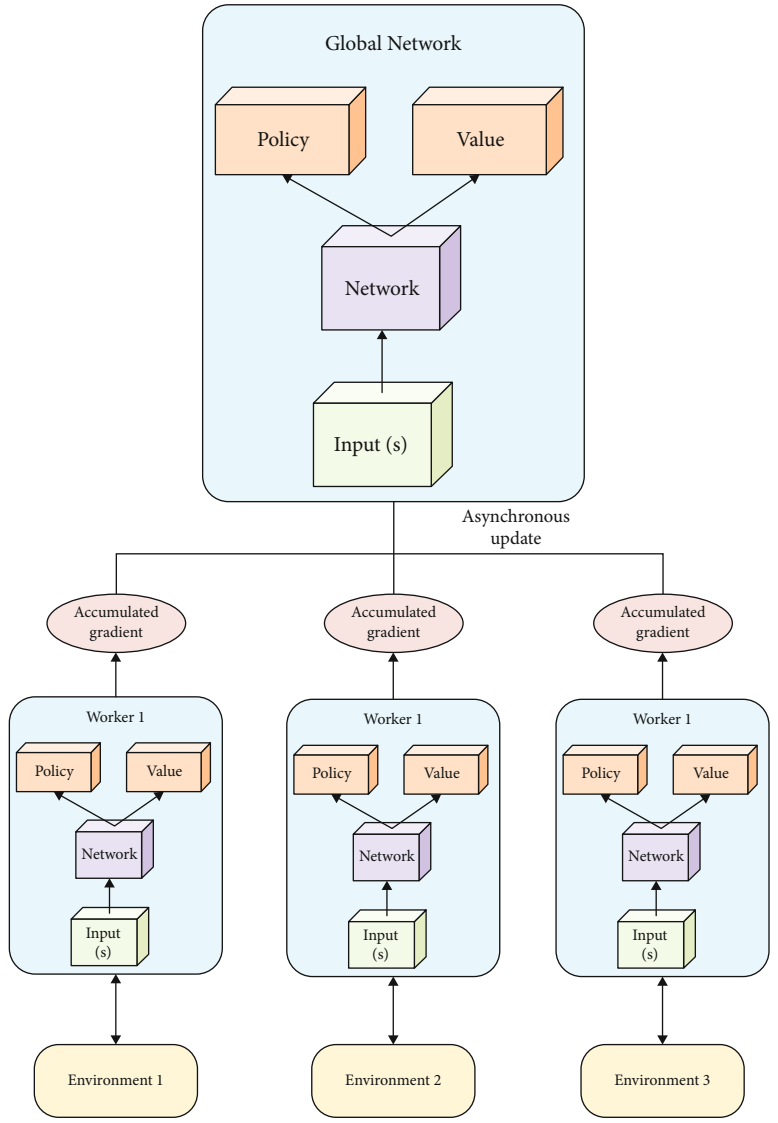

Figure 9: The work process of A3C algorithm.

the reward value $Q^*(s, a)$ can be obtained according to the following:

$$Q^*(s,a) = E^*\left[\sum_{s\in S} r\left(s, a, s^1\right) + \gamma \max Q^*\left(s^1, a^1\right)\right]. \quad (9)$$

In which the reward is computed based on the state s and the action a, γ represents the discount factor which means the future impact on the present calculation, and $E^*[\bullet]$ denotes the expectation function. Hence, DQN chooses the action which can get the Q value to the maximum.

The Q value updated at every step in DQN as follows:

$$Q(s,a) = Q(s,a) + \sigma\left(r_{t+1} + \gamma \max Q\left(s^1, a - Q(s,a)\right)\right). \quad (10)$$

In which σ denotes the leaning rate and should be in the range of [0,1], with the increase of the learning rate, the influence of the past on the present is getting smaller and smaller. The process of the DQN is shown in Figure 8.

In DQN, $<A, S, R, P>$ is a typical Quaternions [17], in which Action A indicates the action set taken by agent, State S indicates the state set observed from the environment, Reward R indicates the set of rewards value, and P denotes the deep learn mode in probability state space of agent learning. Based on the above typical Quaternions, the specific definition of DQN is shown as follows:

(1) *State Space.* $S = (X_1, X_2, \cdots, X_K)$, in which it is a vector and denotes the location of the source nodes of the k-th service; the vector has k dimensions, and only one dimension is 1

(2) *Action Space.* $A = (A_1, A_2, \cdots, A_K)$, in which it is a vector and denotes the nodes of the k-th service nodes can be connected; the vector has k dimensions, and only one dimension is 1

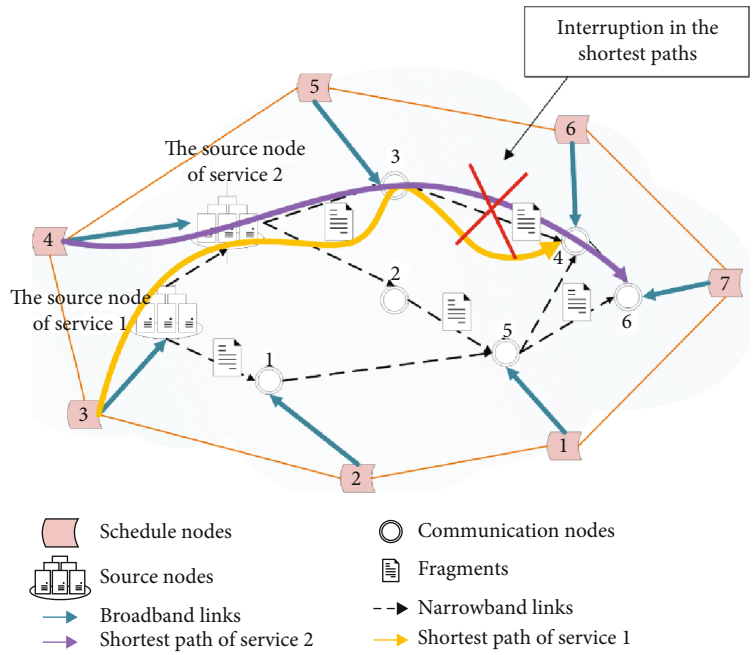

Figure 10: Network topology.

(3) *System Reward.* After each time slot t, the system will get the immediate reward $r(t)$ based on different taken action $a(t)$. In our paper, we denote the reward as the cost of the current node to other nodes; the reward $r(t)$ is smaller if the distance between the nodes is longer; otherwise, reward is bigger; $r(t)$ is shown as follows:

$$r(t) = \begin{cases} \text{distance}(i,j) + c, & \text{Distance is bigger} \\ -100, & \text{No Connectione} \\ \text{distance}(i,j)\text{-c}, & \text{Distance is smaller} \end{cases}. \quad (11)$$

With above analysis, the DQN procedures for the optimal path and throughput are shown in Algorithm 2.

4.2. A3C Algorithm. A3C uses the method of multithreading; at the same time, it interacts with the environment in multiple threads, and each thread summarizes the learning results to global net as shown. In addition, each thread regularly takes back the results of common learning from global net to guide the learning interaction between the thread and the environment. Through this method, A3C avoids the problem of strong correlation of experience playback and achieves asynchronous and concurrent learning model.

The A3C algorithm is based on the actor-critic consists of the value function $V(s_t, \theta_v)$ and policy function $\pi(a \mid s_t; \theta)$, which will not use the traditional Monte Carlo to update the parameters until the end of the scenario episode but uses temporal-difference learning to update parameters in each steps. About the actor-critic, it has two networks; one is the actor network, which is responsible for choosing the actions on account of the policy $\pi(a \mid s_t; \theta)$; the critic network is responsible for evaluating each action from the actor network. And after the actor network obtains the scores of the action, it will optimize the policy to get the maximal reward over the algorithm executions. The critic network use the following formula to calculate the score of the

Table 1: The parameters of simulation.

Parameters	Symbol	Value
Time slot	t	0.1 s
Fragment size	f	1400 bits
Maximum storage space of node	C	10 bits
Velocity of propagation	v_i	$3{\cdot}10^4$ bits
Waiting delay	β_{e_i}	*ats*
Cache rate	α_{e_i}	700 bps
Bandwidth	B	6000 bps
Learning rate for DQN	σ_D	0.001
Discount factor for DQN	γ_D	0.09
Memory pool capacity-DQN	N	2000
Minibatch-DQN	b	320
Execute steps-DQN	T	5000
Global maximum iterations-A3C	Nt_{mt}	1000
Maximum iterations in thread-A3C	Nt_{st}	10
Discount factor for A3C	γ_A	0.9
Learning rate for A3C-actor	σ_a	0.001
Learning rate for A3C-critic	σ_c	0.001
Entropy factor	ρ	0.001

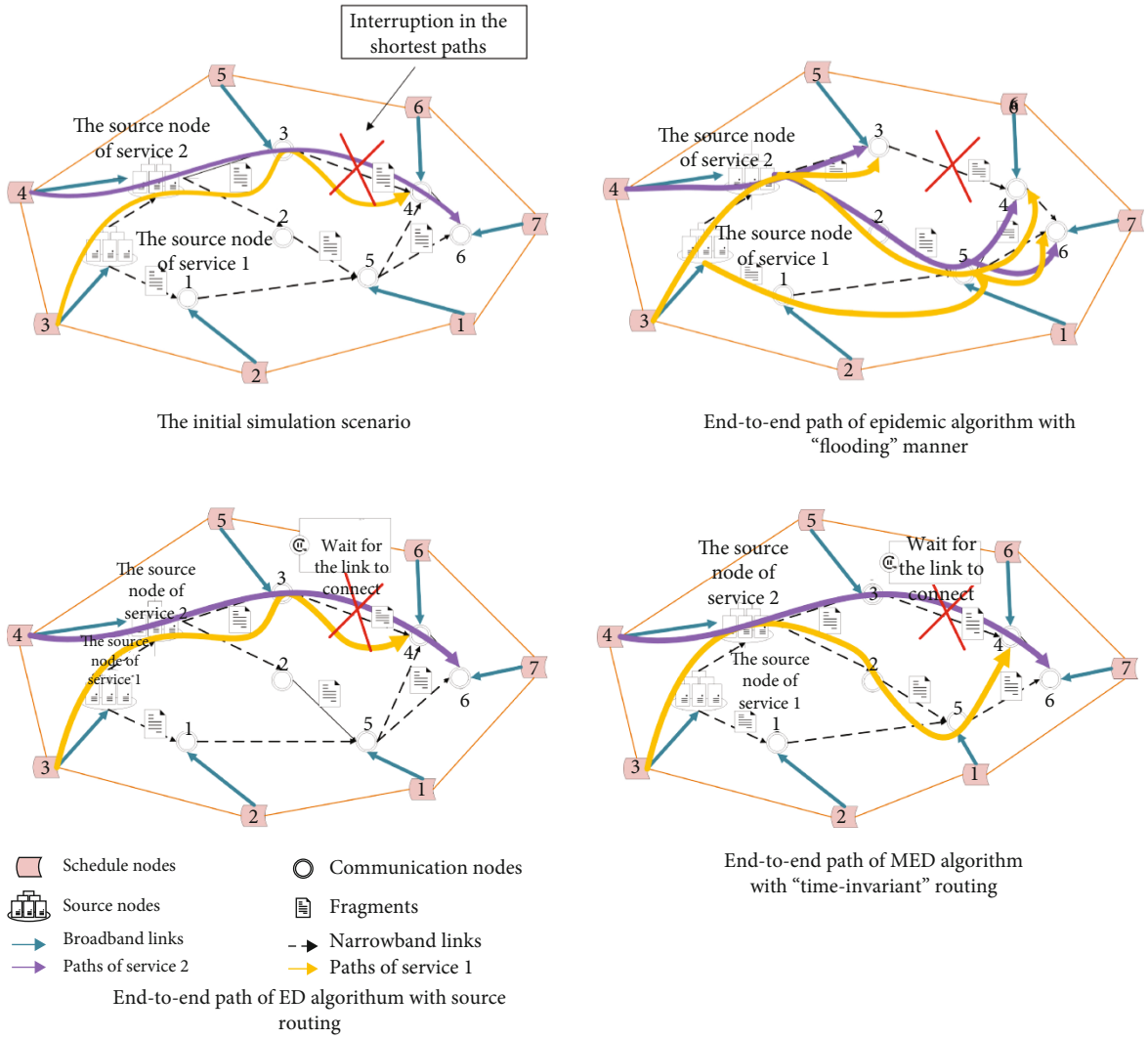

Figure 11: Process of the above algorithms.

actions

$$6\theta_v = \frac{\partial R - V(s_t;\theta_v)^2}{\partial \theta}, \tag{12}$$

in which, the R denotes the reward of taking the action a. Through calculating the gradient of the formula (4), the actor network can update the parameter θ; the gradient can be seen from the following:

$$6\theta_v = R - V(s_t;\theta_v 6_\theta \log \pi(A_i \mid s_t;\theta). \tag{13}$$

In A3C, it has defined a new function which called advantage function $A(s,a)$ that can be shown as

$$A(s,a) = R - V(st;\theta). \tag{14}$$

It expresses that if the action chosen is better than the average, then the advantage function is positive; otherwise, it is negative.

Figure 9 shows the process of A3C; it has one global network which includes the functions of actor network and critical network. And it also has n workers; each worker has the same network structure as the global neural network, and each worker will interact with the environment independently to get experience data. These workers do not interfere with each other and run independently.

Each worker and the environment interact with a certain amount of data, then calculate the gradient of the neural network loss function in its own thread; but these gradients do not update the neural network in its own thread but update the global neural network. In other words, n workers will independently use the accumulated gradient to update the common part of the neural network model parameters. Every once in a while, the thread will update the parameters of its own neural network to the parameters of the common neural network and then guide the subsequent environment interaction.

The specific description of A3C algorithm is presented in Algorithm 1.

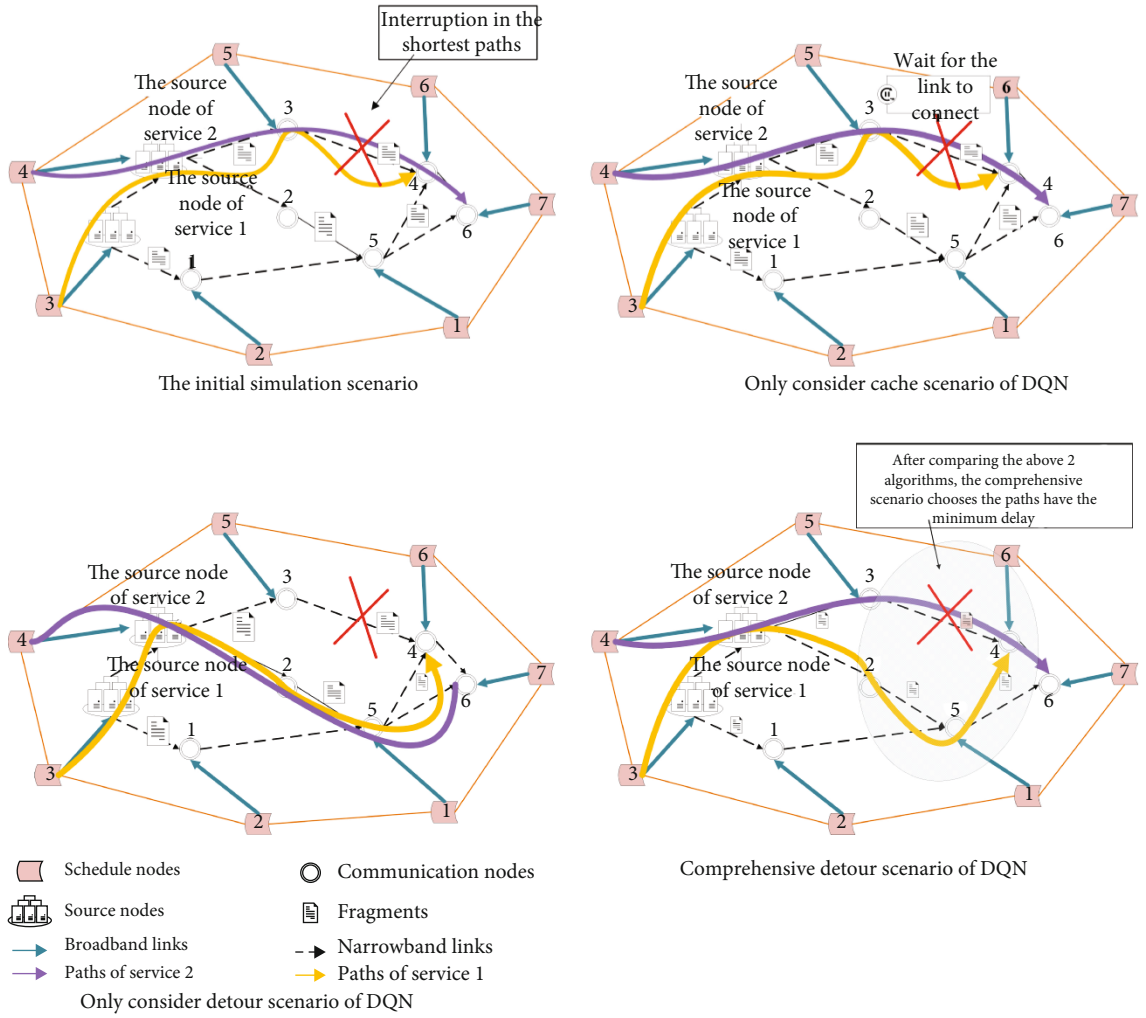

Figure 12: Process of the DQN.

5. Simulation and Analysis

5.1. Simulation Parameters. The simulation scenario is shown in Figure 10, and we assume to send two services in total, and the services are send in the different source nodes, and every service will send hundreds of data segments; the topology not only has broadband links but also has narrowband links, so the two services have different shortest paths, so when the service segments encounter disruption in the process of service data transmission, they can choose to cache in the interrupted nodes or choose other detour paths to arrive the target nodes. The purple and the yellow links denote the service's end-to-end path, respectively; the red links denote the interruption in the shortest end-to-end path of every service.

In the process of simulation, we assume the transmission rate of service data is same in every source node, and the number of data segment cache disunified in every node. And the transmission rate of service segments is the constant value $(3 * 10^4 \text{ bps})$.

In the above topology, the source nodes may send many fragments, which can result in block at the interrupted node for too many fragments store at the node, making DQN algorithm consumes more time to analyze the queue problem in the node. If the simulation topology is more complicated, the DQN algorithm needs more time to train and ensure the end-to-end path of services. So in the simulation process of the DQN algorithm, we assume the DQN network consists of three layers of neural network, and the every layer has a certain number of neurons, and the learning rate of DQN is 0.001; the discount factor to calculate the reward is 0.09 [18], and the size of memory pool which used to execute experience replay is 2000. Other specific parameters can get in Table 1; based on the above settings, the DQN algorithm can find the optimal path of services.

For the execution process of the A3C algorithm, the simulation environment is the same in every worker which is the independent kernel in computer, and we set the learning rate of actor, and critic is 0.001; the discount factor to calculate the reward is 0.09; other specific parameters can get in Table 1. Based on the parameters and the special structure of the algorithm, it has higher execution speed and performance than DQN algorithm.

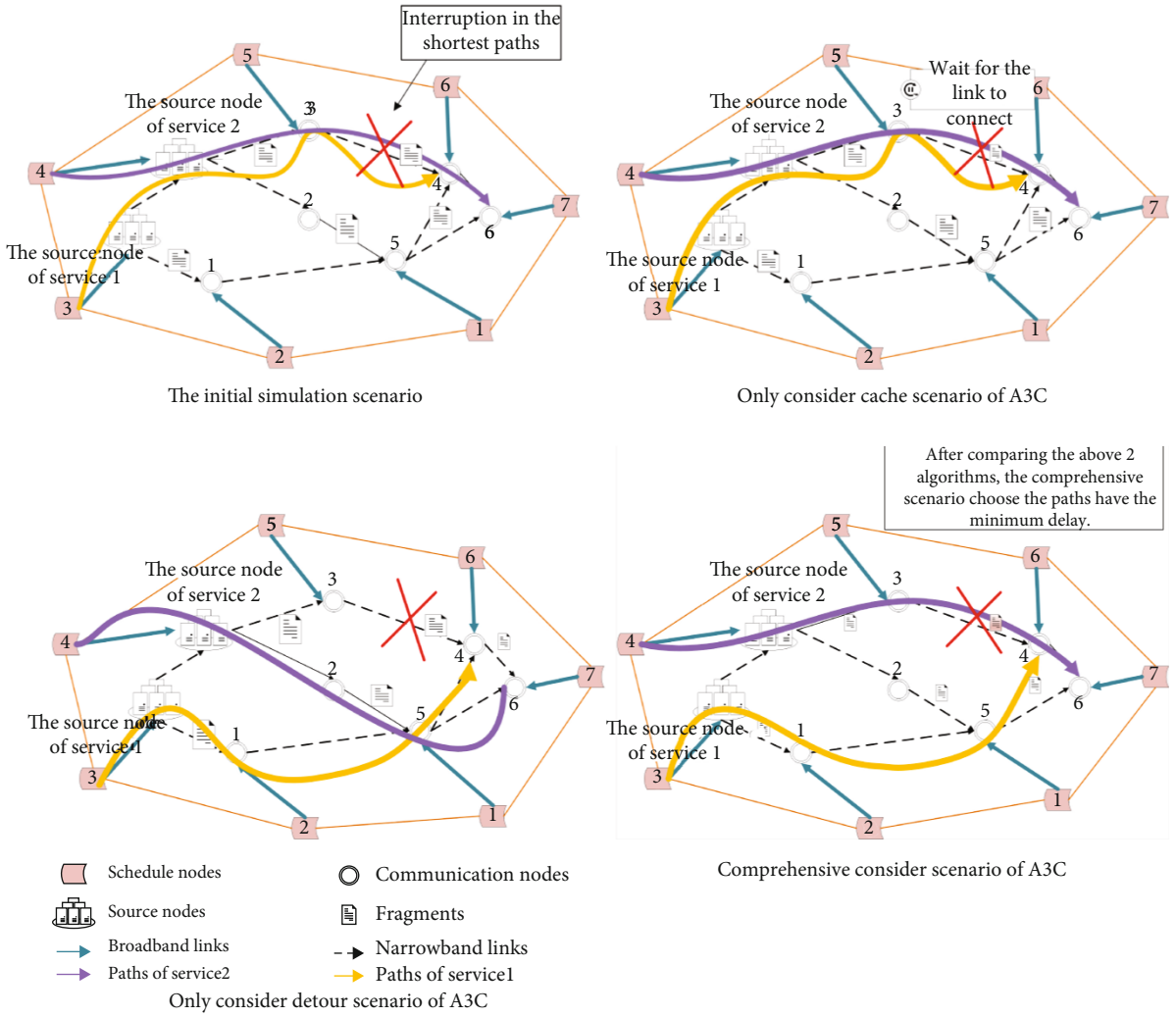

FIGURE 13: Process of the A3C.

We will compare the performance of the algorithm from the following aspects, and the parameters of simulation are shown in Table 1.

The delivery rate can be expressed as the ratio of the number of fragments reached the destination node and the number of fragments send from the source nodes, as the following shows:

$$D = \frac{\sum_{k=1}^{K} Ld_k}{Ls}, \tag{15}$$

in which Ld_k denotes the number of fragments that reach the destination node of service and Ls denotes the total number of fragments that send from the source nodes.

The end-to-end delay can be represented as delay from the time of the source nodes start to send fragments to the time of the last fragment reach the destination, as the following shows:

$$De = t_d - t_s, \tag{16}$$

in which, t_d is the time of the last fragment reach the destination, and t_s is the time of the service data start to transmit.

The throughput of service can be expressed as the number of service data successfully transmit from the source node to the destination node per unit time, as the following shows:

$$Th = \frac{\sum_{k=1}^{K} Ld_i}{De}, \tag{17}$$

in which, $\sum_{k=1}^{K} Ld_i$ is the total service data reach the destination node and De is the end-to-end delay.

The equilibrium of nodes is the number of average service carrying of every node, as the following shows:

$$N = \frac{\sum_{i=1}^{V}}{N_i p}, \tag{18}$$

in which, N_i is the number of service carrying of the i-th node, and P is the total number of nodes in the topology.

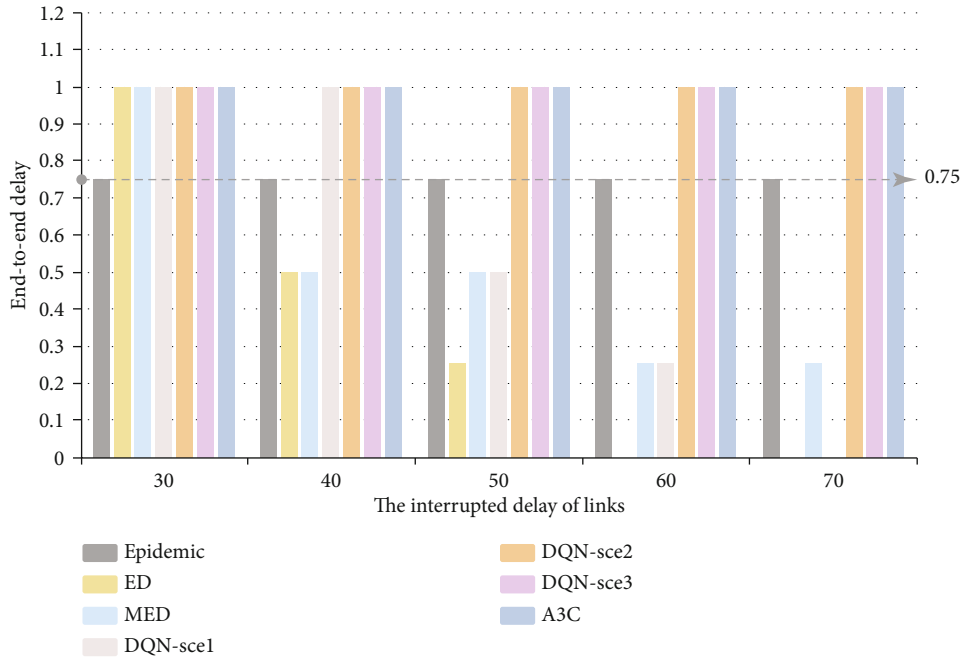

Figure 14: The delivery rate.

Table 2: The delivery rate.

Delivery rate	Algorithms						
	Epidemic	ED	MED	DQN-sce1	DQN-sce2	DQN-sce3	A3C
Interrupted delay							
30	0.75	1	1	1	1	1	1
40	0.75	0.5	0.5	1	1	1	1
50	0.75	0.25	0.5	0.5	1	1	1
60	0.75	0	0.25	0.25	1	1	1
70	0.75	0	0.25	0	1	1	1

The equilibrium of links is the number of average service carrying of every link, as the following shows:

$$L = \frac{\sum_{i=1}^{E}}{L_i Q}, \tag{19}$$

in which, L_i is the number of service carrying of the i-th link, and Q is the total number of links in the topology.

6. Simulation Results

6.1. Simulation Algorithms. The "Epidemic" algorithm belongs to the spread routing; it forwards the data in the manner of flooding; that is to say, all nodes encountered will "infect" the message [19], so there will have lots of copies of the message in the network and occupy much memory and increase the overhead of network.

But in the circumstance of sufficient network resources, the Epidemic algorithm will show faster delivery rate and be the alternative algorithm in this circumstance. If there are too many messages to transmit, some messages may be discarded, resulting in higher packet loss rate.

The ED algorithm has not taken the queue problem into consideration, and routing path is determined when the source nodes send the service data, so the ED algorithm belongs to the source routing. But when there have more fragments and queuing, it will affect the computation of weight of the ED algorithm; thus, making the computation of the source route emerges errors and cannot produce the optimal path.

The MED algorithm has taken the transmission delay, the propagation delay, and the average waiting delay into consideration; the goal of the algorithm is to find the path of minimum delay, and the path adopted is identical when the source nodes and destination nodes are same. After ensuring the path of source routing, even there has better choice, this algorithm will not change the routing choices [20], so it is just the optimal path over the limited prior knowledge and not necessarily global optimal, so the MED algorithm belongs to "time-invariant" algorithm.

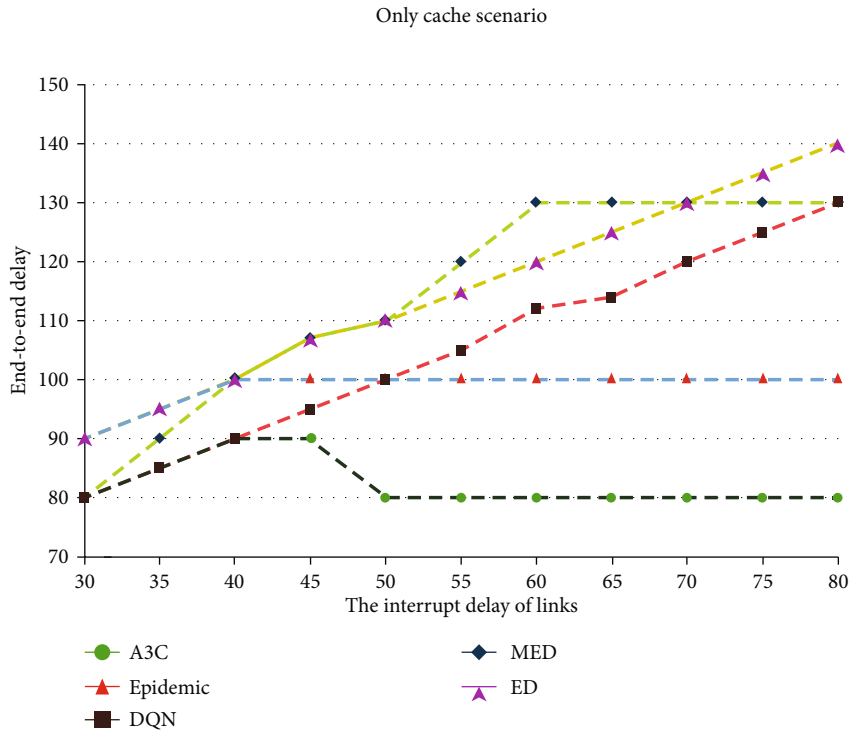

Figure 15: The end-to-end delay of service 1.

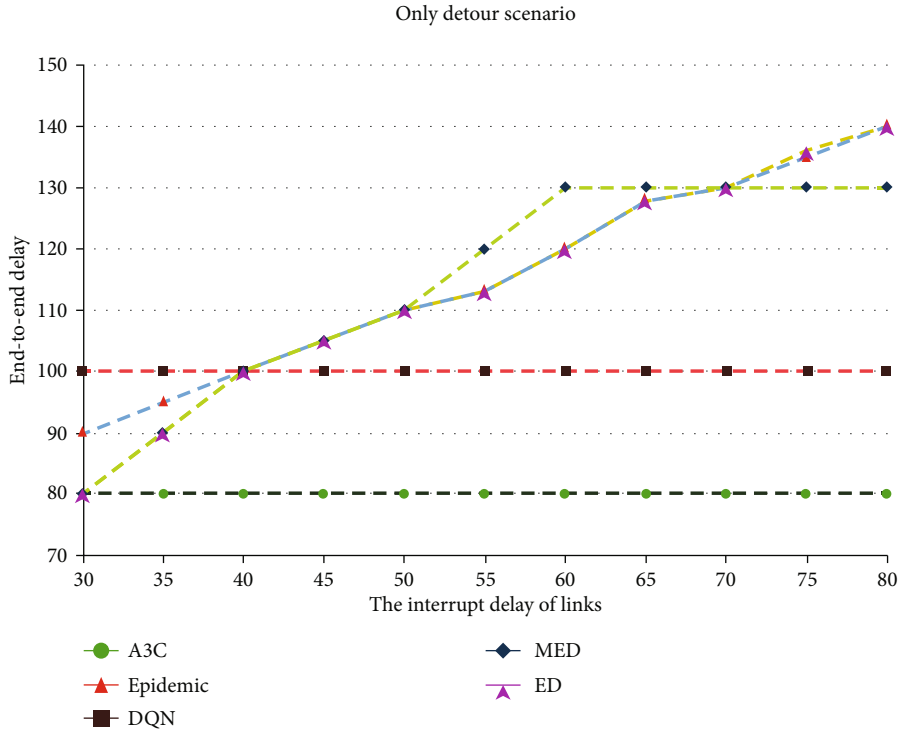

Figure 16: The end-to-end delay of service 1.

Deep Learning has been proved to be a powerful tool to solve nonconvex and high complexity problems and has been widely used in many ways. The deep Q-learning adopts deep neural network to develop an action plan and behaves well when deal with dynamic time-varying environments.

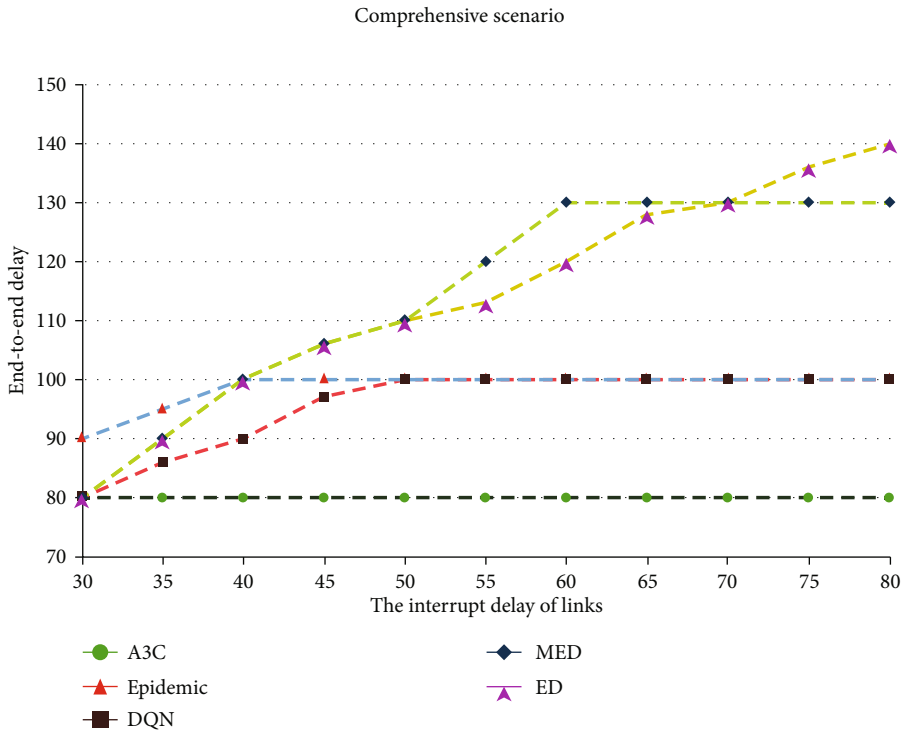

Figure 17: The end-to-end delay of service 1.

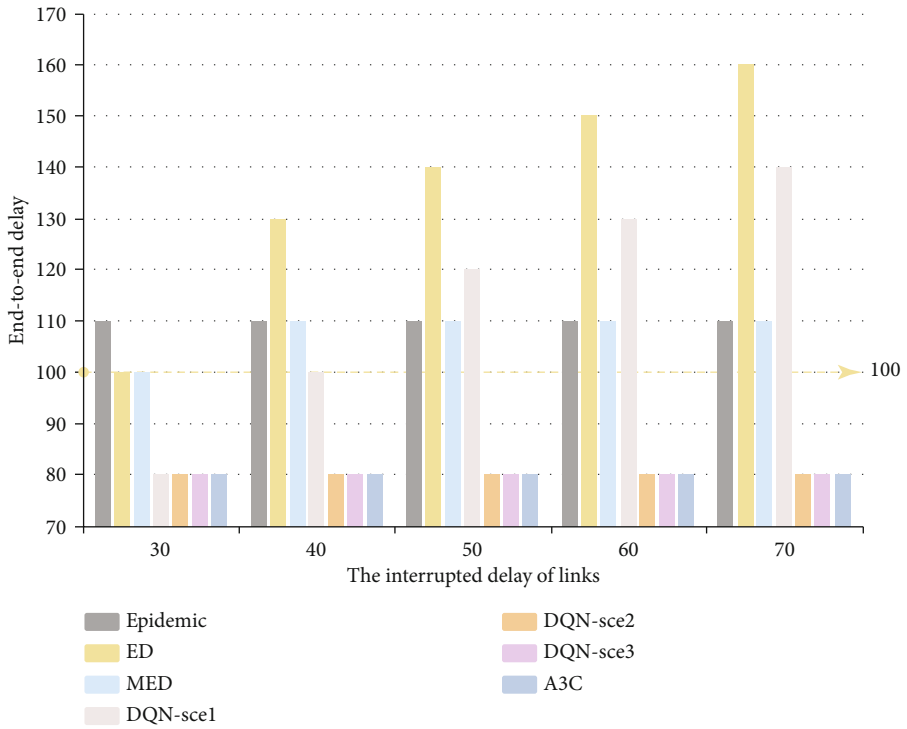

Figure 18: The end-to-end delay of service 2.

A3C uses the method of multithreading; at the same time, it interacts with the environment in multiple threads, and each thread summarizes the learning results to global net. In addition, each thread regularly takes back the results of common learning from global net to guide the learning interaction between the thread and the environment.

Table 3: The end-to-end delay of service 2.

End-to-end delay	Algorithms						
	Epidemic	ED	MED	DQN-sce1	DQN-sce2	DQN-sce3	A3C
Interrupted delay							
30	110	100	100	80	80	80	80
40	110	130	110	100	80	80	80
50	110	140	110	120	80	80	80
60	110	150	110	120	80	80	80
70	110	160	110	140	80	80	80

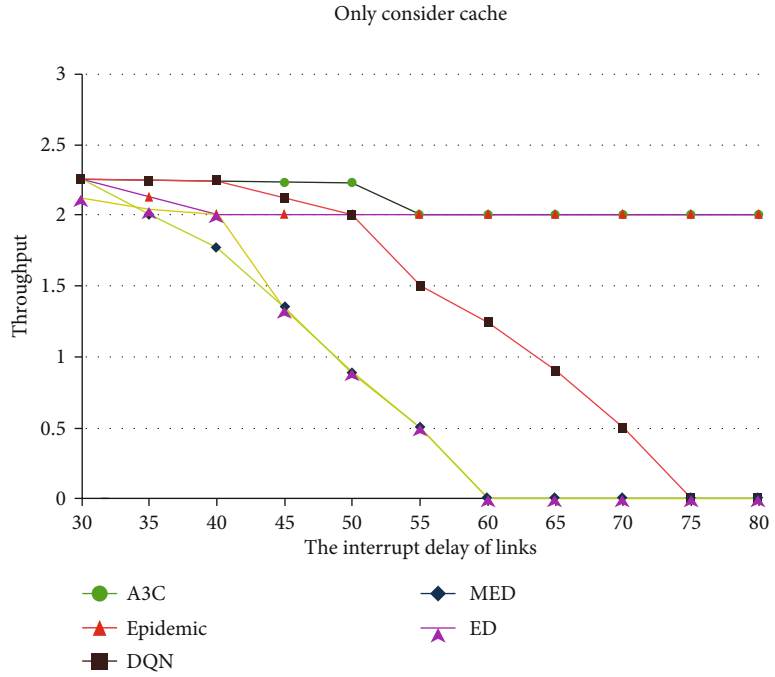

Figure 19: The throughput of service 1.

6.2. Simulation Results

6.2.1. Comparison of End-to-End Paths. The end-to-end paths of different algorithms can be seen in Figures 11–13; owing to the different operating mechanism of algorithms, they choose different paths when encountering the interrupted nodes. We can see that only the DQN algorithm and A3C algorithm can change the route when facing different scenarios, and they often choose the paths that have the minimum end-to-end paths. In the comprehensive scenario, the DQN algorithm and A3C algorithm have compared the end-to-end delay of the above scenarios and have the optimal path in the three scenarios.

6.2.2. Comparison of Delivery Rate. In this paper, we broadcast 2 services in this topology and record delivery rate based on different link break delay over different algorithms and assume the minimum delivery rate is 0.75. It is shown in Figure 14 that as the link break delay increases, the delivery rate is decrease over majority of algorithms, but in the only consider detour scenario and the comprehensive scenario of DQN, the delivery rate is remain unchanged and is the maximum, because in these scenarios, the source nodes choose the detour path and not affected by the interrupted links, so increasing the delivery rate. Through the result of all algorithms shown in Table 2, we found that the DQN algorithm has higher delivery rate in the majority of circumstances, which shows the delivery rate improvement of our proposed models and the DQN algorithm. But the A3C algorithm has the highest delivery rate in every scenario, because the A3C has many subthreads which can find the optimal paths in a very short time. The A3C algorithm and DQN algorithm can satisfy the constrained delivery rate in most cases.

6.2.3. Comparison of End-to-End Delay. In DTN network, due to the particularity of data connection, there may be open circuit between data connections, which makes data have to be stored in the node waiting for the link to be connected again. However, the storage space at the node is limited. When some greedy algorithms are adopted, multiple data storage may cause the use of nodes pace, so that when the following data arrives, it will cause data loss.

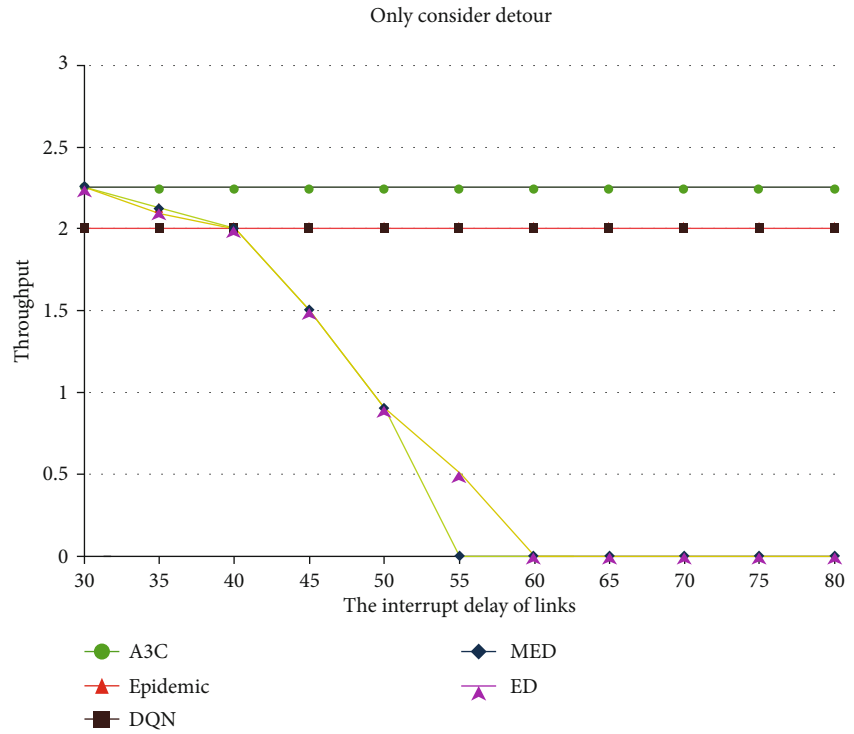

Figure 20: The throughput of service 1.

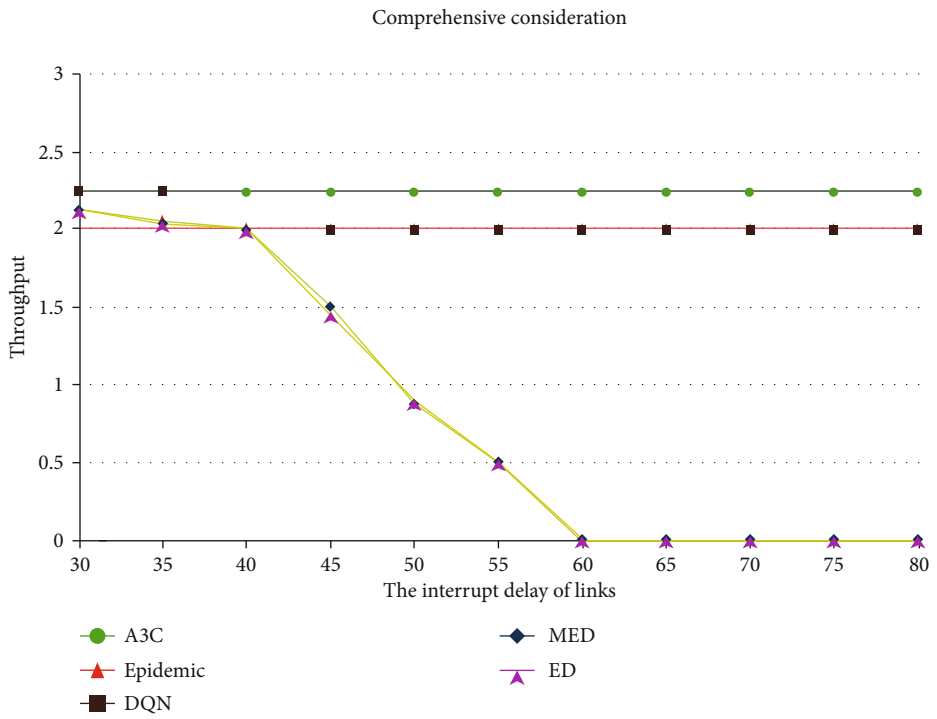

Figure 21: The throughput of service 1.

The waiting delay at the node is reflected in the total end-to-end delay of data transmission, including not only the transmission delay on the link but also the waiting delay at the node. When using epidemic, ED, and other algorithms, due to the particularity of the algorithm, it will copy multiple copies of data to be transmitted in the network, so compared with DQN and other reinforcement learning algorithms, it will increase the waiting delay at the node.

Because the transmission of multiple copies of data will cause congestion at the node and when the transmission continues, it will increase the queuing delay. For the transmission of data, intelligent algorithms such as DQN will

Table 4: The throughput of service in the scenario of comprehensive.

Throughput	Algorithms				
	Epidemic	ED	MED	DQN	A3C
Interrupted delay					
30	2.1	2.1	2.25	2.25	2.25
40	2	2	2.25	2.25	2.25
50	2	0.9	0.9	2	2.25
60	2	0	0	2	2.25
70	2	0	0	2	2.25
80	2	0	0	2	2.25

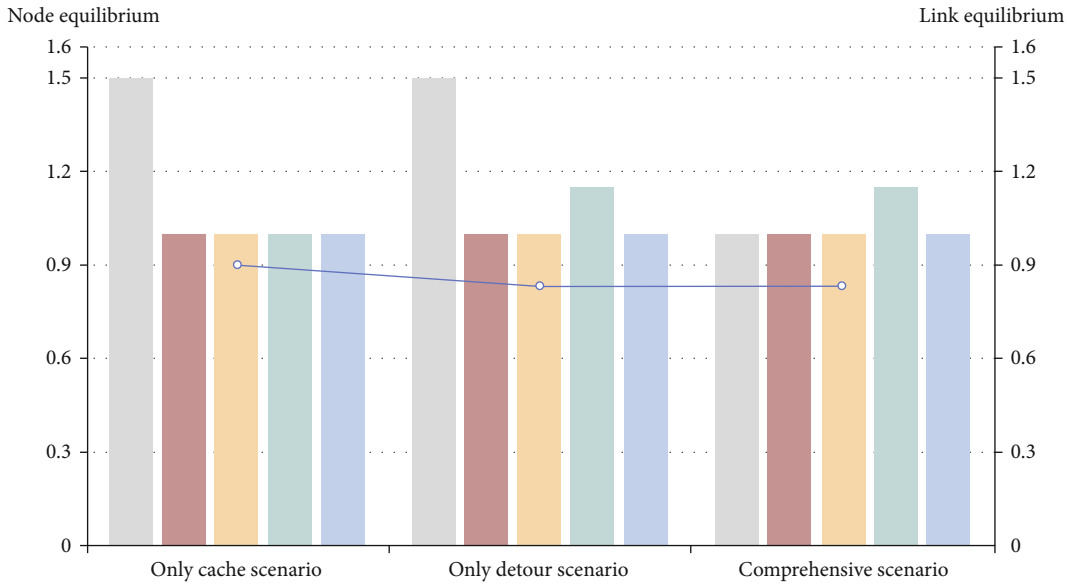

Figure 22: The node and link equilibrium of services.

not transmit multiple copies of the same data but take the reward in the algorithm as the guidance, minimize the end-to-end delay in the network, and reduce the occurrence of congestion at the node, so the end-to-end delay of each algorithm is shown in the figure below.

The total transmission delay of the service 1 and service 2 is shown in Figures 15–18; it can be seen that in the three scenarios, the minimum transmission delay is 100 ms, and the A3C algorithm has the minimum transmission delay, and DQN algorithm has lower delay, but in the scenario 1, the total transmission delay of service 1 and service 2 is increased as the interrupted delay of link is increasing, and the specific data are shown in Table 3, because in this scenario when the fragments encounter the interruption, they choose to store at the node, so the delay keeps on rising. Because in our paper, we think the capacity of node is available, so over the Epidemic algorithm, the fragments can arrive the destination node smoothly, and the total transmission delay of the Epidemic is not too high. But over the ED algorithm, it has the highest transmission delay; for the ED algorithm is the source routing algorithm, it determines the transmission path when the source nodes send the fragments, so when the fragments encounter the interruption, they will not change the path and result the high delay.

6.2.4. Comparison of Throughput. The throughput of service is shown in Figures 19–21; it can be seen that in the three scenarios, the A3C algorithm has the maximum throughput, which is better than the DQN algorithm; in the consider cache scenario, the throughput is decrease, for in this the scenario the transmission delay is a little higher and the nodes that reach the destination node are not a lot. The specific throughput data of all algorithms are shown in Table 4, and the throughput of ED algorithm and MED algorithm of service 1 and service 2 is also decrease, because the delay is increase as the interrupted delay of link increase. But in the only consider detour and comprehensive consider scenarios, there has the maximum throughput, which can be proved that our models and the adopted algorithm have improved the transmission.

6.2.5. Comparison of Node Equilibrium and Link Equilibrium. The node equilibrium of services can be seen from Figure 22; we can see the Epidemic algorithm has the maximum node equilibrium; for in the process of the algorithm, it forwards the fragments over the flooding manner, which means when the node comes into the communication scope of other nodes, if the node found that other nodes

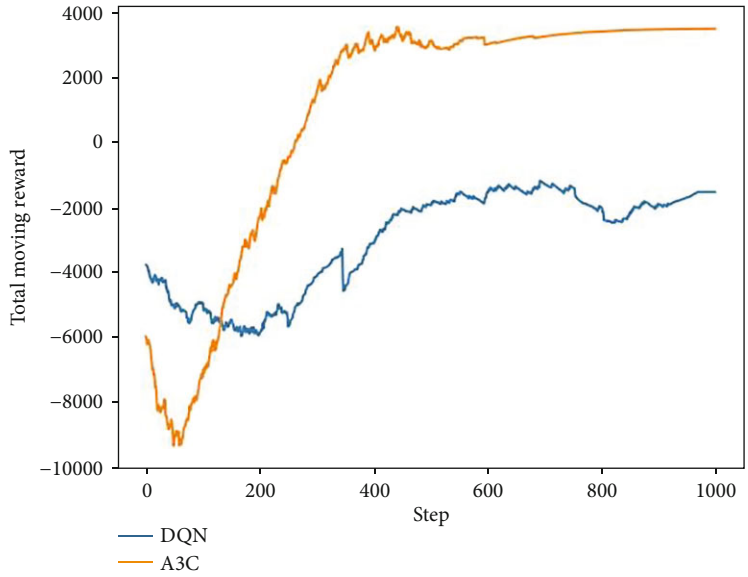

Figure 23: The node and link equilibrium of services.

have not the fragment, it will send the fragment to the other nodes. So it can lead to many copies of fragments in the work, so every node may store every fragment of every service, so the equilibrium is the highest. The node equilibrium of the ED algorithm and the MED algorithm is a little lower, and average node equilibrium of DQN is a little higher, which has to be improved. And for the link equilibrium, the Epidemic algorithm has the maximum link equilibrium; the cause of the result is the same to the node equilibrium, because it forward too many duplicates in the network. The average link equilibrium of DQN is a little higher but not very large as the node equilibrium, so it also has to be improved, so the A3C algorithm has improved the node and link equilibrium, which has lower equilibrium compared to the DQN algorithm.

6.2.6. Comparison of Total Reward and Loss. From Figure 23, we can see that the A3C algorithm has higher reward and can converge very quickly. At first, the value of reward is random jitter, because the exact value cannot be obtained in a short time. And the reward of A3C can get close to its top value with 400 episodes, but DQN needs about 700 episodes. The results prove that our models and algorithms adopted can reach an optimal value and converge.

7. Conclusions

In this paper, we have proposed optimal models based on different scenarios consist of the only consider cache scenario, the only consider detour scenario, and comprehensive consider scenario; the models are intend to jointly consider the behavior and the buffer of the nodes to improve the performance of the data transmission. Owing to different choices of the nodes, there will form three scenarios, and we adopted the DQN algorithm to solve the complex nonlinear optimization problem and to get the optimal solutions, which consist of lower end-to-end delay, higher throughput, and better data delivery guarantees. The results of simulation show that compared to other algorithms like Epidemic, ED algorithm, and MED algorithm, the DQN algorithm we adopted has better performance improvement.

As future work, we are going to improve the optimal models and decrease the overhead of nodes and links, expecting the application of DQN can be further studied in the delay tolerant networks.

Data Availability

The data is available from the following link: https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow/tree/master/contents/10_A3C.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgments

This research was funded open fund project of the Science and Technology on Communication Networks Laboratory (Grant No. SXX19641X073).

References

[1] L. Fang, Y. Li, X. Yun et al., "THP: a novel authentication scheme to prevent multiple attacks in SDN-based IoT network," *IEEE Internet of Things Journal.*, vol. 7, no. 7, pp. 5745–5759, 2020.

[2] W. Zhang, J. Xiong, L. Gui, B. Liu, M. Qiu, and Z. Shi, "On popular services pushing and distributed caching in converged overlay networks," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, Valencia, Spain, June 2018.

[3] B. Fajardo, K. Yasumoto, N. Shibata, W. Sun, and M. Ito, "DTN-based data aggregation for timely information

collection in disaster areas," in *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 333–340, Barcelona, Spain, October 2012.

[4] Y. Wu and Z. Li, "Queueing analysis for delay/disruption tolerant networks with random link interruptions," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 94–99, Chengdu, China, December 2016.

[5] Z. Yang, R. Wang, Q. Yu et al., "Analytical characterization of Licklider transmission protocol (LTP) in cislunar communications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 3, pp. 2019–2031, 2014.

[6] H. Wu, Y. Li, J. Jiao, B. Cao, and Q. Zhang, "LTP asynchronous accelerated retransmission strategy for deep space communications," in *2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pp. 99–104, Aachen, Germany, September 2016.

[7] R. Lent, "Analysis of the block delivery time of the Licklider transmission protocol," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 518–526, 2019.

[8] Y. Xu, J. Liu, Y. Shen, J. Liu, X. Jiang, and T. Taleb, "Incentive jamming-based secure routing in decentralized Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 3000–3013, 2021.

[9] L. Zhi, X. Zhou, and J. Zhao, "Vehicle routing for dynamic road network based on travel time reliability," *IEEE Access*, vol. 8, pp. 190596–190604, 2020.

[10] Z. Ding, L. Shen, H. Chen, F. Yan, and N. Ansari, "Energy-efficient relay-selection-based dynamic routing algorithm for IoT-oriented software-defined WSNs," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 9050–9065, 2020.

[11] N. Bezirgiannidis and V. Tsaoussidis, "Packet size and DTN transport service: evaluation on a DTN testbed," in *International Congress on Ultra Modern Telecommunications and Control Systems*, pp. 1198–1205, Moscow, Russia, October 2018.

[12] H. Lu, F. Jiang, J. Wu, and C. W. Chen, "Performance improvement in DTNs by packet size optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 4, pp. 2987–3000, 2015.

[13] 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.

[14] J. Segui, E. Jennings, and S. C. Burleigh, "Enhancing contact graph routing for delay tolerant space networking," in *Proceedings of the Global Communications Conference, GLOBECOM*, pp. 1–6, Houston, TX, USA, December 2011.

[15] N. Bezirgiannidis, C. Caini, D. D. P. Montenero, M. Ruggieri, and V. Tsaoussidis, "Contact graph routing enhancements for delay tolerant space communications," in *2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pp. 17–23, Berlin, Germany, December 2018.

[16] Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," in *Service Assurance with Partial and Intermittent Resources*, pp. 239–254, Springer, 2014.

[17] P. Maitreyi and M. S. Rao, "Design of binary spray and wait protocol for intermittently connected mobile networks," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–3, Las Vegas, NV, USA, January 2017.

[18] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: routing for vehicle-based disruption-tolerant networks," in *25TH IEEE International Conference on Computer Communications*, pp. 1–11, Barcelona, Spain, January 2016.

[19] R. Lent, "A cognitive network controller based on spiking neurons," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[20] R. Lent, D. Brooks, and G. Clark, "Validating the cognitive network controller on NASA's scan testbed," in *2020 IEEE International Conference on Communications (ICC)*, pp. 554–559, Dublin, Ireland, May 2020.