Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A multipath routing algorithm for satellite networks based on service demand and traffic awareness*

Ziyang XING[1,2], Hui QI[1,2], Xiaoqiang DI[†‡1,2,3], Jinyao LIU[1,2], Rui XU[1,2], Jing CHEN[1,2], Ligang CONG[1,2]

[1]*Jilin Key Laboratory of Network and Information Security, Changchun 130022, China*

[2]*School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China*

[3]*Information Center, Changchun University of Science and Technology, Changchun 130022, China*

[†]E-mail: dixiaoqiang@cust.edu.cn

Received Oct. 25, 2022; Revision accepted Mar. 27, 2023; Crosschecked Apr. 25, 2023

Abstract: With the reduction in manufacturing and launch costs of low Earth orbit satellites and the advantages of large coverage and high data transmission rates, satellites have become an important part of data transmission in air-ground networks. However, due to the factors such as geographical location and people's living habits, the differences in user' demand for multimedia data will result in unbalanced network traffic, which may lead to network congestion and affect data transmission. In addition, in traditional satellite network transmission, the convergence of network information acquisition is slow and global network information cannot be collected in a fine-grained manner, which is not conducive to calculating optimal routes. The service quality requirements cannot be satisfied when multiple service requests are made. Based on the above, in this paper artificial intelligence technology is applied to the satellite network, and a software-defined network is used to obtain the global network information, perceive network traffic, develop comprehensive decisions online through reinforcement learning, and update the optimal routing strategy in real time. Simulation results show that the proposed reinforcement learning algorithm has good convergence performance and strong generalizability. Compared with traditional routing, the throughput is 8% higher, and the proposed method has load balancing characteristics.

Key words: Software-defined network (SDN); Quick user datagram protocol Internet connection (QUIC); Reinforcement learning; Sketch; Multi-service demand; Satellite network

https://doi.org/10.1631/FITEE.2200507 **CLC number:** TP393

1 Introduction

Low Earth orbit (LEO) provides uninterrupted data transmission to users in various Earth environments through full coverage. It has the advantages of wide coverage, low cost, and flexible deployment, and can be widely used (Jia et al., 2020). The current representative LEO satellite networks are Motorola's iridium satellite mobile communication network, LQSS's Globalstar mobile communication satellite system, and Musk's satellite network, providing mainly SpaceX, OneWeb, etc. As shown in Fig. 1, ground and the iridium constellation can be used by ground and air users to communicate with other users without relying on the ground network, but the distribution of the Earth's population is not uniform. Most of the world population is distributed in easily habitable areas, and the population in harsh areas (oceans, deserts, and mountains) is small. Users in different geographical locations have

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. U21A20451), the Science and Technology Planning Project of Jilin Province, China (No. 20220101143JC), and the China University Industry-Academia-Research Innovation Fund (No. 2021FNA01003)

ORCID: Xiaoqiang DI, https://orcid.org/0000-0001-9432-4564

© Zhejiang University Press 2023

significant data-demand differences, which leads to unbalanced network traffic, network congestion, and even data transmission failure. In addition, the topology of LEO satellites changes dynamically, the constellation is large, and the overall perspective is difficult. It cannot obtain real-time traffic in a fine-grained manner (Zhang et al., 2020), which affects optimal routing decisions and ultimately transmission performance.

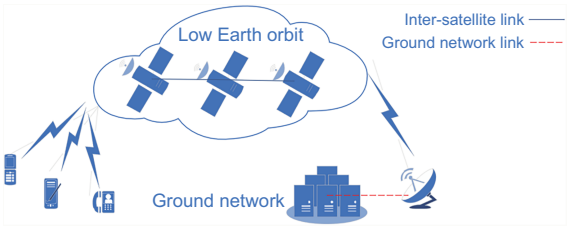

Fig. 1 Low Earth orbit satellite and multi-user communication architecture

In addition, the service demands of different users in the LEO satellite network are different. Most of the traditional routing algorithms are based on the minimum number of hops, as shown in Fig. 2, and do not distinguish the type of service demand. Based exclusively on source Internet protocol (IP) and destination IP, selecting one path causes the data streams of all service demands to be mixed together; that is, the paths of all service demands between the source IP and the destination IP are basically the same, which increases the burden of some paths and easily causes network congestion. The unselected path is always in an idle state, and the link utilization rate is low. In summary, it is necessary to obtain the network state in real time and dynamically generate the optimal route according to the real-time service demand and network state, to ensure the service quality of users with different needs.

Software-defined network (SDN) separates the control plane from the data plane, realizes function virtualization through programming, and can customize the functions to be implemented. The controller can manage the whole network in a unified manner, and the device hardware can perceive the global information. Using the multipath quick user datagram protocol Internet connection (MPQUIC) packet header extension to distinguish different services, deep reinforcement learning generates reward values through continuous interaction with the environment, makes dynamic decisions, and can achieve

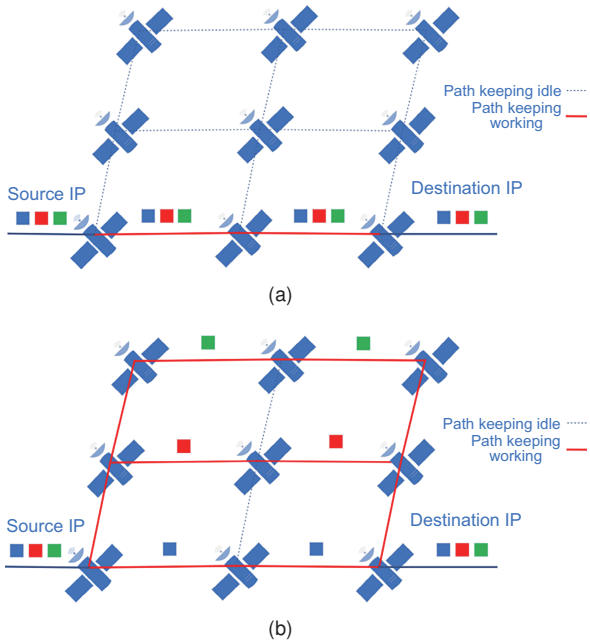

Fig. 2 Multi-service demand routing: (a) different service demands are mixed together; (b) multiple services are demand-oriented (IP: Internet protocol)

multi-service demand routing. In this study, SDNs and deep reinforcement learning are used to solve the above problems.

The main contributions of this paper are as follows:

1. A sketch-based network traffic analysis method based on an SDN has been proposed to realize fine-grained operations, such as classification of traffic and estimation of large or small flows.

2. We propose an artificial intelligence multi-service driven and traffic-aware routing scheme to convert different service demands and network states into reinforcement learning reward functions to achieve real-time dynamic routing.

3. To verify the multipath routing of the LEO satellite network based on service demand and traffic awareness, the iridium constellation is designed with a dynamic topology. After a large number of simulations, the throughput is improved by 8%.

2 Related works

The essence of multipath routing in LEO satellite networks is to select a suitable path from the source host to the destination host via the constellation, to allocate substreams according to the strategy, and to transmit data through multiple paths.

Therefore, the overall transmission performance can be improved only when the selected path cooperates with the transmission subflow. Based on these issues, in this paper we present the research from the following aspects:

The routing in the current satellite network does not support serving demand for multiple services at the same time. When there are multiple service requests, the transport layer protocol needs to be modified to meet the demands of multi-service characteristics, and the service demands cannot be adapted. The current routing research is divided mainly into two categories. One routing research category concerns allocating a reasonable number of subflows to avoid the subflow collisions that form bottleneck links. For example, Gao et al. (2019) proposed a method for low bandwidth utilization of multipath transmission control protocol (MPTCP) links. QSMPS, an SDN multipath routing optimization mechanism aimed at improving quality of service (QoS), adjusts the number of subflows according to service demands, which can significantly improve throughput. SDNs have been standardized by IETF (https://www.rfc-editor.org/info/rfc7426). Many scholars have studied software-defined satellite networks using software-defined abstract network hardware devices, which are no longer constrained by traditional TCP/IP network physical devices or business logic, and eliminate the hardware architecture for the entire network transmission. Through programming, SDN can realize the basic functions and customize the network according to the actual service demand. Liu ZG et al. (2020) proposed a software-defined information-centric satellite network (SDICSN) based on an SDN and an information-centric network (ICN), which aimed at the problems of low routing efficiency and complex satellite network control process. The network architecture, based on the periodicity and predictability of the satellite network, reduced the time complexity of the routing algorithm. Li et al. (2022) proposed the joint routing and task placement (JRTP) algorithm to optimize the number of transmission paths in an integrated air-ground network. Through the task topology graph model, the route and task placement were jointly determined, and the route problem was transformed into the classic shortest path problem. The second routing research category focuses on whether the allocation data on the selected path are overloaded and cause congestion. For example, Wang et al. (2021) applied SDN to obtain the global state of the network and used traffic prediction based on the autoregressive moving average model. The method classifies the data of the real-time distribution network nodes of the link, improves the throughput, realizes the adaptation of the bandwidth required by different services to the distribution path, and avoids network congestion. In summary, the current research on SDNs in satellite network routing still focuses on traditional performance optimization, cannot cope with multi-service demands, lacks intelligent routing, and cannot obtain the best route in real time when the load changes significantly (unbalanced resource distribution). Due to high dynamics, limited bandwidth, and slow convergence of LEO satellite networks, traditional routing optimization algorithms need to rely on specific network environments and states, and their generalization capabilities are weak.

The multipath transmission protocols include MPTCP and MPQUIC. MPQUIC can implement more refined data flow operations (Rabitsch et al., 2018). The QUIC (https://www.rfc-editor.org/info/rfc9000) advantages include 0-RTT, forward error correction, flexible congestion control, and connection migration. The unique connection ID in the connection mechanism is more suitable for satellite networks. Many scholars have studied the transmission performance of QUIC in 5G networks (Mogensen et al., 2019) and satellite networks (Yang WJ et al., 2021). Yang SY et al. (2018) analyzed the performance of QUIC in satellite networks and compared the performances of QUIC and TCP in LEO and geosynchronous Earth orbit (GEO). The experimental results showed that QUIC has better transmission performance than traditional TCP by virtue of the 0-RT handshake advantage. Arfeen and Uddin (2020) applied QUIC to the air-sea network and compared the page-loading speed for different browsers. The experimental results showed that in most cases, the throughput of QUIC is better than those of TCP and SPDY. Shi et al. (2021) designed a QUIC-based MPDTP system according to the characteristics of satellite networks. MPDTP determines whether to send redundant data packets according to the time when the acknowledgement (ACK) returns data packets. Kuhn et al. (2020) summarized the advantages of QUIC in satellite network

communication from the perspective of network operators. The new QUIC protocol can address many challenges in LEO satellite networks and has advantages in many aspects, such as periodic on-off of satellite links, bandwidth saving, multipath routing, and secure transmission (Bujari et al., 2020). Liu D et al. (2022) proposed a deep learning routing algorithm to predict a network topology to improve the QoS of satellite paths and to address the inability of traditional air-ground network routing schemes to meet the needs of heterogeneous services. Han et al. (2020), aiming at the high dynamics affecting the routing problem in satellite networks, proposed a routing algorithm of deep reinforcement learning to obtain a subset of available routes, with lower routing cost and better anti-jamming performance. In addition, some scholars have applied reinforcement learning to save energy. Liu JH et al. (2021) found that in a giant constellation, the incorrect use of satellite batteries in the routing phase may increase energy consumption and quickly lead to node failure. A new deep reinforcement learning based high-efficiency and energy-saving routing protocol, DRL-ER, avoids the battery energy imbalance of constellations and can extend the lifespan of giant constellations.

In summary, some research results have been achieved on multipath routing in the current LEO satellite network. However, due to the periodic on-off of the constellation topology and unbalanced traffic distribution, the controller cannot realize the fine-grained operation of the traffic, and the demand adaptability is also insufficient and cannot achieve multipath routing well, which affects the multipath transmission performance of MPQUIC. In this study, an SDN controller is used to obtain the state of the entire network, and a sketch is used to calculate the size of the data stream. The MPQUIC data packet header extension field distinguishes different services to deal with the above drawbacks.

3 System architecture

The system architecture is shown in Fig. 3, consisting of an SDN controller, some switches supporting OpenFlow, a client, and a server. The transport layer uses the MPQUIC protocol, the SDN controller manages the switches through the OpenFlow protocol, and all switches install count-min sketch (CMS).

The SDN controller collects the basic information of each switch node (network topology, remaining bandwidth, etc.), deploys a demand-driven and traffic-aware routing algorithm based on deep reinforcement learning, and uses CMS to obtain the MPQUIC flow size.

3.1 Setting multi-service demand

When the client sends a request to the server, it distinguishes specific services by setting different service demand flags (multi-service flags in Fig. 3). The multi-service flags are stored in the extension field of the QUIC packet header. To achieve service-oriented transmission, when the client initiates a request to the server, the user can set three service demands on the client—low latency, high bandwidth, and unlimited service demand—which are marked as D, B, O, respectively. When the user's client multi-service flag is set to D, it means that the current service request requires low-latency transmission; when the multi-service flag is set to B, it means that the current service request requires high-bandwidth transmission; when the multi-service demand flag is O or the flag is not set, it means that there is no demand for bandwidth or delay, and round-robin transmission is used.

The structure of the QUIC data packet header is shown in Fig. 4, where [Header Extensions] is a 16-bit extension used to store multipleServicesFlag. ConnectionID is the MPQUIC connection identifier, a globally unique 64-bit number generated by the client to identify the connection. SourceIP and destinationIP are the source IP and destination IP of this connection, respectively. FlowID is the number of data flows, used to identify different data streams in the same connection. PacketNumber is the data packet number, used to identify the number of data packets transmitted in a data stream.

The above MPQUIC packet header fields are in a public and unencrypted state, and can be parsed and extracted by the controller to be used as input parameters for online learning of routing decisions (https://datatracker.ietf.org/doc/draft-xing-alto-sdn-controller-aware-mptcp-mpquic/).

3.2 Network state collection based on sketch

Knowing the size of the network data flow is extremely important for multipath routing. The

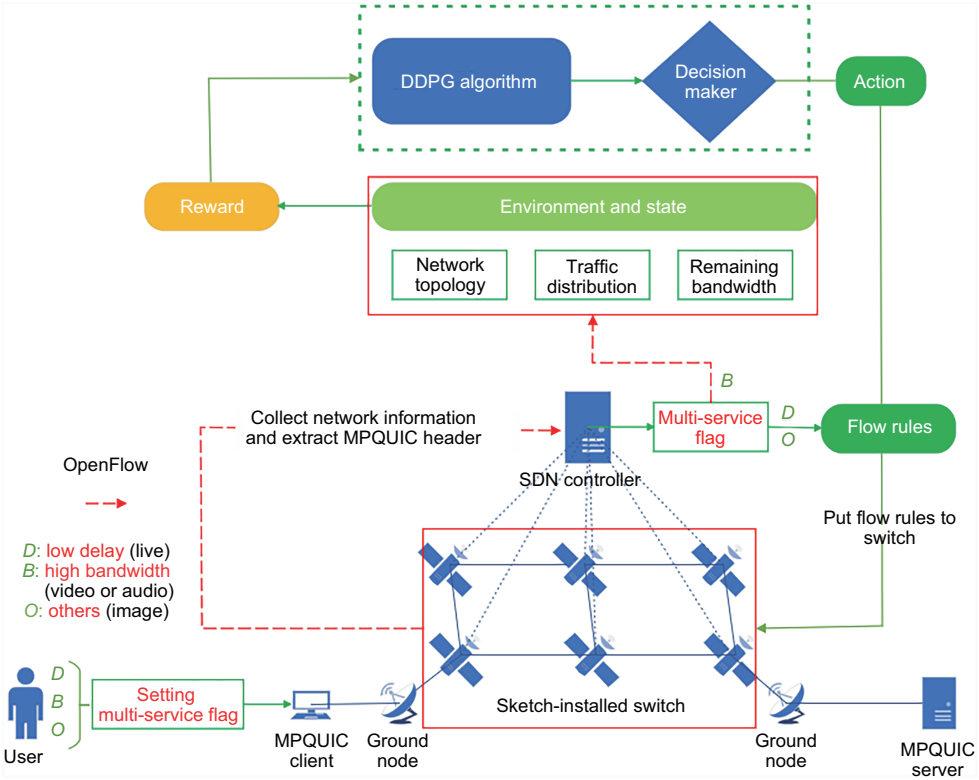

Fig. 3 System architecture (a client sends a request to the remote server, and the server will push data to a client with multiple paths by the request)

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	
S	Type			Next				Magic "uic"/"UIC"																							
ConnectionID (64)																															
Version																															
PacketNumber																															
[Header Extensions] (16) multipleServicesFlag																															...
Payload																															...

Fig. 4 Location of the multi-service flag in QUIC packets

current SDN controller link layer discovery protocol (LLDP) is not fine-grained enough to obtain network state information. The method of querying the number of packets through the REST API (https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343539/Floodlight+REST+API) to obtain network state information requires a large number of flow table entries (Huo et al., 2022), and it is impossible to obtain network state information in larger-scale satellite networks. In the process of multipath transmission, the path is selected according to the transmission path bandwidth, round-trip delay, packet loss rate, and attributes of the transmission data stream (elephant stream, mouse stream, etc.). A scientific routing algorithm should try to estimate the transmission performance of the path and the size of the data waiting to be transmitted to avoid network congestion or path load imbalance caused by the mutual influence of elephant flow and mouse flow. Commonly used data flow measurement methods include machine learning, sketch, and so on. Because the data flow needs to be sensed in real time, the memory needs to occupy less storage space, to have low complexity and high reading speed, and to guarantee a highly accurate data flow rate measurement method.

A sketch is a hash-type data structure (Tang et al., 2019) that is widely used in network traffic measurement. It has the advantages of requiring less memory space, quick deployment, and highly accurate measurement results. The principle of the sketch's measurement of data flow is shown in Fig. 5. The column d consists of w counters (Ya et al., 2021). When the data flow arrives, the data packet ID is added to the corresponding position of the counter in the sketch after the hash function $h(i)$ operation, from which the size of the data flow is obtained. Sketch is a method for accurately determining the frequency of an element in a set of data sets. It uses a

multi-dimensional array counter. When an element is added, the hash function is used to calculate its position in the counter and update the count. The disadvantage is that different elements will collide due to the same value being obtained from the hash calculation. When we want to query the occurrence frequency of an element, we need only to use the hash calculation to find its corresponding counter. Because of possible conflicts, there is a certain error in the sketch method of counting the occurrence frequency, and the result is higher than the true value. The sketch statistical method is accurate and fast, requires less space, and has been widely used in data flow statistics, especially in statistics with large data volumes. These search methods (such as HashMap, binary search tree, and binary sort tree) are limited by the memory size, so they cannot complete data flow statistics well in satellite networks.

Liu LT et al. (2021) designed an FO-Sketch with hierarchical storage to address the difficulty of global traffic statistics and classification in cloud environments. Murua and Reviriego (2020) applied CMS to detect elephant flow and mouse flow. Because the LEO satellite network also has mixed flows such as elephant flow and mouse flow, we design a flow measurement framework (Fig. 6) based on OpenSketch (Yu et al., 2013) and deploy CMS in all switches. Before the measurement, the controller sets up the switches as needed, and the switches implement the statistics of the data flow according to the algorithm and report the results to the controller.

The whole process of calculating the data flow

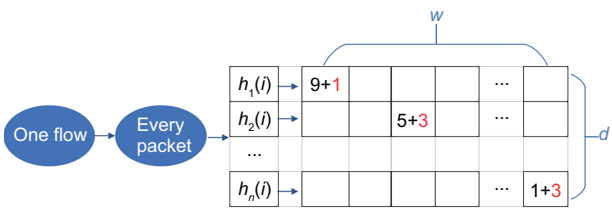

Fig. 5 Sketch data flow measurement principle

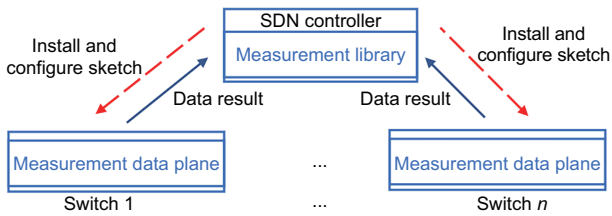

Fig. 6 Software-defined network (SDN) application sketch flow measurement framework

size in a sketch is shown in Fig. 7. It consists of flow table entries, CMS, flow quotient filter, and so on. Flow table entries store the basic structure of the data flow, CMS counts the number of data packets in a data stream, and the flow quotient filter stores the statistical results of CMS, wherein the CMS steps for implementing data packet statistics are as follows:

Suppose that the structure of CMS is a two-dimensional array CSC $[x, y]$. Its initial value is 0, x is set to the row value, and y is set to the column value. The statistical function counter is counter[1, 1], counter[1, 2], $\cdots$, counter$[x, y]$, and the hash(element) function is calculated for mapping, in which there are x functions, namely, hash(element)$_1$, hash(element)$_2$, $\cdots$, hash(element)$_x$. The construction method of the hash(element) function is as follows:

$$\text{int } i = \text{random}\,(0, x+y)\,, \tag{1}$$

$$\text{int } j = \text{random}\,(0, x+y)\,, \tag{2}$$

$$\begin{aligned}&\text{hash(element)}_{\text{row}}\\ =&((i\cdot\text{element}+j)\bmod p)\bmod x,\end{aligned} \tag{3}$$

where i and j are randomly generated integers in the range $[0, x+y]$, "element" is the element to be calculated, p is a hash constant, and "mod" is a modulo operation. It can be seen from the above that the result of hash(element)$_{\text{row}}$ is mapped in $[1, x]$.

When a new element(row$_{\text{time}}$, value$_{\text{time}}$) arrives at the row/value time, after calculation by hash(element)$_{\text{row}}$, the statistical function counter is updated to

$$\text{counter}(\text{element}, \text{hash(element)}_{\text{row}}) += \text{value}_{\text{time}}. \tag{4}$$

When the element is updated, the query method is as follows:

$$\begin{aligned}&\text{query(element)}\\ =&\min(\text{counter}(k, \text{hash(element)}_{\text{row}}))\,,\ k\in[1, x]\,.\end{aligned} \tag{5}$$

The streaming quotient filter, similar to the bloom filter, can quickly retrieve the number of elements in a large-scale database. The fewer the hash functions used, the shorter the query time. When a certain element needs to be counted, its quotient and remainder are calculated and stored in the specified slot. When other new remainders are encountered in the future, it is necessary to compare only the size with the value stored in the slot.

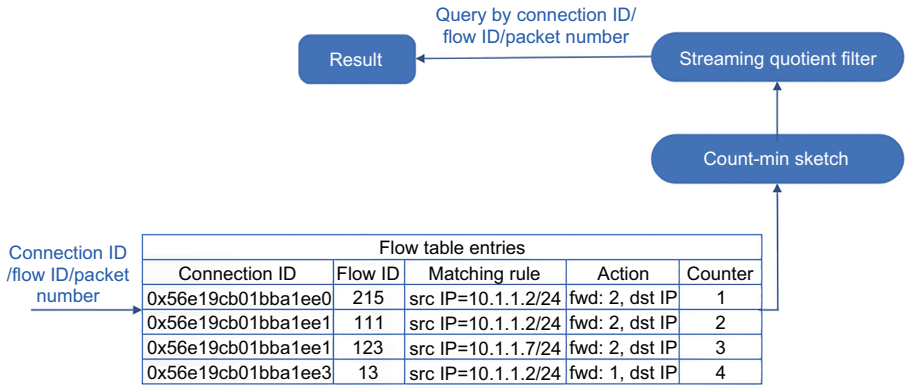

Fig. 7 Principle of calculating the MPQUIC data flow size in a sketch

When we want to query the size of the MPQUIC flow, we read only the results saved in the flow quotient filter according to the connectionID, flowID, and packetNumber.

To verify the performance of the sketch flow measurement in this study, the controller method of collecting the flow table through the REST API by the controller in the SDN is compared with the method of OpenSketch. As shown in Fig. 8a, the X axis is the number of switches, and the Y axis is the transmitted data. Because the controller REST API method of saving traffic needs to occupy many flow table entries, as the number of switches increases, the ability to count the number of data packets per unit time is reduced. In the sketch method, the performance of processing data packets is stable, and there is no performance degradation. When the number of switches reaches 300, the performance gap between these two methods is the largest. As shown in Fig. 8b, the X axis is the number of flows, which means the size of the data stream to be measured, and the Y axis is the runtime of the measurement scheme. The controller REST API method is limited by computational complexity and space, and runtime is much higher than that of the sketch method. The runtime of the sketch method is stable and is less affected by the size of the data flow.

In summary, the SDN sketch flow measurement in this study can realize tasks such as traffic statistics and classification in the LEO satellite network.

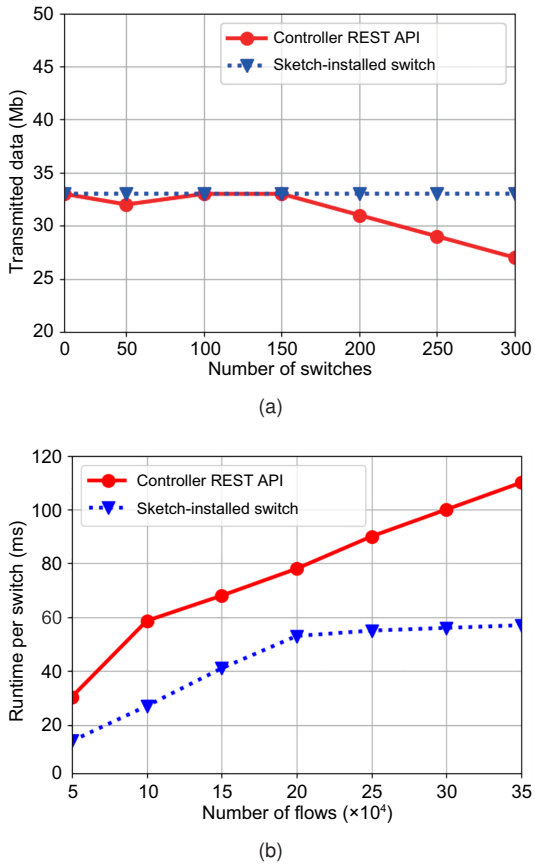

Fig. 8 Comparison of the performance of non-sketch and sketch methods in SDN: (a) transmitted data; (b) runtime

3.3 Online learning of routing decision

The routing decision online learning module is the core of the entire architecture. As shown in Fig. 9, routing decision online learning uses deep reinforcement learning to generate real-time optimal routing according to the network state information collected in the network state collection module. First, the agent extracts features through the convolution layer of the neural network according to the previously obtained state, and then maps them into the probability of a certain action by the fully

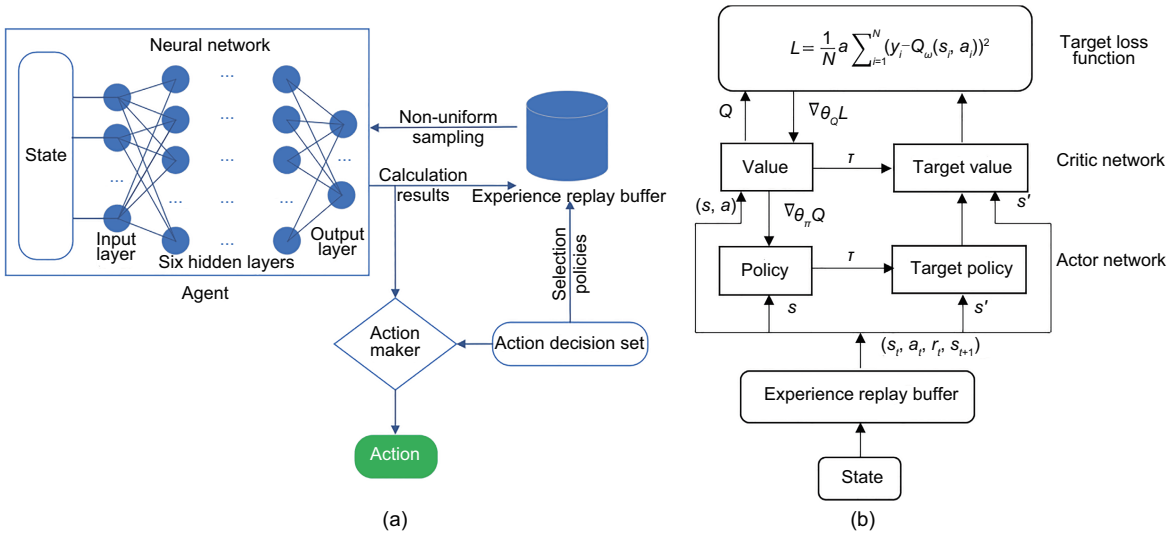

Fig. 9 Deep deterministic policy gradient procedure for routing decisions: (a) online learning of routing decisions; (b) actor network and critic network

connected layer. To reuse the previous important experience, an experience playback mechanism is introduced. This mechanism can avoid related sequences in the samples that affect the training. Finally, the transmission path is output according to Algorithm 1, and the flow rules are sent to the switch in the SDN.

The process of achieving the optimal routing is a continuous control problem. Different from the finite discrete actions, the continuous action space is a continuous set. The commonly used algorithm for dealing with discrete actions is deep Q-network (DQN), which can output n dimensions of n actions, but DQN is not suitable for dealing with continuous control spaces, in which DQN is solved with a policy network. The principle of the discrete problem is shown in Fig. 10.

As shown in Fig. 11, the deep deterministic policy gradient (DDPG) consists of a policy network (called an actor) and a value network (called a critic). The actor generates action a according to state $s, a = \pi(s;\theta)$. The value network scores the action a, denoted as $q(s,a;w)$. The value network has two inputs: one is the state s and the other is the action a. The output is for a certain action evaluation, and the better the action, the greater the output value. DDPGs consider the advantages of DQN self-learning.

State: Define state s composed of the following parts: connectionID, flowID, packetNumber,

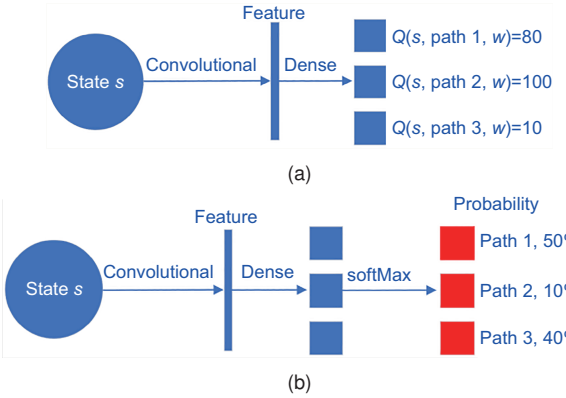

Fig. 10 Deep Q-network (DQN) and policy network to solve discrete control problems: (a) DQN for discrete action space; (b) policy network for discrete action space

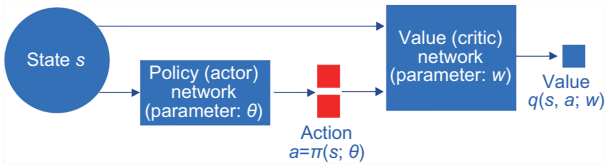

Fig. 11 Principle of the deep deterministic policy gradient

sourceIP, destinationIP, multipleServicesFlag, and remainingBandwidth:

$$s = (\text{connectonID}, \text{flowID}, \text{packetNumber}, \text{sourceIP}, \text{destinationIP}, \text{multipleServicesFlag}, \text{remainingBandwidth}). \tag{6}$$

1. connectonID is the connection number

Algorithm 1 DDPG online learning for routing

Input: multipleServicesFlag $R_x \in [D, B, O]$, topology G
Output: pathList

1: **if** $R_x == D$ **then**
2: return minimum-hop pathList // live
3: **else if** $R_x == B$ **then**
4: critic network: ω, $Q_\omega(s,a)$; actor network: $\theta, \mu_\theta(s)$
5: critic network (shadow): ω', $Q_{\omega'}(s,a)$; actor network (shadow): θ', $\mu_{\theta'}(s)$
6: initialize replay buffer, and randomly initialize the network parameters Q_{ω^-} and μ_{θ^-}
7: $\omega^- \leftarrow \omega$ and $\theta^- \leftarrow \theta$
8: **for** each episode $e \in \{1, 2, 3, \cdots\}$ **do**
9: obtain the initial STATE of the environment s_0
10: timeBegin = time()
11: **for** $t = 1 \rightarrow T$ **do**
12: choose an action a_t based on the current strategy
13: execute action a_t and obtain reward r_t. STATE of the environment changes to s_{t+1}
14: store (s_t, a_t, r_t, s_{t+1}) in experience replay buffer R
15: calculate each tuple using the target network: $y_i = r_i + \gamma Q_{\omega^-}(s_{i+1}, \mu_{\theta^-}(s_i'))$
16: minimize the target loss function and update the current critical network: $L = \frac{1}{N} a \sum_{i=1}^{N} (y_i - Q_\omega(s_i, a_i))^2$
17: calculate the sampled policy gradient to update the current actor network: $\nabla_\theta J \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta u_\theta(s_i) \nabla_a Q_\omega(s_i, a)|_{a=\mu_\theta a(s_i)}$
18: update target networks: $\omega^- \leftarrow \tau\omega + (1-\tau)\omega^-$ and $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$
19: create pathList
20: **end for**
21: **end for**
22: tx = time() − timeBegin
23: **if** tx < making decision deadline **then**
24: $R = \frac{\sum_{p=1}^{P}\left(\frac{b_p}{B_p}-1\right)}{P} - \alpha\frac{1}{\pi}\arctan\left(\frac{\sum_{p=1}^{P}(b_p-\bar{B})^2}{P}\right)$
25: return maximum(R)
26: **end if**
27: return pathList // video or audio
28: **else if** $R_x == O$ or length(R_x) < 1 **then**
29: return round-robin pathList // others
30: **end if**

obtained when the controller parses the MPQUIC packet header, and is the unique number of a connection.

2. flowID is the data flow number obtained when the controller parses the MPQUIC data packet header, and is the unique number of the data flow in a connection.

3. packetNumber is the data packet number obtained when the controller parses the MPQUIC data packet header, and is the unique number of the data packet in a data stream in a connection.

4. sourceIP and destinationIP are source IP and destination IP, respectively.

5. multipleServicesFlag is a service demand flag set by the user on the client side.

6. remainingBandwidth is the current remaining bandwidth, obtained by the controller by collecting the CMS on the switch and storing it in the streaming quotient filter.

Action: Define action a as a set of selected paths, namely, pathList $= \{p_1, p_2, \ldots, p_n\}$. The pathList is generated by the online learning module in the controller, and the flow rules are added to all switches in the network to achieve optimal routing.

Reward: Define the reward R as the feedback after an action is executed, and R_x is the value of the [Header Extensions] field of the MPQUIC packet header parsed by the controller, which is the value of the multi-service flag.

The meaning of each row in Algorithm 1 is as follows:

In lines 1–2, when the service demand is low delay D, a route is generated based on the shortest path.

In lines 3–27, when the service demand is high bandwidth B, the reward value needs to be calculated. In line 24, in $\frac{\sum_{p=1}^{P}\left(\frac{b_p}{B_p}-1\right)}{P}$, b_p is the remaining bandwidth of the pth path and P is the total bandwidth. $\alpha\frac{1}{\pi}\arctan\left(\frac{\sum_{p=1}^{P}(b_p-\bar{B})^2}{P}\right)$ is normalized with respect to the variance function, which can indicate whether the bandwidth of each path is balanced. $\bar{B}$ is the average bandwidth, and $R \in [-1, 0]$. When $R = 0$, this action is encouraged, which is conducive to obtaining good results; otherwise, the action is punished, and the optimal action is calculated after many iterations. α is a hyperparameter obtained from debugging.

In lines 28–30, when the service demand is O or the flag is not set (the flag length is zero), a round-robin route is generated.

There are four neural networks in routing decision deep reinforcement learning, namely:

1. Actor network, which is responsible for

updating the policy network and selecting an action a according to state s.

2. Critic network, which is responsible for updating the value network and calculating the current Q value.

3. Target actor network, which randomly samples one element from experience replay, chooses the next state s', and chooses its action a'.

4. Target critic network, which is responsible for calculating the target Q value.

Among them, the objective function is soft update; that is, the target Q network is slowly updated, and its formula is

$$\omega^{-} \leftarrow \tau\omega + (1-\tau)\,\omega^{-}, \tag{7}$$

where τ is a relatively small number, ω is a parameter of the critical network, and ω^{-} is the value of ω after updated.

Algorithm 1 is an offline strategy algorithm. The inputs are the current topology and network state information, and the output is the selected path set. The decision is finally made to perform the action that is conducive to load balancing and has the best performance.

4 Performance evaluation

The iridium constellation is set in the STK software (https://stksteakhouse.com/), the satellite orbit parameters are exported in Table 1, the network topology is configured in IPMininet (https://ipmininet.readthedocs.io/en/latest/) according to the parameters, and the MPQUIC client and server are interconnected through the iridium constellation. Parameters such as transmission delay and throughput are used to evaluate the performance of the algorithm.

Computer: Intel® Core™ i5 12400F CPU @2.50 GHz with six processors, 16 GB memory. Operating system: Ubuntu 22.04.

MPQUIC version: QUIC-go v0.22.0 (https://github.com/lucas-clemente/quic-go).

Controller: Floodlight v1.2 (https://github.com/floodlight/floodlight) is deployed in the GEO. The operation period is equal to the rotation period of the Earth, it is stationary relative to the ground station, and the satellite state information in LEO can be obtained.

Table 1 Iridium constellation parameters

Parameter	Value
Orbit	LEO
Altitude	780 km
Number of satellites	66
Link bandwidth	100 Mb/s
Inter-satellite link delay	40 ms

Analysis tool: Wireshark v3.6.3 (https://www.wireshark.org) is used for network protocol analysis, routing analysis, and packet analysis.

Network traffic simulation tool: Manimahi (http://mahimahi.mit.edu) is used to simulate network parameters in simulations and can record and playback operations.

The simulation ground stations are located in Jiamusi, China, in the Northern Hemisphere, and Mandurah, Australia, in the Southern Hemisphere.

4.1 Simulations

According to the operation law of the iridium constellation, the satellites will be switched every 1 min. When the satellites are running, there may be a phenomenon that one corresponds to multiple satellites and produces a continuous number of satellites. This work ignores the satellite that is too far away from the satellite and discusses only the one-to-one corresponding satellites. After the above analysis, the distance matrix $\boldsymbol{D'}$ is converted into a visible matrix $\boldsymbol{D}$.

$$\boldsymbol{D'} = \begin{bmatrix} 1 & D_{(1,2)} & D_{(1,3)} & \cdots & D_{(1,k)} \\ D_{(2,1)} & 1 & D_{(2,3)} & \cdots & D_{(2,k)} \\ D_{(3,1)} & D_{(3,2)} & 1 & \cdots & D_{(3,k)} \\ D_{(4,1)} & D_{(4,2)} & D_{(4,3)} & \cdots & \cdots \\ D_{(5,1)} & D_{(5,2)} & D_{(5,3)} & \cdots & 1 \end{bmatrix}. \tag{8}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & D_{(1,2)} & D_{(1,3)} & \cdots & D_{(1,k)} \\ D_{(2,1)} & 0 & D_{(2,3)} & \cdots & D_{(2,k)} \\ D_{(3,1)} & D_{(3,2)} & 0 & \cdots & D_{(3,k)} \\ D_{(4,1)} & D_{(4,2)} & D_{(4,3)} & \cdots & \cdots \\ D_{(5,1)} & D_{(5,2)} & D_{(5,3)} & \cdots & 0 \end{bmatrix}. \tag{9}$$

When there is a link between satellites, it is recorded as 1; otherwise, it is recorded as 0. $\boldsymbol{D}$ is the satellite visible matrix composed of 0 and 1 after being sorted by $\boldsymbol{D'}$. The pseudocode of the dynamic switching link of each satellite in the iridium constellation is shown in Algorithm 2.

Algorithm 2 Iridium constellation dynamic network

Input: matrix $\boldsymbol{D}\left(\boldsymbol{D}^1, \boldsymbol{D}^2, \cdots, \boldsymbol{D}^N\right), N = 66, \Delta t = 60$ s
Output: NULL

```
for each D^n != null do
    if D^n = D^{Δt+n} then
        do nothing
    else if D^n_{(j,k)} != D^{Δt+n}_{(j,k)} and D^{Δt+n}_{(j,k)} = 1 then
        put D^n_{(j,k)} and D^{Δt+n}_{(j,k)} up
    else if D^n_{(j,k)} != D^{Δt+n}_{(j,k)} and D^{Δt+n}_{(j,k)} = 0 then
        put D^n_{(j,k)} and D^{Δt+n}_{(j,k)} down
    end if
end for
```

In Algorithm 2, $D^n_{(j,k)}$ represents the value of the jth row and the kth column element in the nth visible matrix. When its value and the value of the element in the next cycle are 1, it means that the link in the state is connected; when the value is 0, it means that the link is disconnected. Δt is the satellite switching time interval, and the whole process simulates the periodic on-off of satellites.

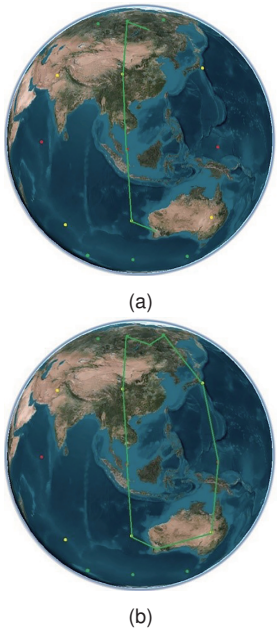

Fig. 12 Transmission path in the iridium constellation: (a) only one path is selected by the others' routing; (b) two paths are selected with the proposed method

4.2 Performance evaluation

Under the dynamic network with periodic connect-disconnect in Section 4.1, the network throughput, server CPU utilization, algorithm convergence, load-bandwidth utilization, and load balancing are compared among the scheme in this study, minimum-hop routing (Chen et al., 2022), round-robin routing, and other schemes. To decrease the influence of other factors on the test results, the first 100 results of the test are averaged, and then every 10 consecutive results are averaged into one value and included in the statistics.

Without the scheme in this study, the default path selected by the LEO satellite network is based on the minimum number of hops, as shown in Fig. 12a, and other paths are idle. After using the scheme in this study, multiple paths are allocated, as shown in Fig. 12b, which improves the link utilization and realizes parallel transmission of multiple paths.

4.2.1 Throughput

Throughput is the ratio of the actual amount of transmission to the unit time, and is an important indicator of network transmission performance. For example, in Eq. (10), the higher the throughput, the stronger the transmission capacity:

$$\text{throughput} = \frac{\text{number of data packets}}{\text{time 2} - \text{time 1}}. \tag{10}$$

The LEO satellite network is dynamic and time-varying, and the link will be disconnected and connected during the period. As shown in Fig. 13, the performance of the traditional routing algorithm is not good at 60, 120, and 180 s. The throughput decreases due to link switching, while the algorithm in this study has the ability to perceive traffic and is less affected by dynamic network link switching. The algorithm in this study uses traffic perception and obtains state information of the overall network to comprehensively select the optimal route, which can avoid the influence of network dynamics and organize throughput. The throughput of the algorithm in this study is obviously higher by 8% than that of the minimum-hop algorithm in the transmission process.

The path selected by the minimum-hop algorithm works well in a limited time. When the link is switched, the hop number needs to be recalculated, which will cause different subflows to converge, increase the burden of some paths, cause congestion, and reduce throughput (the red line in Fig. 13). Round-robin routing has the same drawbacks as

minimum-hop routing. Round-robin routing has no obvious advantages in dynamic networks and will also cause congestion and reduce throughput. Both minimum-hop routing and round-robin routing fluctuate due to link switching.

To further study the impact of the dynamic network formed by link switching on multipath routing, the HSR-CC algorithm (Xu and Ai, 2021) was compared in the simulations. This algorithm uses DQN to deal with the data transmission in the air-ground network. The transmission protocol is MPTCP. Fig. 13 shows that HSR-CC can deal with performance degradation of link switching in dynamic networks, but MPTCP is prone to header blocking and affects transmission, and the overall throughput is not as good as that of the proposed method.

For the impact of network traffic awareness on the entire transmission, in the simulations the throughput of multiple clients is discussed with traffic awareness enabled or disenabled, as shown in Fig. 14. When traffic perception is not used, the selection of the optimal path is affected by unbalanced traffic, resulting in low throughput. The HSR-CC algorithm cannot perceive network traffic, and the throughput decreases sharply with the increase in the number of clients. The traffic perception is relatively stable, the throughput does not drop sharply, and the traffic is allocated according to the idle traffic of the path, which can avoid congestion.

4.2.2 CPU utilization

The computing and storage resources in the satellite network are limited. Excessive consumption of computing resources may cause program suspension and affect network transmission. The MPQUIC server CPU deals mainly with encryption, user datagram protocol (UDP) data packet sending and receiving, and MPQUIC state maintaining (Langley et al., 2017). In the simulations, the CPU utilization within 100 s is analyzed, as shown in Fig. 15. The CPU utilization of the proposed method is relatively stable, and after 80 s, it tends to be stable and is suitable for use in LEO satellite networks.

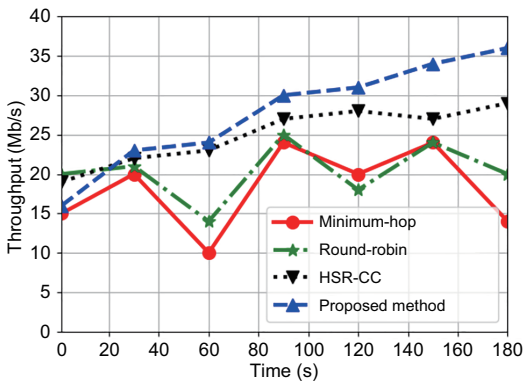

Fig. 13 Throughput of several schemes under a dynamic low Earth orbit network (References to color refer to the online version of this figure)

4.2.3 Convergence

To verify the convergence of the algorithm, the proposed scheme, DQN (Wu et al., 2021; Oroojlooyjadid et al., 2022), and the HSR-CC algorithm are compared in different training steps and normalized costs, as shown in Fig. 16. Fig. 16 shows that the proposed algorithm can accelerate the convergence and significantly reduce the training time. The proposed method is the best one in convergence.

4.2.4 Load-bandwidth utilization

Load-bandwidth utilization is an important indicator for measuring whether the

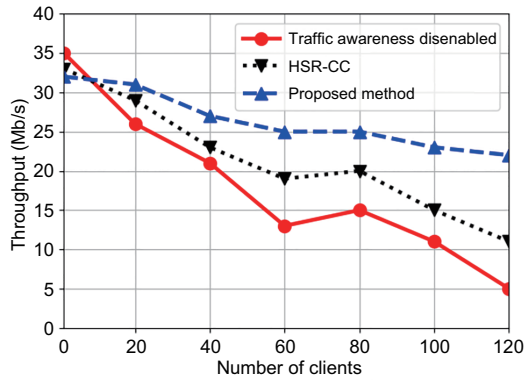

Fig. 14 Throughput of several schemes under a dynamic network

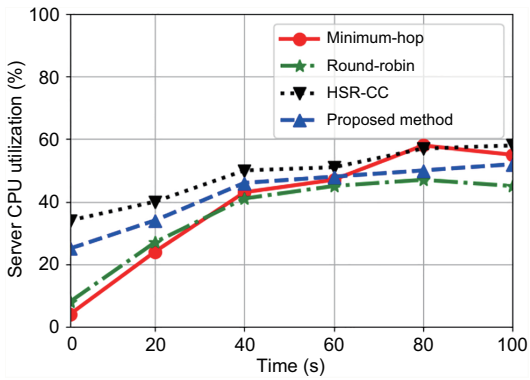

Fig. 15 CPU utilization

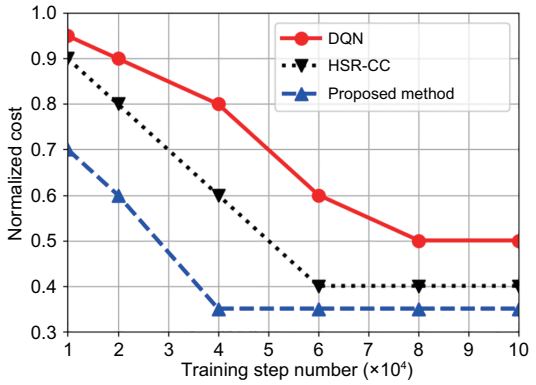

Fig. 16 Convergence process of different methods

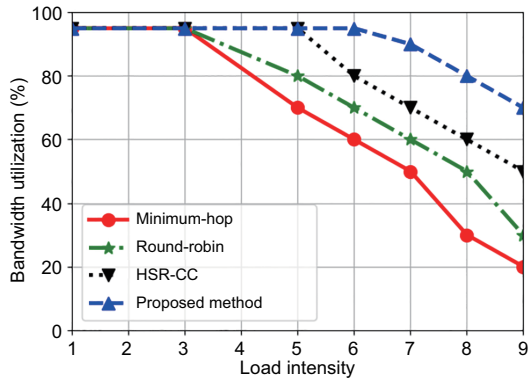

Fig. 17 Load-bandwidth utilization comparison

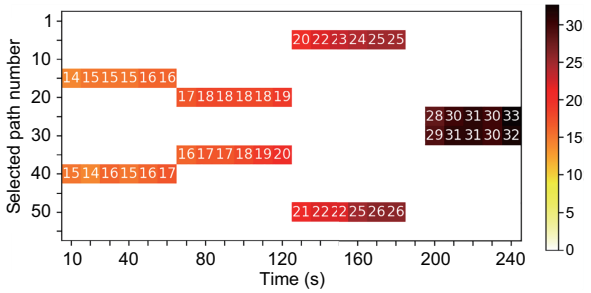

Fig. 18 Traffic distribution of inter-satellite links (References to color refer to the online version of this figure)

network is congested during transmission (https://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/8141-calculate-bandwidth-snmp.html). It is the ratio of the sending bandwidth to actual bandwidth. The higher the load-bandwidth utilization, the lighter the network congestion. The calculation method is as follows:

$$\eta = \frac{\sum \frac{\text{Bandwidth}_\text{s}}{\text{Bandwidth}_\text{o}}}{\text{TotalByte}}, \tag{11}$$

where TotalByte is the size of the transmission data stream, $\text{Bandwidth}_\text{s}$ is the stream sending bandwidth, and $\text{Bandwidth}_\text{o}$ is the actual bandwidth occupied. As shown in Fig. 17, the X axis represents load intensity. The larger the value, the higher the load. The Y axis is the load-bandwidth utilization value. When the network load is not high at first, the load-bandwidth utilization of several schemes is 98%. When the load increases, the load-bandwidth utilization begins to decrease. The route with the minimum-hop number decreases the fastest, indicating that the network is congested. The proposed method supports traffic awareness and can dynamically generate the optimal route. By comparing the start time of network congestion, the proposed method has the highest bandwidth utilization.

4.2.5 Load balancing

In the multipath transmission of the LEO satellite network, the links are complex, and the goal of load balancing is to make the traffic of the selected links as equal as possible to avoid congestion caused by overloading some paths. As shown in Fig. 18, the X axis is time, the Y axis is the path number selected at a certain time, and the colors in the heatmap are from light to dark, indicating that the flow value in the path is from small to large. The path traffic increases steadily without major fluctuations, indicating that the proposed method has stable transmission. The maximum traffic difference between these two paths selected by the algorithm within 240 s is approximately 1 Mb, indicating that the proposed method has achieved load balancing.

5 Conclusions and future work

This paper uses an SDN controller to obtain the low Earth orbit satellite network state information, the MPQUIC packet header extension field to distinguish different service demands, and CMS to measure the size of data streams (live, video, or others), and proposes a deep reinforcement learning multipath routing algorithm to dynamically generate real-time routing decisions. It can solve the problem of network congestion caused by unbalanced network traffic due to the significant differences in data demands of low Earth orbit satellites. Through the STK construction of low Earth orbit satellite network simulations, it is known that this algorithm has high convergence speed and high throughput (the throughput is 8% higher), and

achieves load balancing. In future work, we will continue to study the routing characteristics of SDN-based information-centric networks in low Earth orbit satellite networks, and to study other efficient algorithms for deep reinforcement learning to solve routing problems.

Contributors

Ziyang XING and Xiaoqiang DI designed the research. Hui QI processed the data. Ziyang XING drafted the paper. Jinyao LIU and Rui XU helped organize the paper. Jing CHEN and Ligang CONG revised the paper. Ziyang XING and Xiaoqiang DI finalized the paper.

Compliance with ethics guidelines

Ziyang XING, Hui QI, Xiaoqiang DI, Jinyao LIU, Rui XU, Jing CHEN, and Ligang CONG declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Arfeen A, Uddin R, 2020. Quality of experience-based optimization of satellite Internet-at-sea using WAN accelerators. *Int J Satell Commun Netw*, 38(6):527-556. https://doi.org/10.1002/sat.1366

Bujari A, Luglio M, Palazzi CE, et al., 2020. A virtual PEP for web optimization over a satellite-terrestrial backhaul. *IEEE Commun Mag*, 58(10):42-48. https://doi.org/10.1109/MCOM.001.2000322

Chen Q, Yang L, Guo DK, et al., 2022. LEO satellite networks: when do all shortest distance paths belong to minimum hop path set. *IEEE Trans Aerosp Electron Syst*, 58(4):3730-3734. https://doi.org/10.1109/TAES.2022.3143090

Gao K, Xu CQ, Qin JR, et al., 2019. QoS-driven path selection for MPTCP: a scalable SDN-assisted approach. Proc IEEE Wireless Communications and Networking Conf, p.1-6. https://doi.org/10.1109/WCNC.2019.8885585

Han C, Huo LY, Tong XH, et al., 2020. Spatial anti-jamming scheme for Internet of satellites based on the deep reinforcement learning and Stackelberg game. *IEEE Trans Veh Technol*, 69(5):5331-5342. https://doi.org/10.1109/TVT.2020.2982672

Huo LW, Jiang DD, Zhu XN, et al., 2022. A SDN-based fine-grained measurement and modeling approach to vehicular communication network traffic. *Int J Commun Syst*, 35(12):e4092. https://doi.org/10.1002/dac.4092

Jia ZY, Sheng M, Li JD, et al., 2020. LEO-satellite-assisted UAV: joint trajectory and data collection for Internet of Remote Things in 6G aerial access networks. *IEEE Int Things J*, 8(12):9814-9826. https://doi.org/10.1109/JIOT.2020.3021255

Kuhn N, Michel F, Thomas L, et al., 2020. QUIC: opportunities and threats in SATCOM. Proc 10th Advanced Satellite Multimedia Systems Conf and the 16th Signal Processing for Space Communications Workshop, p.1-7.

Langley A, Riddoch A, Wilk A, et al., 2017. The QUIC transport protocol: design and Internet-scale deployment. Proc Conf of the ACM Special Interest Group on Data Communication, p.183-196. https://doi.org/10.1145/3098822.3098842

Li X, Tang FL, Zhu YM, et al., 2022. Processing-while-transmitting: cost-minimized transmission in SDN-based STINs. *IEEE/ACM Trans Netw*, 30(1):243-256. https://doi.org/10.1109/TNET.2021.3107413

Liu D, Zhang JK, Cui JJ, et al., 2022. Deep learning aided routing for space-air-ground integrated networks relying on real satellite, flight, and shipping data. *IEEE Wirel Commun*, 29(2):177-184. https://doi.org/10.1109/MWC.003.2100393

Liu JH, Zhao BK, Xin Q, et al., 2021. DRL-ER: an intelligent energy-aware routing protocol with guaranteed delay bounds in satellite mega-constellations. *IEEE Trans Netw Sci Eng*, 8(4):2872-2884. https://doi.org/10.1109/TNSE.2020.3039499

Liu LT, Shen YL, Zeng SG, et al., 2021. FO-Sketch: a fast oblivious sketch for secure network measurement service in the cloud. *Electronics*, 10(16):2020. https://doi.org/10.3390/electronics10162020

Liu ZG, Zhu J, Zhang JM, et al., 2020. Routing algorithm design of satellite network architecture based on SDN and ICN. *Int J Satell Commun Netw*, 38(1):1-15. https://doi.org/10.1002/sat.1304

Mogensen RS, Markmoller C, Madsen TK, et al., 2019. Selective redundant MP-QUIC for 5G mission critical wireless applications. Proc IEEE 89th Vehicular Technology Conf, p.1-5. https://doi.org/10.1109/VTCSpring.2019.8746482

Murua J, Reviriego P, 2020. Faking elephant flows on the count min sketch. *IEEE Netw Lett*, 2(4):199-202. https://doi.org/10.1109/LNET.2020.3035272

Oroojlooyjadid A, Nazari M, Snyder LV, et al., 2022. A deep Q-network for the beer game: deep reinforcement learning for inventory optimization. *Manuf Ser Oper Manag*, 24(1):285-304. https://doi.org/10.1287/msom.2020.0939

Rabitsch A, Hurtig P, Brunstrom A, 2018. A stream-aware multipath QUIC scheduler for heterogeneous paths. Proc Workshop on the Evolution, Performance, and Interoperability of QUIC, p.29-35. https://doi.org/10.1145/3284850.3284855

Shi H, Zhang L, Zuo XT, et al., 2021. Multipath deadline-aware transport proxy for space network. *IEEE Int Comput*, 25(6):51-57. https://doi.org/10.1109/MIC.2021.3112804

Tang L, Huang Q, Lee PPC, 2019. MV-Sketch: a fast and compact invertible sketch for heavy flow detection in network data streams. Proc IEEE INFOCOM Conf on Computer Communications, p.2026-2034. https://doi.org/10.1109/INFOCOM.2019.8737499

Wang F, Jiang DD, Qi S, et al., 2021. An Adaboost based link planning scheme in space-air-ground integrated networks. *Mob Netw Appl*, 26(2):669-680. https://doi.org/10.1007/s11036-019-01422-4

Wu Q, Chen X, Zhou Z, et al., 2021. Deep reinforcement learning with spatio-temporal traffic forecasting for data-driven base station sleep control. *IEEE/ACM Trans Netw*, 29(2):935-948. https://doi.org/10.1109/TNET.2021.3053771

Xu JP, Ai B, 2021. Deep reinforcement learning for handover-aware MPTCP congestion control in space-ground integrated network of railways. *IEEE Wirel Commun*, 28(6):200-207. https://doi.org/10.1109/MWC.001.2100116

Ya D, Bin Q, Wei N, 2021. DW-Sketch: a sketch-based scheme for realizing multi-network measurement tasks. Proc 2nd Int Conf on Computer Communication and Network Security, p.191-195. https://doi.org/10.1109/CCNS53852.2021.00043

Yang SY, Li HW, Wu Q, 2018. Performance analysis of QUIC protocol in integrated satellites and terrestrial networks. Proc 14th Int Wireless Communications & Mobile Computing Conf, p.1425-1430. https://doi.org/10.1109/IWCMC.2018.8450388

Yang WJ, Shu SJ, Cai L, et al., 2021. MM-QUIC: mobility-aware multipath QUIC for satellite networks. Proc 17th Int Conf on Mobility, Sensing and Networking, p.608-615. https://doi.org/10.1109/MSN53354.2021.00093

Yu ML, Jose L, Miao R, 2013. Software defined traffic measurement with OpenSketch. Proc 10th USENIX Conf on Networked Systems Design and Implementation, p.29-42.

Zhang R, Liu J, Yang D, et al., 2020. A survey on satellite networks based on software-defined networking. *Front Data Comput*, 2(3):3-17.