



Routing Algorithm with Virtual Topology Toward to Huge Numbers of LEO Mobile Satellite Network Based on SDN

Min Jia^{1,2} · Siyu Zhu¹ · Linfang Wang¹ · Qing Guo¹ · Haitao Wang² · Zhihui Liu²

Published online: 13 October 2017
© Springer Science+Business Media, LLC 2017

Abstract SDN network is a dynamic, controllable, cost-effective and adaptable system. It is suitable for communication networks with high bandwidth and high dynamic characteristics. Therefore, combining SDN ideas with the new generation of LEO satellite networks can achieve more flexible monitoring and management of the network, and can make the network expansion more convenient. Joint the Depth-First-Search (DFS) idea and Dijkstra algorithm for the huge numbers of LEO mobile satellite network based on SDN is proposed to improve the computational efficiency and the reliability of calculation result. Moreover, the communication performance of space-based network based on SDN and traditional space-based network is compared and analyzed. The simulation results show that the huge numbers of LEO mobile satellite network based on SDN

breaks through the performance limitations of the traditional network architecture, and it can achieve better performance of the network.

Keywords LEO satellite network · Virtual topology · Routing algorithm · SDN · Huge numbers of LEO satellites

1 Introduction

With the development of Space-Ground integrated network, base station in the traditional ground mobile cellular network cannot be deployed to the particularly complex geographical environment, so the research of low earth orbit (LEO) satellite network technology has become a hot topic [1]. In order to realize the real-time access of small power mobile terminals and the seamless global coverage of satellite signals, there are more and more satellite nodes in the LEO satellite network being deployed, so the study of LEO satellite network characteristics and related routing algorithms has become an urgent issue to be solved. With the increase of the number of satellites, the topological structure of the satellite network changes faster and faster, and the routing calculation becomes more difficult. We use routing algorithm based on the idea of virtual topology [2, 3], which is widely used for routing calculation in the LEO satellite network. When routing the static topology within each time slice, we introduce the Depth-First-Search (DFS) algorithm [4, 5] and Dijkstra algorithm [6], and combine them together to improve the computational efficiency and the reliability of the calculation results.

Because of the large number of nodes in the network and the complexity of the network deployment, it is difficult to control the distribution of data traffic in the network, so as to the management of the network. The new network

✉ Min Jia
jiamin@hit.edu.cn

Siyu Zhu
527853664@qq.com

Linfang Wang
56304784@qq.com

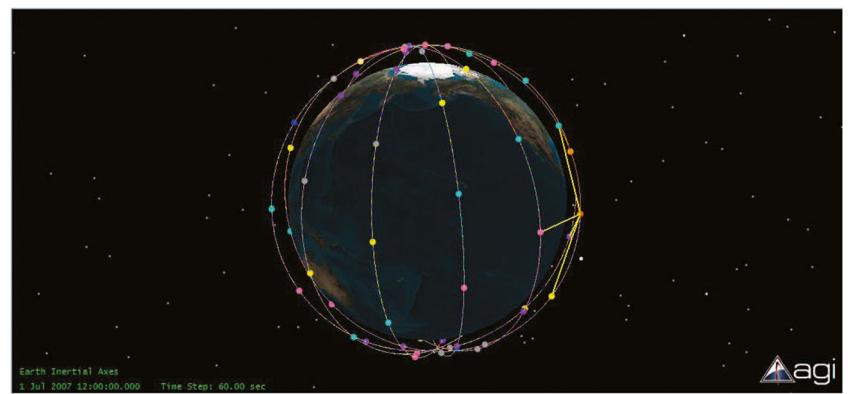
Qing Guo
qguo@hit.edu.cn

Haitao Wang
wanght@sgiiit.cn

Zhihui Liu
liuzhh@sgiiit.cn

¹ Communication Research Center, School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China

² State Key Laboratory of Space-Ground Integrated Information Technology, Beijing, China

Fig. 1 Iridium system 3D view

architecture of the software defined network (SDN) [7, 8] separates data layer and control layer. The control layer is monitoring the whole network, a comprehensive analysis of the real-time information on the network traffic distribution and the network node load is underway, according to the information, the optimal path is calculated. And it controls the equipment in the network to transmit data and provides real-time management and maintenance to the network to improve the operating efficiency and the stability of the network. So combining SDN ideas with the new generation of LEO satellite networks [9–13] can effectively improve the network performance.

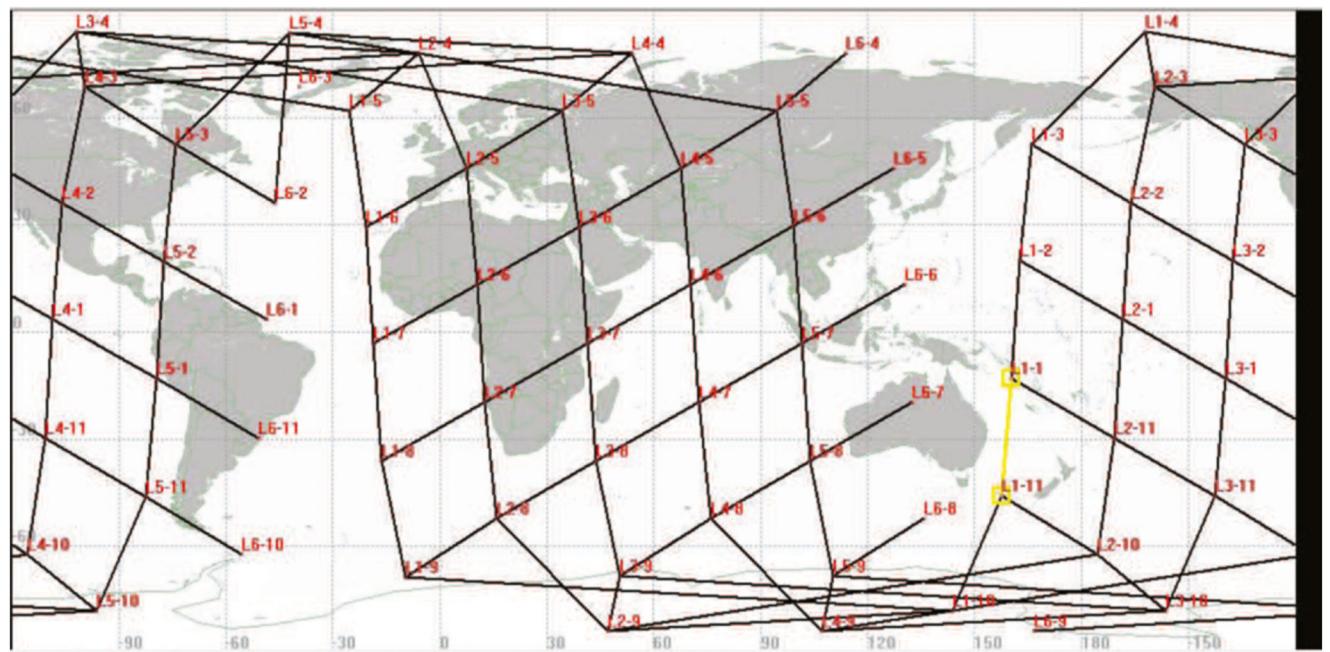
The rest of this paper is arranged as follows. Section 2 introduces the system model used in this paper. Section 3 introduces the routing algorithms for LEO network. The analysis and the simulation results will be described and shown in Section 4. And Section 5 gives a conclusion of this paper.

2 System model

The iridium system is a typical LEO satellite system and built by Motorola. It consists 66 LEO satellites, the weight of each satellite is about 689 kg, the 66 satellites are equally distributed on 6 orbits, each orbit having 11 satellites with a track height of about 780km and the angle between adjacent orbital planes is 30 degrees, the inclination of the orbit is 86.4 degrees. The satellite running speed is 27000km/h, its operating cycle is about 1.66h, which is 100min.

Based on the orbit parameter, we build the iridium system in the STK environment. Figure 1 shows the 3D view of Iridium system.

Figure 2 shows the 2D view of Iridium system. We can see from it that when spreading the topological structure of iridium system into 2D view, the topological structure between 4 adjacent stars is parallelogram, and around

**Fig. 2** Iridium system 2D view

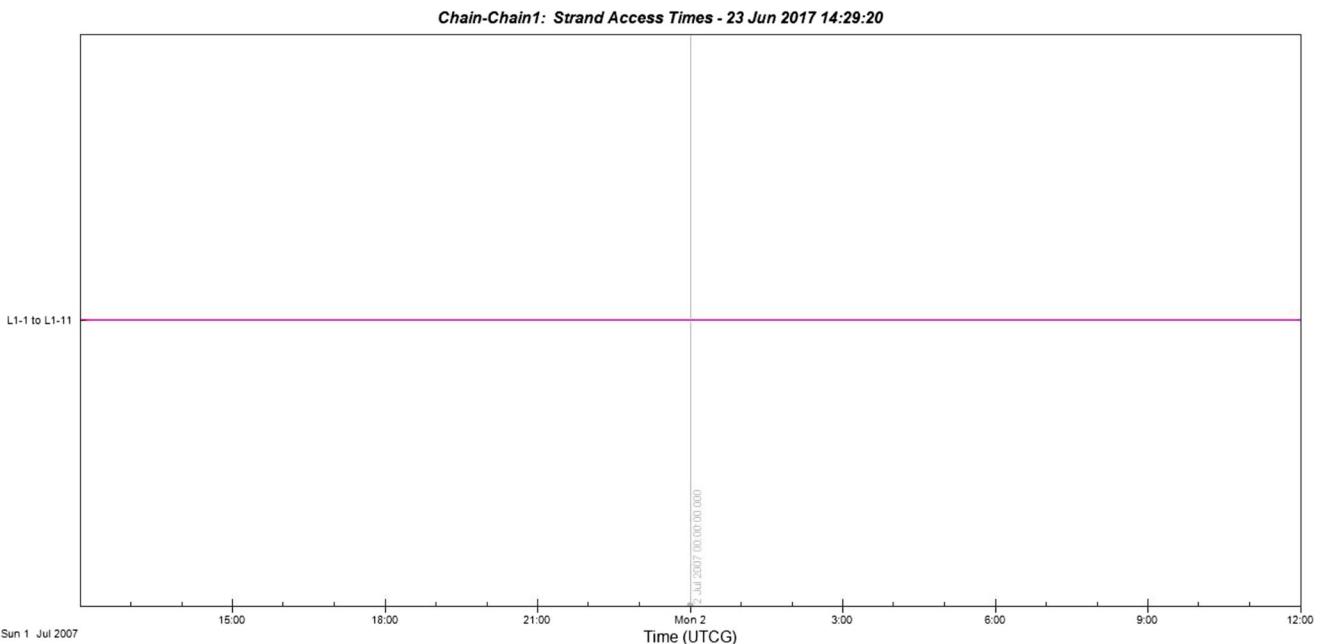


Fig. 3 Link between L1-1 and L1-11 satellites

each satellite there are 4 communication links, and the 4 communication links are formed by linking two adjacent satellites in the same orbit and two satellites in the adjacent orbits.

In order to study the satellite routing algorithm, we need to understand the inter-satellite link, because the changes of link states will lead to changes in the network topology.

We set up the inter satellite link to observe its link status in iridium communication system built in the STK

environment. In this paper, 4 links are set up to be observed. Among these 4 links, two are formed between two adjacent satellites in the same orbit and the other two are formed between two adjacent orbits.

Figures 3 and 4 show the inter-satellite link between two adjacent satellites in orbit 1. It can be seen that the inter satellite links between two adjacent satellites on the same orbit are always connected, so they can communicate with each other at any time.

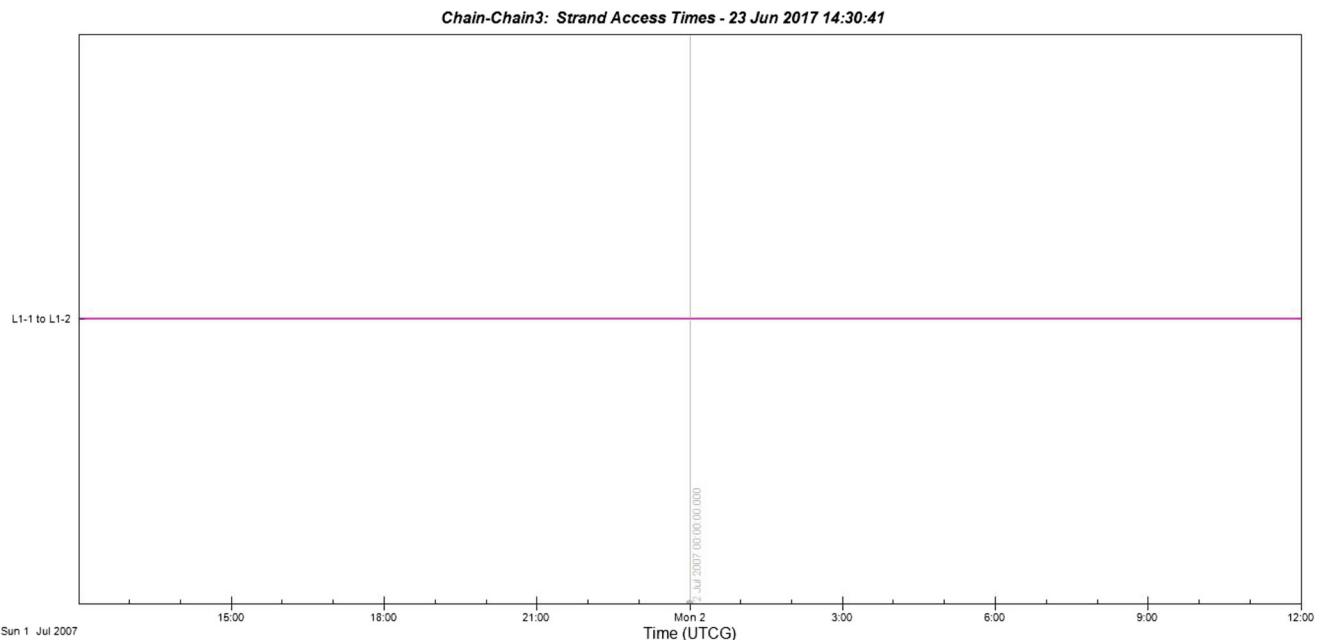


Fig. 4 Link between L1-1 and L1-2 satellites

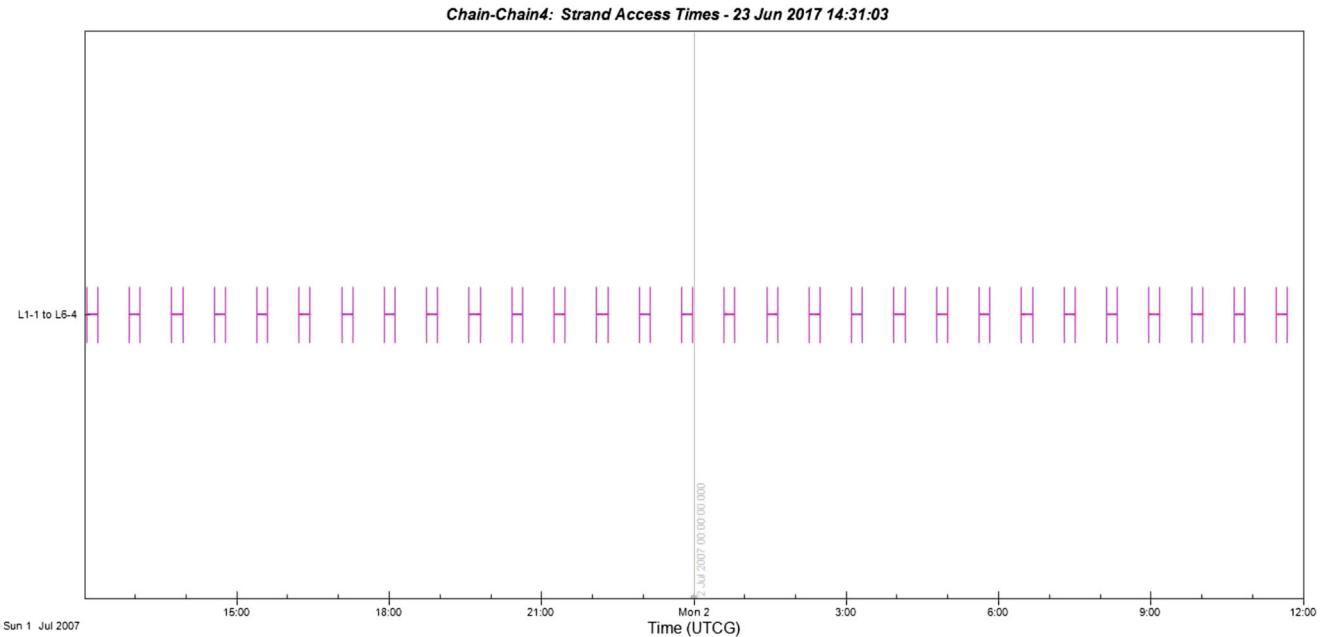


Fig. 5 Link between L1-1 and L6-4 satellites

Figures 5 and 6 show the inter-satellite link between two satellites in the adjacent orbits. It can be seen that the inter-satellite link between two satellites in the adjacent orbits is only partially connected and they can communicate only when connected.

To study the specific link duration, we use the link operation report between the first satellite in orbit 1 and the fourth satellite in orbit 6.

Figure 7 shows the specific link duration between L1-1 and L6-4. It can be seen that the shortest link duration between adjacent orbits is 777.189 seconds, the longest is 777.197 seconds, and the average connection time is 777.192 seconds, about 13 minutes.

Thus, source and destination are known when designing routing algorithms at the point of data communication. The time required to compute a required reliable path should

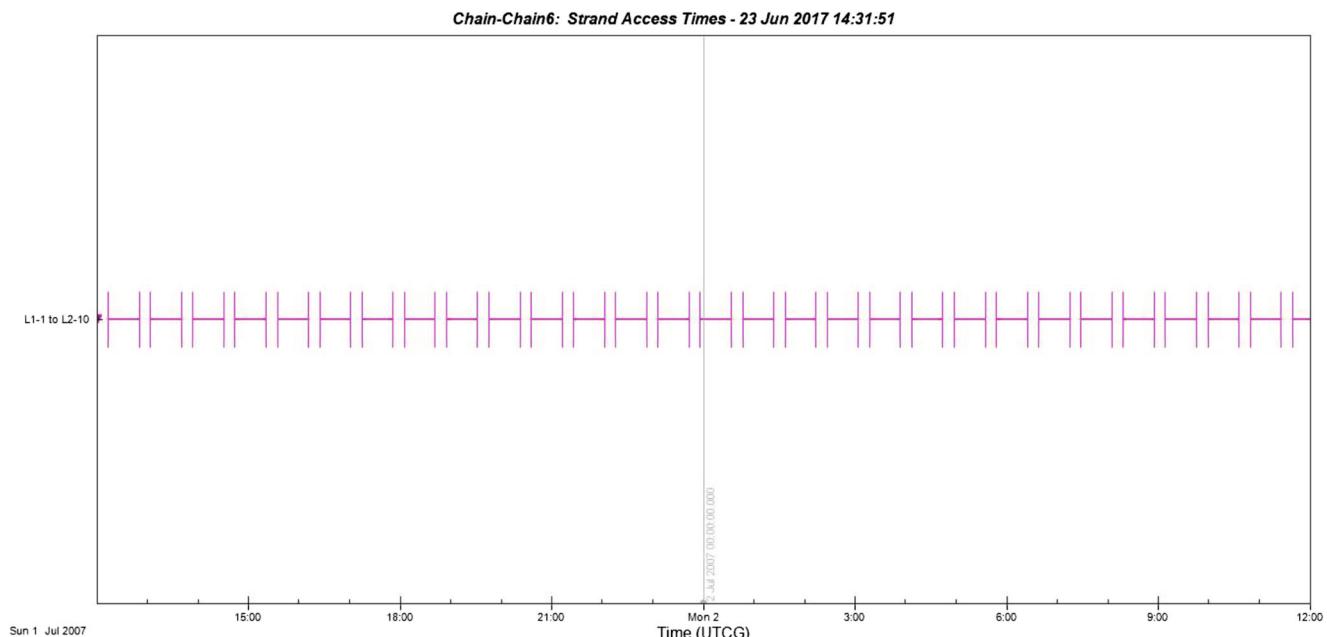


Fig. 6 Link between L1-1 and L2-10 satellites

Fig. 7 Link report between L1-1 and L6-4 satellites

Access	Start Time	Stop Time	Duration (sec)
3	1 Jul 2007 13:43:04.086	1 Jul 2007 13:56:01.278	777.192
4	1 Jul 2007 14:33:16.204	1 Jul 2007 14:46:13.393	777.189
5	1 Jul 2007 15:23:28.320	1 Jul 2007 15:36:25.510	777.190
6	1 Jul 2007 16:13:40.437	1 Jul 2007 16:26:37.628	777.191
7	1 Jul 2007 17:03:52.554	1 Jul 2007 17:16:49.748	777.194
8	1 Jul 2007 17:54:04.674	1 Jul 2007 18:07:01.865	777.192
9	1 Jul 2007 18:44:16.786	1 Jul 2007 18:57:13.983	777.197
10	1 Jul 2007 19:34:28.909	1 Jul 2007 19:47:26.100	777.192
11	1 Jul 2007 20:24:41.022	1 Jul 2007 20:37:38.218	777.197
12	1 Jul 2007 21:14:53.144	1 Jul 2007 21:27:50.333	777.189
13	1 Jul 2007 22:05:05.262	1 Jul 2007 22:18:02.451	777.189
14	1 Jul 2007 22:55:17.379	1 Jul 2007 23:08:14.568	777.189
15	1 Jul 2007 23:45:29.497	1 Jul 2007 23:58:26.686	777.189
16	2 Jul 2007 00:35:41.613	2 Jul 2007 00:48:38.803	777.190
17	2 Jul 2007 01:25:53.727	2 Jul 2007 01:38:50.921	777.194
18	2 Jul 2007 02:16:05.848	2 Jul 2007 02:29:03.038	777.191
19	2 Jul 2007 03:06:17.967	2 Jul 2007 03:19:15.156	777.189
20	2 Jul 2007 03:56:30.084	2 Jul 2007 04:09:27.273	777.190

at least be less than 13 minutes. If computing a required path takes too long, the topology of the satellite network will change and the packets that are being transmitted may not find the correct path. If the routing update time is too long, packets will be required to be retransmitted, which will result in higher end-to-end delay and higher packet loss rate.

Therefore, in the process of route discovery, the efficiency of the routing algorithm should be as high as possible, so that it can better adapt to the requirements of the fast changing topology of LEO satellite communication system.

2.1 Software defined network

Software defined network is a new type of network architecture. It makes the whole network architecture more flat and centralized.

Figure 8 shows the SDN framework, the SDN network consists of application layer, control layer and data layer. The core idea is to separate the control plane of the network from the data plane. SDN network adopts centralized management mode and the control layer has a global view of the network, include status information and load conditions of all devices in the network, therefore, the control layer in SDN network can control and manage the network more reasonably according to the real-time information of the network. It can effectively improve the stability and security of the network.

In SDN network, the underlying communication equipment transmits data according to control command from the control layer, which is specified according to the flow table items sent by the control layer. There is a communication link between every device and the controller, and it follows the OpenFlow protocol.

Network control is directly programmable because it is decoupled from forwarding functions, making the deployment of network protocols and the expansion of networks more convenient.

Figure 9 shows the communication mode between different layers of SDN network. The communication between the SDN control layer and the application layer is communicated via the northbound interface, and its interface protocol has not yet been unified. It can be seen from Fig. 9, control layer provides a lot of extensible API interface for the application layer, each API interface corresponds to an application, so the SDN can carry a variety of network services. Because of the flexible extensibility, SDN can

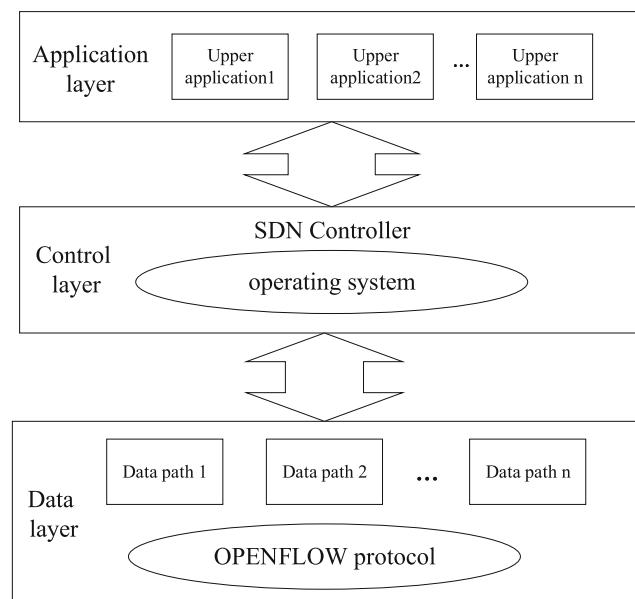


Fig. 8 SDN Framework

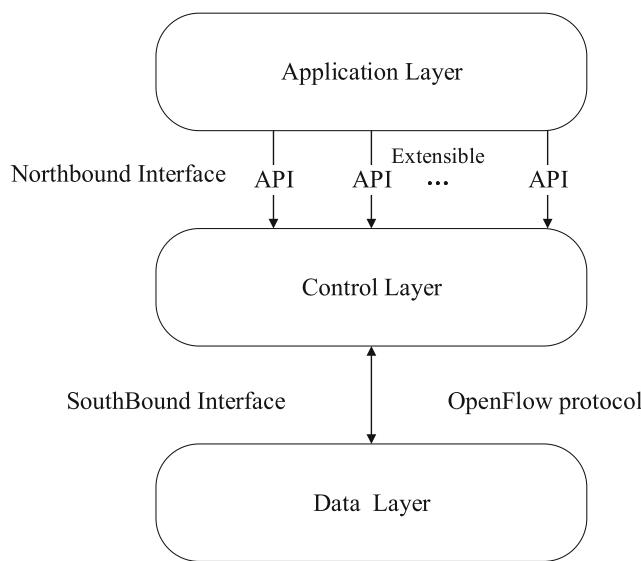


Fig. 9 SDN layer communication mode

provide the required API interface for the new web service in time. In the SDN, the underlying switches and control layer communicate through the southbound interface. The controller sends the corresponding flow table to it through the interface, and then the data transition is performed by

the switches. The protocol to the southbound interface is the OpenFlow protocol.

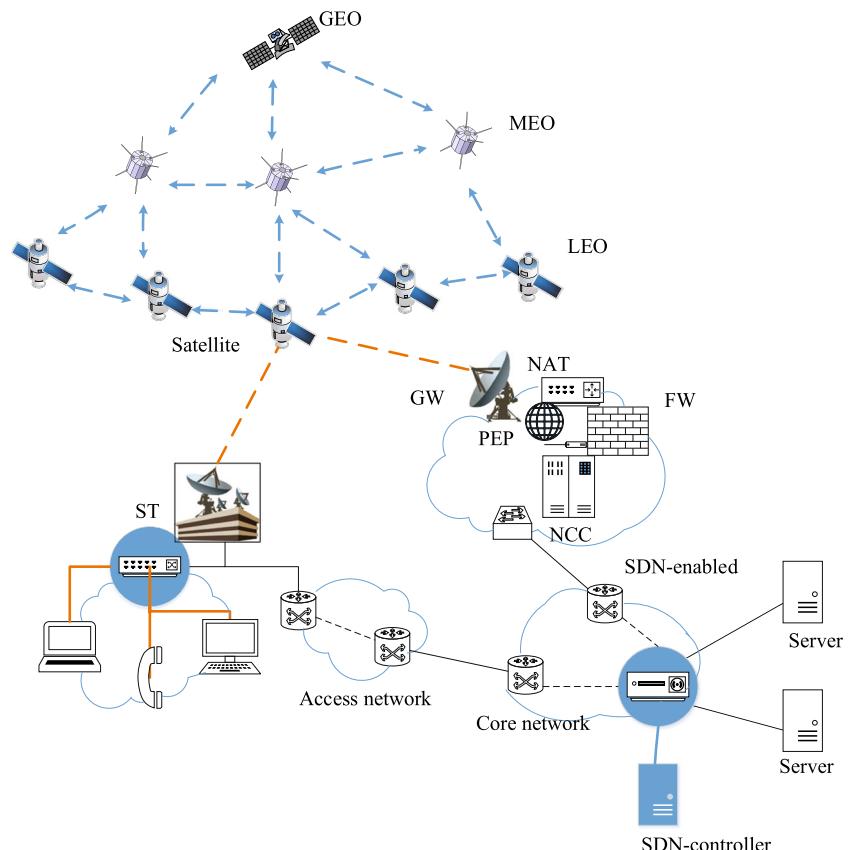
So we can see that SDN is an architecture purporting to be dynamic, manageable, cost-effective, and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications.

2.2 Space-based network based on SDN

The key problems in space-based network are long delay, high bit error rate and high interruption probability. The topology of LEO satellite network is highly dynamic and the satellite communication has high bandwidth. So the SDN network is suitable for space-based network. With the increase of the number of satellites in LEO satellite network, the management of the network has become even more difficult. Combining SDN with traditional space-based satellite network makes management and maintenance easier and can better improve the performance of the network.

Figure 10 shows the satellite network framework based on SDN. The LEO satellite is in the data layer of the SDN network, they are based on OpenFlow protocol and transmitting data according to the instruction of the controller. The ground control station is in the control layer of the SDN network and it has a full network view, so that the ground

Fig. 10 Satellite network framework based on SDN



control station can provide the best path for data transmission as well as real-time maintenance and management of the network.

3 Routing algorithm for leo network

3.1 Routing algorithm based on virtual topology

Figure 11 shows the basic idea of the routing algorithm based on the virtual topology. The main idea is to divide the satellite period into several short time slices, that is $[t_0, t_1], [t_1, t_2], [t_2, t_3], \dots, [t_{n-1}, t_n]$. Because the topology of satellite network exhibits periodic changes, the position of satellites in orbit can be predicted, and the dynamic satellite network topology is regarded as a static topology in each time slice as long as the time slice is small enough. The main advantage of the routing algorithm based on the idea of virtual topology is that it considers the topological structure of the high speed low orbit satellite network as a series of continuous static topological structures sorted by time slices, routing can be computed on the corresponding static topology at each time interval, thus enabling smaller routing computation expense. The disadvantage of it is that it needs a lot of storage space to store the routing information of each time. Moreover, the routing algorithm based on the idea of virtual topology is badly adapt to the congestion of the network link in real-time, the change of network traffic flow and the real-time processing of network fault.

As Fig. 12 shows, in order to solve the problem that the poor adaptability of the algorithm to the network unexpected situation, we can compute the static routes corresponding to each time slice in real time.

When calculating the corresponding route for each time slice, we consider the DFS algorithm and the Dijkstra algorithm, and integrated use of two algorithms to improve the efficiency of calculation.

3.2 Depth-first-search algorithm

DFS algorithm is a traditional traversal search algorithm in routing calculation, given the source node and the destination

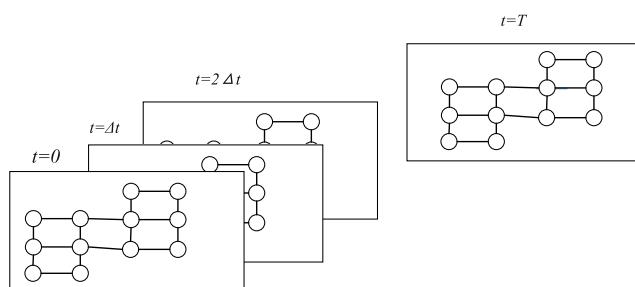


Fig. 11 Topology based on virtual topology strategy

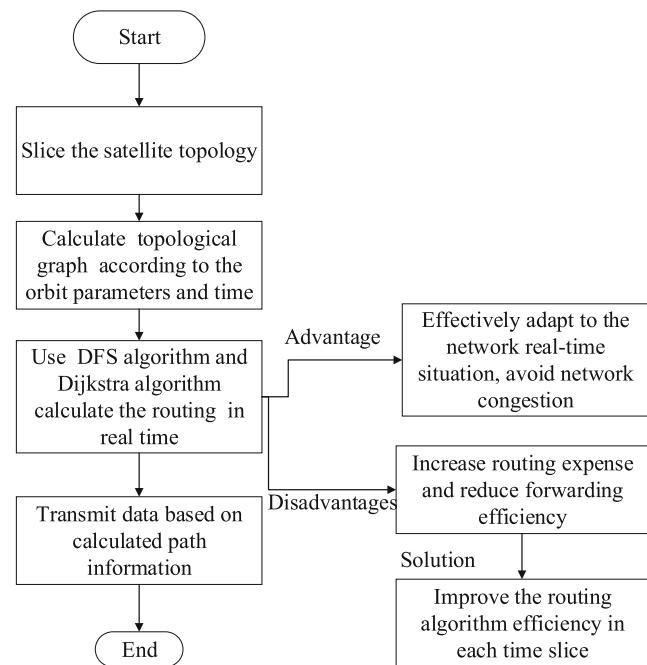


Fig. 12 Advanced routing algorithm based on virtual topology

node, DFS can find all the path between two nodes. DFS algorithm is used to calculate the desired path between two points, the algorithm follows a search strategy for vertical depth search of a directed graph or an undirected graph structure. For a branch under a node, only when the bottom of the branch is searched, is it returned to the starting point of the branch and searched for the next branch.

In depth first search algorithm, searching process starts from the given source node. For the currently searched vertex v , the search continues along the edges between the two vertices if the vertex has an adjacent node that has not been accessed.

Each time we visit a link and the link cost is recorded, so we can know the path cost of each path we find, and choose the better path through comparison.

Search along the vertex v until the edges from the beginning of the v are all searched, and the search will come back to the last node to the vertex v . The end of the process of depth search is the path between the source nodes and all the other nodes is searched. If the destination node is given, each time it is searched to the destination node, the path is marked until all branches to the destination node have been searched, and the entire depth search process is completed.

As Fig. 13 shows, v stands for the source node, D stands for destination node. When we start from v , we call $DFS(v, flag)$, in which $flag$ means whether the destination node D is visited. If destination node D is unvisited, then $flag = 0$ and call $DFS(v_{next}, 0)$ and when the next node does not have adjacent node, we go back to the source node and call $DFS(v, 0)$.

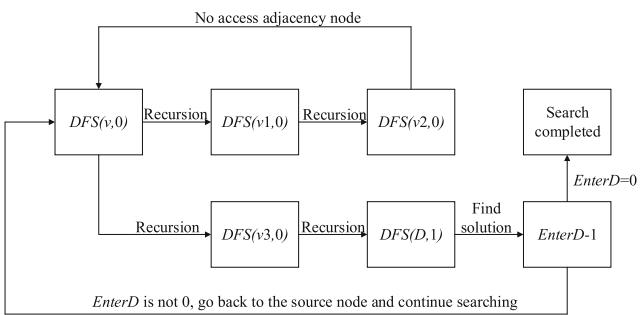


Fig. 13 DFS search process

If destination node D is visited, then $flag = 1$, $EnterD - 1$. $EnterD$ stands for the in-degree of the destination node, if $EnterD = 0$, we go back to the source node and continue searching until $EnterD = 0$, depth search process is complete.

Figure 14 shows that the core idea of the DFS algorithm is recursion. For a given vertex, DFS looks for its adjacency point. For adjacent nodes that are not accessed, we regard this node as a starting point and use DFS algorithm until all the nodes is visited, the entire searching process is complete.

3.3 Joint DFS and Dijkstra algorithm

The Dijkstra algorithm is a classic routing algorithm for finding the shortest path. The algorithm is used to find the

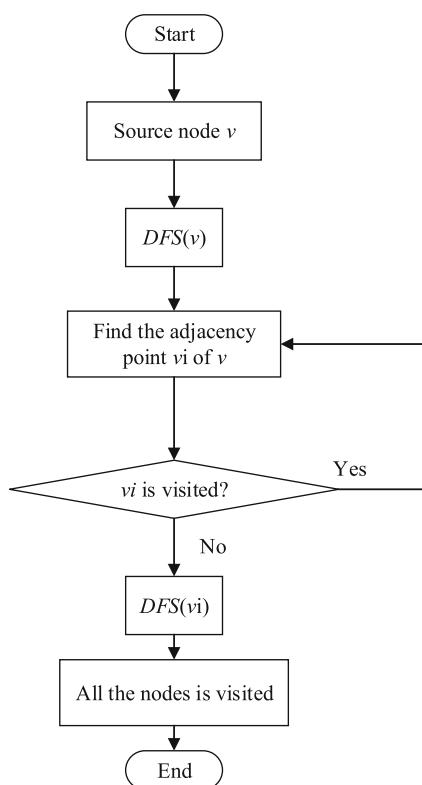


Fig. 14 Main idea of DFS

path with the least link cost from the source node to all other nodes in the network. But for a given source node and the destination node of the directed graph, the classical Dijkstra algorithm needs to be improved appropriately, the cycle will be end as soon as the destination node is marked, output the shortest path between the source node and the destination node and the shortest path link cost.

DFS algorithm can find all the communication paths between the source node and the destination node to meet the communication requirements because they will search each node of the graph. Therefore, all link information can be obtained, and the reliability of the result is better. But because it requires every node in the topological graph to be searched, the time complexity of the algorithm is very high, which is exponential order $O(n!)$. Thus, when the number of satellite nodes in the satellite communication network reaches hundreds or more, the computational efficiency of the algorithm will be very low. The topology of the satellite may have changed as it has not calculated the desired path.

Pruning can be employed in order to improve the efficiency of DFS algorithm. The key idea of pruning is to remove the useless search branches. The commonly used pruning methods are feasible pruning, probabilistic pruning, optimality pruning, etc.

Because the Dijkstra algorithm only needs to find the link with the minimum cost between the source node and the destination node, the time complexity of the algorithm is low, which is square order $O(n^2)$, so the calculation efficiency is higher. But it only finds part of the path with minimal link cost, so that when looking for the path, it may skip the necessary nodes required, and the path found does not meet the requirements, leading to routing failure.

Figure 15 shows one part of the topology of a satellite communication network, in which S represents the source satellite, D represents the destination satellite, and $S1$ is the necessary satellite node. The Dijkstra algorithm will choose the path with smaller link cost according to the link cost, the calculated transport path is $S - S2 - D$, it can be seen that the path does not include the necessary satellite node $S1$.

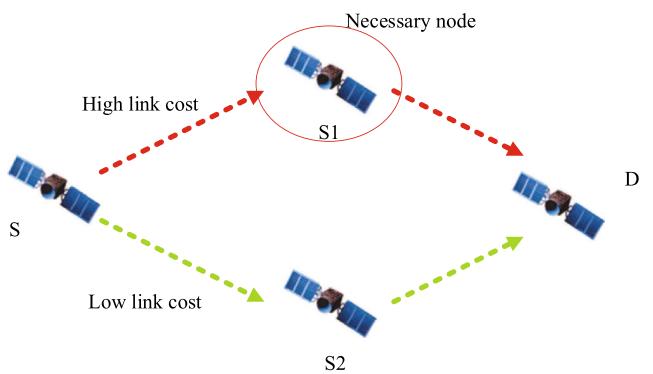


Fig. 15 Partial topology

which does not meet transmission requirements. The DFS algorithm will compute all the transport paths between the source node and the destination node, so the result contains two paths $S - S_2 - D$ and $S - S_1 - D$, according to the communication requirements, thus the transmission path can be selected as $S - S_1 - D$ with requirement. Therefore, when calculating routing information in the LEO satellite network, the advantages of the two algorithms can be taken into account. Thus, combining the two algorithms can not only improve the efficiency of the algorithm, but also improve the reliability of the required path found.

In the case of a large number of satellite nodes, we integrate DFS and Dijkstra algorithms together to improve the efficiency of calculation. According to the network real-time situation, dividing the LEO satellite network into a series of clusters. Thus, the DFS algorithm and the Dijkstra algorithm are adopted according to the link conditions of different groups of stars. The Dijkstra algorithm is used in the group where the network traffic is large and the network congestion is high and the DFS algorithm is adopted at the group where there is a large number of necessary satellites. It can not only improve the efficiency of calculating the communication path, but also avoid the limitation of Dijkstra algorithm, which may cause the result of calculation which dose not meet the requirements.

Figure 16 shows the process of joint DFS and Dijkstra algorithm. Firstly, in the process of satellite communication, we set the source satellite, the destination satellite and the set of necessary satellite nodes. At the source satellite node, the Dijkstra algorithm is used to find the path with the minimum link cost from other satellite nodes to the source satellite nodes. When the path with minimal link cost between the first found necessary satellite node and the source satellite is found, the Dijkstra algorithm ends and $Subsetnode-1$, in which $Subsetnode$ stands for the number of necessary satellite node required. Then, we use the first found necessary satellite node N_1 as start node, adopting the DFS algorithm, that is $DFS(N_1)$, every time a necessary node N_i is searched, checking whether $Subsetnode$ is equal to 0, if $Subsetnode \neq 0$, we have and continue using the DFS algorithm for deep searching, that is $DFS(N_i)$, else if $Subsetnode = 0$, the DFS algorithm ends. At last, we use the last found necessary satellite node as the start node and adopt the Dijkstra algorithm to find the path with minimal link cost between the last found necessary satellite node and the destination satellite. By integrating the above steps, we can find a communication link that meets the requirements.

In this way, the computational time can be greatly reduced by reducing the size of the problem scale of the DFS algorithm, and because the use of depth first search does not skip the necessary satellite nodes, the reliability of the path is improved. The Dijkstra algorithm can effectively avoid the satellite nodes with larger traffic distribution

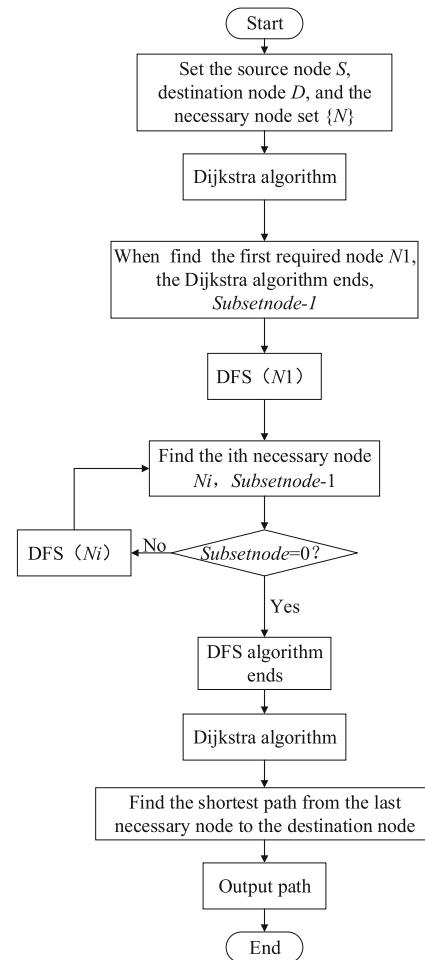


Fig. 16 Joint DFS and Dijkstra algorithm

in the satellite group beyond the necessary satellite nodes, which reduces the queuing time of data packets in the on-board memory, and reduces the average end-to-end delay of data communications, thus improving the effectiveness of the satellite network.

4 Simulation and analysis

4.1 Original DFS algorithm

DFS algorithm can find a path with a given source node, a destination node and necessary nodes set to meet the communication requirements. With DFS algorithm, each node of the given network topology will be searched. We will set flags to the nodes required to go through, and flag will be marked when the required node is went through. When all the flags are marked, the communication meeting the communication requirement is precisely found.

Figure 17 shows the comparison of routing discovery time of the DFS algorithm as the number of nodes changes

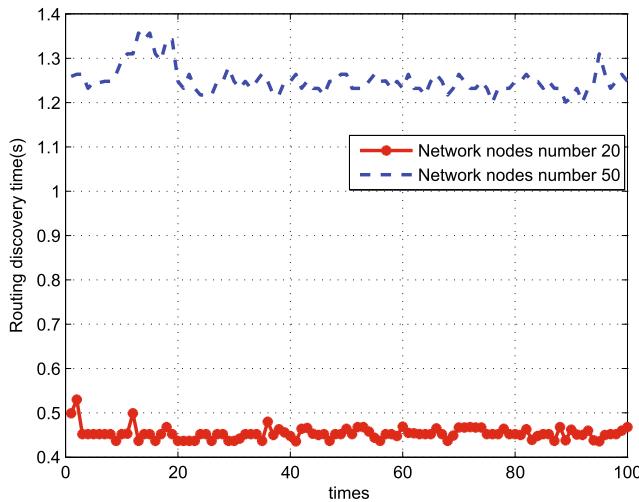


Fig. 17 Routing discovery time of DFS algorithm with 4 necessary nodes

in the network. When the number of nodes in the network is 20, and the number of necessary nodes is 4, the time to calculate a required path is about 0.4s~0.5s. When the number of nodes in the network increases to 50, and the number of necessary nodes remain unchanged, the time to calculate a required path is 1.2s~1.3s, the computation time increases by more than two times when the number of nodes is 20.

The DFS algorithm requires to go through the whole network topology, and the algorithm complexity is very high, which is exponential complexity $O(n!)$, the corresponding time frequency function is $T(n) = C * n!$, where C is a constant. It can be seen from the time frequency function that, with the increase of the number of satellite nodes in the network, DFS algorithm calculates the path satisfying the requirements of the communication with an exponential time increase.

Figure 18 shows calculation result of the DFS algorithm when the number of nodes in the network is 300. Having ran for 10 minutes, DFS algorithm did not find the communication path needed. After 10min, the network topology of LEO

```
D:\C\CodeB\bin\Debug\CodeB.exe D:\C\CodeB\topo.csv D:\C\CodeB\demand.csv D:\C\CodeB\Co...
Total number of edges:844
Total number of condition:19, 87, 55122133120
Here is the condition:know that the number of nodes is 300
Start_node:19
End_node:87
Total_subset_node:5
55 22 33 28 64
Here is the statistics of the topology
Max_node_ID:299
Total_valid_node:300
Total_skip_node:0
Total_only_out_node:40
Total_only_in_node:0
Total_discarded_edge:0
DFS calculates for 10 minutes,
no results were found
```

Fig. 18 Performance of DFS algorithm when nodes is 300

satellite network will change and be not consistent with the requirements of the routing algorithm based on virtual topology. Thus the efficiency of the DFS algorithm is very low when the number of nodes in the network is very large.

So the key factor affecting the performance of DFS algorithm is the number of nodes in the network. When the number of nodes in the network is large, the performance of the DFS algorithm is poor.

4.2 Joint DFS and Dijkstra algorithm

The DFS algorithm is a classic routing algorithm. Through analysis we can see the time complexity of the DFS algorithm is very large, so the efficiency of DFS algorithm is greatly affected by the scale of the problem. The problem refers to the number of nodes in the network. Therefore, in order to improve the efficiency of the algorithm, we can combine the algorithm with lower time complexity, and make use of the advantages of the two algorithms to obtain better routing performance.

The Dijkstra algorithm aims to find the shortest path from all other nodes in the network to the source node. For a given destination node, the Dijkstra algorithm ends the calculation by finding the shortest path between the source and destination nodes, which avoids the additional computation cost of additional routing computation.

Figure 19 shows the simulation result of the routing discovery time of the DFS algorithm and the joint DFS and Dijkstra algorithm. When the number of nodes in the network is 20, the computation time of using the joint DFS and Dijkstra algorithm is about 0.4s~0.5s, and the computation time is basically the same as that of DFS algorithm. It can be seen that when the number of nodes in the network topology is small, the efficiency of the DFS algorithm and the DFS and Dijkstra algorithm is basically the same. When the

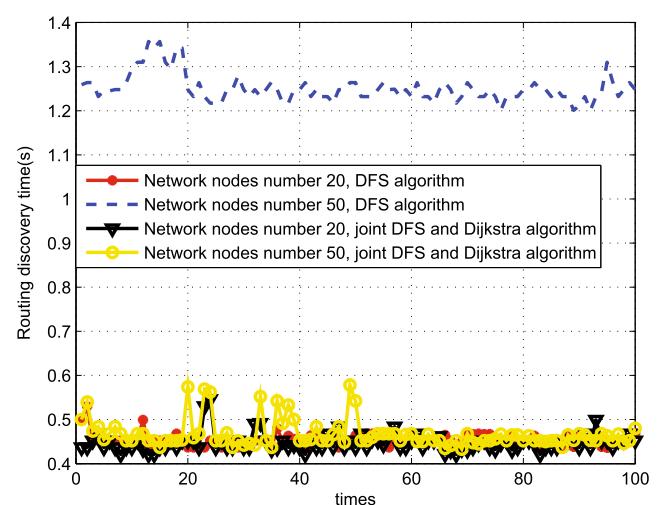


Fig. 19 Comparison between DFS algorithm and joint algorithm

number of nodes in the network increases to 50, the computation time of using the DFS and Dijkstra algorithm is increasing, but it is a small increase, and the average computing time is still 0.4s~0.5s. But the computation time of DFS algorithm is increased to 1.2s~1.3s. It can be seen that compared with the DFS algorithm the computation time of the joint DFS and Dijkstra algorithm is less affected by the increase of the number of nodes in the network.

Since the Dijkstra algorithm only considers the path with smaller link cost when routing, in the calculation of the Dijkstra algorithm, the considered link cost includes the distance between two nodes and the link load between two nodes, so it avoids many costly branches of the link. Therefore, the size of the network branches gone through by the Dijkstra algorithm is greatly reduced, so the time complexity of the Dijkstra algorithm is low, which is square order complexity $O(n^2)$. The corresponding time frequency function is $T(n) = C * n^2$, where C is a constant. It can be seen from the time frequency function that with the increase of the number of nodes in the network, the time complexity of the Dijkstra algorithm grows in square, compared to exponential complexity $O(n!)$, the time complexity of the Dijkstra algorithm is very low, so when the number of nodes in the network is large, the computational efficiency of the joint DFS and Dijkstra algorithm is better than that of the DFS algorithm.

From the above analysis, we can know that the computation efficiency of DFS algorithm is very low when the number of nodes in the network reaches hundreds, it does not meet the requirements of the routing algorithm in satellite networks. However, the computational efficiency of the joint DFS and Dijkstra algorithm is still relatively high, it can better adapt to the routing computation in satellite networks.

Figure 20 shows how the routing discovery time of the joint DFS and Dijkstra algorithm changes when the number of nodes changes in the network. It can be seen that the computation time of the joint DFS and Dijkstra algorithm is less affected by the increase of the number of nodes in the network. With the increase of the number of nodes in the network, the time to calculate the required path increases in general, but the trend is very small. When the number of nodes in the network is 20, the routing discovery time is about 0.4s~0.5s; when the number of nodes is 50, the route discovery time is still 0.4s~0.5s. When there are hundreds of nodes in the network, the DFS algorithm can not get result in required time, so it does not meet the requirement of routing calculation in the LEO satellite network. But the joint algorithm can still calculate the required communication path in a relatively short time. When the number of nodes in the network is 300, the routing discovery time is about 0.5s~0.6s. When the number of nodes in the network increases to 600, the route discovery time is

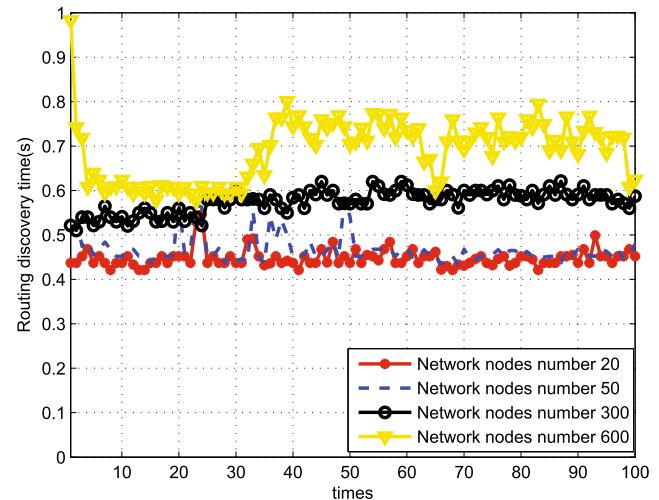


Fig. 20 Routing discovery time of the joint algorithm with 4 necessary nodes

about 0.6s~0.8s. It can be seen that the routing discovery time is less than 1s when the number of nodes is 600, within this shorter time interval, the dynamic topology of the LEO satellite communication network can be viewed as fixed, the route of satellite network can be calculated according to the method of calculating the route of the static topology, thus effectively shielding the dynamics of the topology of the satellite communication network.

From the above analysis, we can see that the number of nodes in the network has little influence on the computational efficiency of the joint DFS and Dijkstra algorithm and it is not the core factor that affects the efficiency of the algorithm.

Figure 21 shows the influence of the number of necessary nodes on the routing discovery time of the joint DFS and

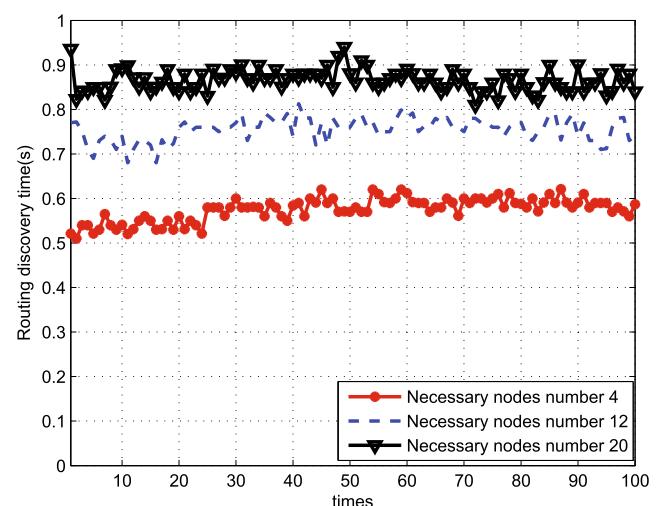


Fig. 21 Influence of necessary nodes number on the joint DFS and Dijkstra algorithm

Dijkstra algorithm. The number of the nodes in this simulation is 300. When the number of necessary nodes is 4, the route discovery time is about 0.5s~0.6s. When the number of necessary nodes is 12, the route discovery time is about 0.7s~0.8s and when the number of necessary nodes is 20, the route discovery time is about 0.8s~0.9s.

It can be seen that the number of necessary nodes has greater impact on the computational efficiency of the algorithm more than the number of nodes in the network. Because the idea of the joint DFS and Dijkstra algorithm is to use the Dijkstra algorithm to find the first necessary node. Then, the node is marked and the first found node is used as the first node for DFS search until all the necessary nodes are searched, the DFS search is over. Then Using the last found necessary node as the starting node finds the shortest path between it and the destination node with the Dijkstra algorithm. This not only improves the efficiency of the algorithm, but also avoids the fact that the necessary nodes are not in the calculated path.

The number of necessary nodes actually determines the scale of the problem of the DFS algorithm. The time complexity of DFS algorithm is $O(n!)$, but the time complexity of the Dijkstra algorithm is only $O(n^2)$, therefore as the number of network nodes increases, the DFS algorithm runs much longer than the Dijkstra algorithm. Actually, the computational efficiency of the algorithm is affected more by the number of necessary nodes than by the number of nodes in the network topology and the algorithm should be optimized according to the given conditions when the algorithm is improved.

4.3 Joint DFS and Dijkstra Algrithom in iridium system

To verify the performance of the proposed algorithm, we refer to iridium system model. We define that the latitude between 80 and 90 degrees is the polar region. When the satellite is running into the polar region, the satellite orbit is not connected by the inter satellite links, and the satellites can not communicate with each other. The topological structure of the iridium system is written with C language and the operating environment is VS2013.

Due to the use of routing algorithm based on virtual topology, and the 100 minutes operation cycle of iridium system, the iridium network is decomposed into 100 continuous time slices, each time slice is 1 minutes. The duration of the inter satellite links between two satellites in adjacent orbits is 13 minutes, thus the topological structure of the iridium system can be regarded as fixed within 1 minute and its dynamic topology is decomposed to 100 static topology structure.

For the 100 static topologies generated, respectively using the DFS algorithm and the joint DFS and Dijkstra algorithm for calculation. And the routing discovery time of

each topology and the link cost of the corresponding path are stored, and the average end-to-end delay is calculated.

Figure 22 shows the comparison of route discovery time between two algorithms. It can be seen that when calculating the 100 consecutive static topology on the iridium system, the route discovery time of the DFS algorithm is more than the joint DFS and Dijkstra algorithm. The routing discovery time of the DFS algorithm is about 0.5s~0.6s, the average is about 0.55s and the routing discovery time of the joint DFS and Dijkstra algorithm is about 0.4s~0.5s, with an average of about 0.45s.

Since the joint DFS and Dijkstra algorithm integrated using two algorithms at a time, it greatly reduces the time complexity of the algorithm so when the number of nodes in the network and the number of the necessary nodes is fixed, the computational efficiency of the joint DFS and Dijkstra algorithm is better than that of the DFS algorithm.

Figure 23 shows the comparison of average end-to-end delay in Iridium system with using the two algorithms. The transmission delay is calculated according to the link cost of the transmission path corresponding to each topology, the transmission path is calculated repetitively by the DFS algorithm and the joint DFS and Dijkstra algorithm. It can be seen that the average end-to-end delay is less than that calculated by the DFS algorithm when the packets are transmitted by the path calculated by the joint DFS and Dijkstra algorithm. The average end-to-end delay of the path calculated by the joint DFS and Dijkstra algorithm is 0.07s~0.1s, while the average end-to-end delay of the path calculated by the DFS is 0.1s~0.15s.

Since the joint DFS and Dijkstra algorithm considers the advantages of the two algorithms at a time and the Dijkstra algorithm could avoid the link with higher cost, so the link cost of the calculated path for data transmitting is less than

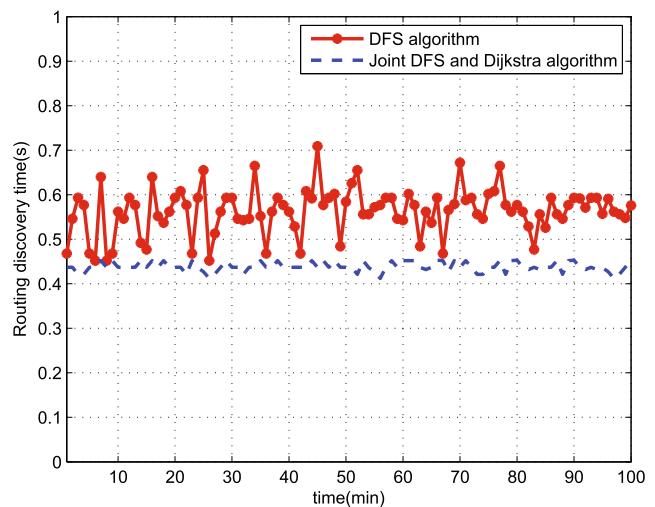


Fig. 22 Comparison of route discovery time

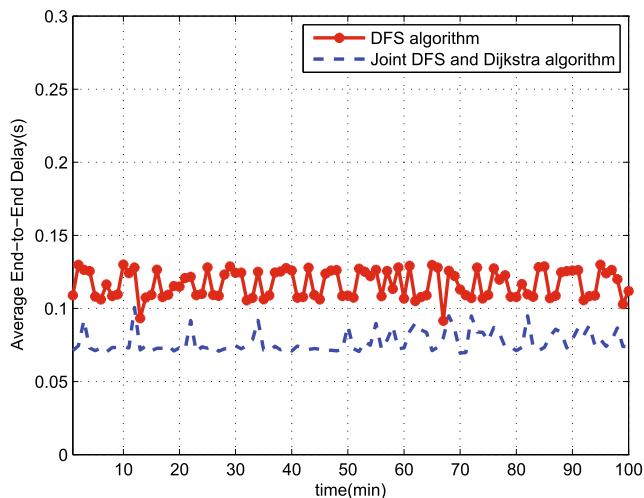


Fig. 23 Comparison of average end-to-end delay

the link cost calculated by the DFS algorithm. The corresponding average end-to-end delay of the data transmission is decreasing as well.

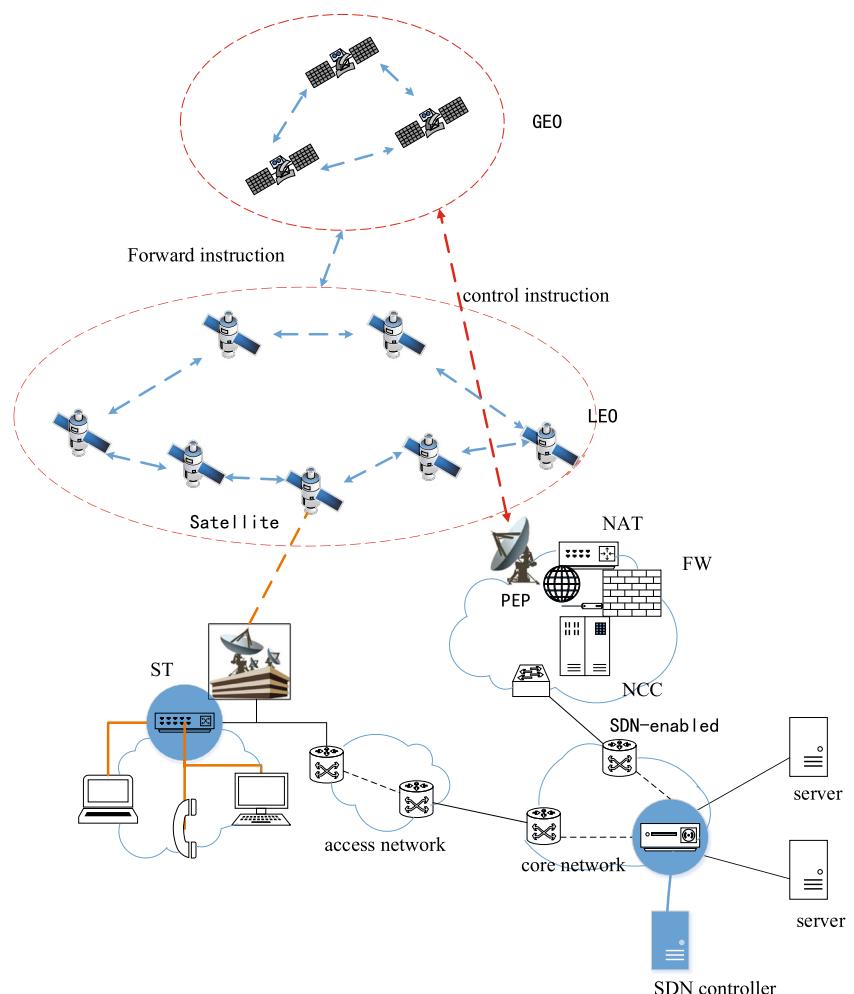
4.4 Proposed algorithm for hybrid space-based network based on SDN

This section focuses on the performance simulation and analysis of SDN based LEO networks and traditional LEO network.

At present, due to the rapid increase in the number of network users, along with the rapid growth of communications business, the requirement of the quality of communication services for users is getting higher and higher, fast access to the network in real time is the key factor to ensure the quality of network service. In order to realize the global seamless coverage of satellite signals, the number of satellite nodes in satellite networks is also increasing, so the ability to grasp the whole network becomes the core issue of satellite communication.

In the SDN network, the controller has a global view of the network and plays the role in the unified configuration and management of the network. It meets the requirements of satellite communication network development. So we integrate SDN's ideas with traditional space-based networks.

Fig. 24 Hybrid space-based network based on SDN



The space-based network based on SDN is mainly composed of satellite network and terrestrial network. The satellite networks include LEO satellites, MEO satellites, and GEO satellites. In the ground network, the server, network control center, firewall and PEP are in the application layer of SDN network, they access the SDN control layer through the core network. The ground control station is in the SDN control layer, it grasps the whole network information by interacting with the units in the network to get the real time information, controls and manages the network at the same time. The control station sends the instructions to the GEO satellites in the space-based network, and the GEO satellites translate and transmit these instructions to the LEO satellites to control the data transmitted in the network.

Figure 24 shows the network infrastructure of hybrid space-based network which is composed of GEO and LEO based on SDN. When there is a new satellite in the LEO satellite network, the ground station will link with the satellite and establish a communication link, at the same time, the corresponding configuration information is sent to the GEO; the control instructions are translated by the GEO and sent to the LEO satellite, all the nodes in the LEO network are updated uniformly so that a satellite is connected to the network. As a result of the command issued by the control station, and then unified configuration to the LEO satellite by GEO, this greatly improves the efficiency of satellite access, and makes the network of the space-based network becoming more flexible.

SDN network separates the data plane and the control plane, using the network centralized control mode, so it can realize the rapid deployment of updated information of the satellite network, including the switching of the network link, the data traffic in the network distribution, network topology changes and other information. SDN provides a

flexible, detailed and scalable network architecture for traditional LEO network. Figure 25 shows the comparison of average end-to-end delay performance between LEO networks based on SDN and traditional LEO networks. It can be seen that the average end-to-end delay of SDN based LEO network is generally lower than that of traditional LEO networks.

Since the ground control station of SDN based LEO network is in the control layer of the network, it has all the information of satellite nodes in the network, including the load of satellite nodes, the link condition of inter satellite links. Then, the controller collects the information in real time, analyzes and processes the traffic information of the whole network, and calculates the best transmitting path of the packet. So when the packet transmission is carried out, the region with dense network traffic and nodes with large load are avoided, thus effectively avoid the packet waiting for so long in the satellite node buffer queue and the transmission delay is effectively reduced. And because the satellite node is only responsible for data transmitting, it greatly reduces the additional power cost of satellite equipment, improves the data forwarding efficiency of satellite nodes and further reduces the data transmission delay.

Figure 26 shows the comparison of packet loss rate performance between LEO networks based on SDN and traditional LEO networks. It can be seen that the packet loss rate of SDN based LEO network is generally lower than that of traditional LEO networks.

Because in SDN network, the path for data transmitting is the best transmission path calculated according to the whole network traffic and node information, it effectively avoids packets going through the nodes with heavy load, thereby reducing the probability of packet retransmission or loss due to packet waiting in the buffered queue for a long time.

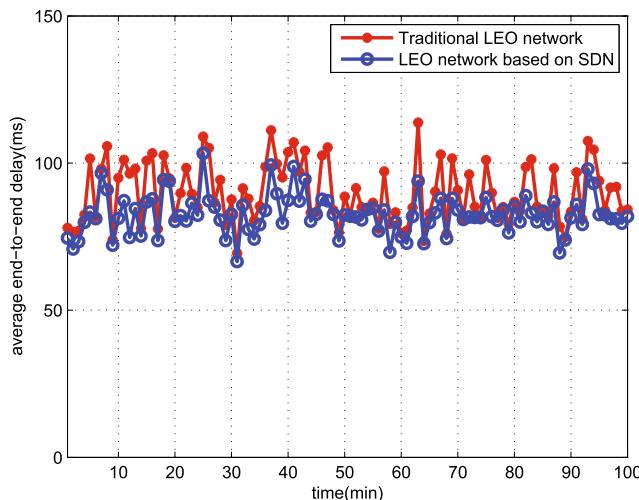


Fig. 25 Comparison of average end-to-end delay

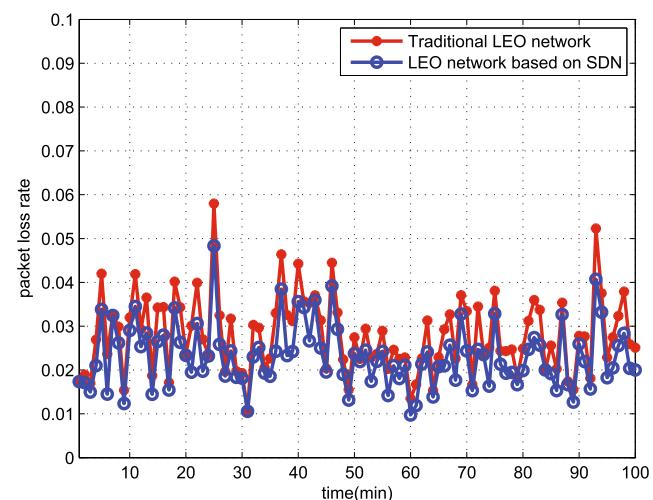


Fig. 26 Packet loss rate comparison

And in the LEO network based on SDN, the control layer can grasp the condition of communication link in real-time, and can maintain the stability of the whole network in real-time according to the link condition and then reduce the probability of communication interruption. Therefore, the application of SDN network structure improves the stability of the network, so the data transmission is more stable, which effectively reduces the packet loss rate in the entire data communication process.

Based on the above analysis, we can see that the combination of SDN can break through the limitations of traditional network architecture and improve the communication performance of the network.

5 Conclusion

In this paper, a LEO satellite network model is built, and the inter-satellite link characteristics in the model are analyzed. When considering the routing algorithm in LEO network, we adopt the routing algorithm based on the idea of virtual topology. The algorithm decomposes the operating cycle of the satellite into continuous time slices, thus the satellite topology can be regarded as a static topology in the period of time as long as the time slice is short enough. Thus, the efficiency of routing algorithm dealing with each time slice decides the performance of the routing algorithm based on virtual topology. When routing the static topology within each time slice, the idea of DFS algorithm is adopted to improve the computational efficiency and the reliability. Based on it, a novel algorithm joint DFS and the Dijkstra algorithm is proposed for the huge numbers of LEO mobile satellite network based on SDN. As the simulation results show, the performance of the joint DFS and Dijkstra algorithm is better than the traditional DFS algorithm. Moreover, the communication performance of LEO network based on SDN and traditional LEO network is compared and analyzed. The results show that the LEO network based on the SDN architecture breaks through the performance limitations of the traditional network architecture.

Acknowledgements This work was supported by National Science Foundations of China (No.61671183 and 91438205) and the Open Research Fund of State Key Laboratory of Space-Ground Integrated Information Technology under grant NO.2015_SGIIT_KFJJ_TX_02.

References

- He Jf, Yong J, Li Gx (2008) Topology control technology based on dynamical inter-orbit ISL linkage in LEO satellite constellation communication system, 1-6. In: 2nd International Symposium on Systems and Control in Aerospace and Astronautics, Shenzhen
- Liu Y, Xu W, Tang F, Kuang L, Yang Y (2014) A novel routing algorithm based on virtual topology snapshot in LEO Satellite NetworksBook title. In: IEEE 17th International Conference on Computational Science and Engineering, Chengdu, pp 357–361
- Liu Y, Xu W, Tang F, Kuang L, Yang Y (2016) An improved multi-path routing algorithm for LEO satellite networks, 1101–1105. IEEE Trustcom/bigdataSE/ISPA, Tianjin
- Qingfen L, Yifei W, Fei T, Jian Y, Yu L (2014) An islanding surface searching approach based on Floyd-warshall and DFS algorithm. In: International Symposium on Power Electronics, Electrical Drives, Automation and Motion, Ischia, pp 84–87
- Ibrahim MAM, Xinda L, Rwakarambi JM (2001) Parallel execution of an irregular algorithm depth first search (DFS) on heterogeneous clusters of workstation. In: Proceedings of the International Conferences on Info-Tech and Info-Net, vol 3, pp 328–322
- Dijkstra EW (1959) A note on two problems in connection with graphs. Numerische Math 1(1):29–271
- Alvizu R, Maier G, Kukreja N, Pattavina A, Morro R, Capello A, Cavazzoni C (2017) Comprehensive survey on t-SDN: software-defined networking for transport networks. IEEE Commun. Surv. Tutorials PP:1–1
- Sato G, Uchida N, Shibatalg Y (2015) Performance evaluation of software defined and cognitive wireless network based disaster resilient system. In: IEEE International Symposium on Systems Engineering, Volume, pp 332–337
- Nazari S, Du P, Gerla M, Hoffmann C, Kim JH, Capone A (2016) Software defined naval network for satellite communications (SDN-SAT). In: IEEE Military Communications Conference, Baltimore, pp 360–366
- Li T, Zhou H, Luo H, Xu Q, Ye Y (2016) Using SDN and NFV to implement satellite communication networks. In: 2016 International Conference on Networking and Network Applications, Hakodate, pp 131–134
- Zhang Y, Wang Y (2016) SDN based ICN architecture for the future integration network, 474–47. In: 16th International Symposium on Communications and Information Technologies, Qingdao
- Min J, Xuemai G, Qing G, Wei X, Naitong Z (2016) Broadband hybrid satellite-terrestrial communication systems based on cognitive radio towards 5G. IEEE Wirel Commun 23(6):96–106
- Min J, Xin L, Xuemai G, Qing G (2017) Joint cooperative spectrum sensing and channel selection optimization for satellite communication systems based on cognitive radio. Int J Satell Commun Netw 35(2):139–150



Min Jia received her M.Sc degree in information and communication engineering from Harbin Institute of Technology (HIT) in 2006, and her Ph.D. degree from SungKyungKwan University of Korea and HIT in 2010. She is currently an associate professor and Ph.D supervisor at the Communication Research Center and School of Electronics and Information Engineering, HIT. Her research interests focus on advanced mobile communication technology for 5G and LTE, cognitive radio, digital signal processing, machine learning and broadband satellite communications.



Siyu Zhu received bachelor degree in communication engineering from Harbin Institute of Technology, in 2015 and master of engineering degree in electronics and communication engineering from Harbin Institute of Technology, in 2017. His research interests include routing algorithm and satellite communication.



Haitao Wang received his PhD degree from Tianjin University. He is currently a professor and the director of the State Key Laboratory of the Space-Ground Integrated Information Technology.



Linfang Wang received bachelor degree in communication engineering from Harbin Institute Technology (Weihai), in 2016. He is now a graduate student in school of electricity and information engineering, in Harbin Institute Technology. His research interests include non-orthogonal multiple access technology.



Zhihui Liu received her B.S. degree in information and computing science, North University of China, Taiyuan, Shanxi, China, in 2009, her M.S. degree in computational mathematics, Xiangtan University, Xiangtan, Hunan, in 2012, and her Ph.D. degree at the school of Information and Communication Engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in signal and information processing. She is currently

an engineer in State Key Laboratory of Space-Ground Integrated Information Technology. Her research interests include cognitive radio networks, green communication, wireless communication and satellite communication networks.