MPTCP Based Load Balancing Mechanism in Software Defined Satellite Networks

Ziyi Ma[1,2], Xiaoqiang Di[1,2(✉)], Jinqing Li[1,2], Ligang Cong[1,2], and Ping Li[1]

[1] School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China
dixiaoqiang@cust.edu.cn
[2] Jilin Province Key Laboratory of Network and Information Security, Changchun, China

Abstract. The instability of satellite network links, unbalanced user distribution, and limited resources have caused severe network congestion and load imbalance. To address this problem, we propose a load balancing mechanism based on Multipath TCP (MPTCP) and apply it to software defined satellite networks (SDSN). This mechanism obtains the topology and the state of the whole network through the monitoring function of controller, and then selects k shortest paths from all available paths. It first divides the flow into several subflows according to the cost of different paths, and then establishes the MPTCP connection. To improve the efficiency of the network, we install OpenFlow rules to each switch. In the end, we conduct experiments in the Mininet simulation platform. The results show that the proposed mechanism can achieve load balancing of different links and improve the throughput while reducing delay.

Keywords: Multipath TCP · SDSN · Load balancing

1 Introduction

Satellite networks have many advantages such as flexible networking, wide coverage, convenience, and it can also provide various communication services regardless of geographical environment and climatic conditions. It is especially suitable for remote regions where the ground communication coverage is imperfect. The Starlink project currently being developed by SpaceX aims to build a constellation of thousands of satellites that can beam the Internet to the entire globe, including remote regions that currently do not have Internet access [1]. SES's SaT5G project covers geostationary earth orbit (GEO) and medium earth orbit (MEO) satellites to achieve satellite integration and improve the efficiency of 5G services [2]. This shows that the new trend of satellite networking is quantity and multilayer. Due to the high speed of the satellite, the link connection is unstable. When the link is interrupted, the redesign of the route is required. In addition, the Internet users are imbalanced distributed, resulting in congestion on local networks. What is more, satellite resources are relatively limited, so it is difficult to meet the QoS requirements of newly added services. The problems such

M. Jia et al. (Eds.): WiSATS 2019, LNICST 280, pp. 294–302, 2019.
https://doi.org/10.1007/978-3-030-19153-5_30

as the increase of delay and packet loss ratio, also pose a great impact on the communication quality of satellite networks. Therefore, it is important to choose the proper path for data transmission in order to balance the load of the entire network.

Considering the characteristics of the satellite networks mentioned above, multipath TCP (MPTCP) is very suitable for data transmission in satellite networks [3]. MPTCP is an extension of TCP that enables two applications to communicate over a reliable byte stream while utilizing one or more paths. A regular TCP connection can only communicate through one interface, but MPTCP can send data through multiple interfaces simultaneously. This can improve the reliability of parallel transmission for high-bandwidth data streams in a satellite network, and provide users efficient service.

The architecture design of satellite network has been imitating the layered design of terrestrial network, where each layer has different protocols to realize various functions. Unlike the terrestrial network, the satellites move at high speed in the air. So when the satellite network protocols need to be updated, it is more difficult than terrestrial network. Thus, this paper proposes a new type of satellite network architecture, which is based on software-defined networking (SDN). Considering the performance of MPTCP over software defined satellite networks (SDSN) [4], this architecture contains a centralized controller, which provides a platform for planning route of the ground control center in the MPTCP research of the satellite network. The protocol used between the switch and the controller is OpenFlow in this architecture. Subflows on each path are forwarded based on this protocol. In this paper, we first collect topology information and monitor link state in the controller, and then make MPTCP decision to select k shortest paths. Secondly, according to the path cost, we carry out the load balancing strategy, and select different paths for the subflows. Finally, we install these paths to each switch. We teach the switches to make decisions themselves. So the switch does not have to ask the controller what to do every time a packet arrives to the switch.

The rest of the paper is organized as follows. The second part discusses the related work of MPTCP and SDN in satellite networks. In the third part, we analyzes the specific implementation of the MPTCP routing decision in the satellite network. The fourth part introduces the software-defined satellite networks. The fifth part covers the experimental results and performance analysis of the proposed solution. Finally, the sixth part summarizes this paper.

2 Related Work

MPTCP splits the dataflow into separate subflows. Each subflow establishes a regular TCP connection, but the process of handshake contains specific MPTCP options, such as MP_CAPABLE and MP_JOIN. The MP_CAPABLE option is exchanged between the two hosts that establish the connection to indicate that both parties support MPTCP, after which both parties can create a subflow through the SYN packet with the MP_JOIN flag. The main advantage of using MPTCP in satellite networks is to avoid the impact of link disruption on data transmission and to make better use of the available link capacity in the presence of multiple available satellite links. Hwang and Yoo proposed another packet scheduling method [5]. When the difference of delay

between slow and fast paths is large, the slow path will be frozen. It facilitates the transmission of small amount of data on the fast path. Singh et al. [6] compared two solutions of Dynamic Window Coupling and Opportunistic Linked Increase Algorithm. Both schemes can maintain fairness over traditional protocols such as TCP and SCTP in a crowded environment while increasing link throughput under different bottlenecks. The strategy for adding or removing subflows in MPTCP has been studied in many papers, but there are still some problems in effective identification of the available paths between source hosts and destination hosts. It is still a challenge to consider the network status and routing to select the best path. Krupakaran et al. [7] determined the set of paths that exist between the source hosts and destination hosts, and made decisions based on the number of subflows. Baidya et al. [8] used the MPTCP-based slow path adaptation (SPA) to monitor the quality of the path. If the data transmission is slow, MPTCP-SPA will change a path. MPTCP is usually combined with Equal-Cost Multi-Path Routing (ECMP). ECMP uses random hashing to assign subflows to different paths [9]. Different subflows may share the same path, so this method is not ideal. Zannettou et al. [10] adopted an SDN controller supporting MPTCP. Considering both FatTree and Jellyfish topologies, the controller calculates the shortest path and the disjoint path between two hosts. The above works did not take the impact of link load on subflow transmission. Therefore, our focus is on the impact of load balancing on the overall performance of the network.

Due to the limited costs of updating network services and the limited resources of satellite nodes, many new algorithms and protocols cannot be deployed to satellite networks. This limits the implementation of new technologies and makes it difficult to meet the diverse need of the users. In this case, SDN is introduced into the satellite networks, and new application functions can be achieved [11]. In [12], the author proposed a multi-layer satellite network architecture based on SDN. GEO controls the forwarding of dataflow in the data plane of the LEO/MEO according to commands given by the network operation and control center on the ground. SERvICE [13] used SDN to carry out QoS-oriented routing and bandwidth allocation algorithm to ensure the QoS requirements of multiple users. [14] explored SDN and NFV technologies to provide further innovative services and flexible operations for satellite networks. The difference with the above papers in that we pay more attention to the use of MPTCP to optimize network load between SDN and the terminal host.

3 The MPTCP-SDN Architecture Design

Our MPTCP-based path load balancing strategy applies to the software-defined satellite network architecture. This architecture, which is shown in Fig. 1, is characterized by the separation of the control plane and the data plane. Load balancing is achieved across multiple paths between the source node and the destination node. At the same time, network resources should be made full use of to ensure the correctness of the path. Therefore, the controller selects k shortest paths by using the topology discovery module and the traffic state monitoring module. This helps balance load when transmitting subflow links.

3.1 Topology Collection and Traffic Monitoring

The topology discovery module of the controller is used to obtain the location of the terminal host in the global network. The controller sends LLDP packets to all switches by sending a Packet_out message. After receiving the packets, the switch sends LLDP packets to all of its ports. If an OpenFlow switch receives these outgoing LLDP packets, it will send the link information between the two switches to the controller through the Packet in message. After the controller collects the link information of its own management area, it can build the network topology based on these information. The traffic statistics module is used to periodically collect the number of packets and bytes forwarded by the OpenFlow switch port. It can also record the forwarding rate of each stream on each port, get the total capacity of the port and the available bandwidth.

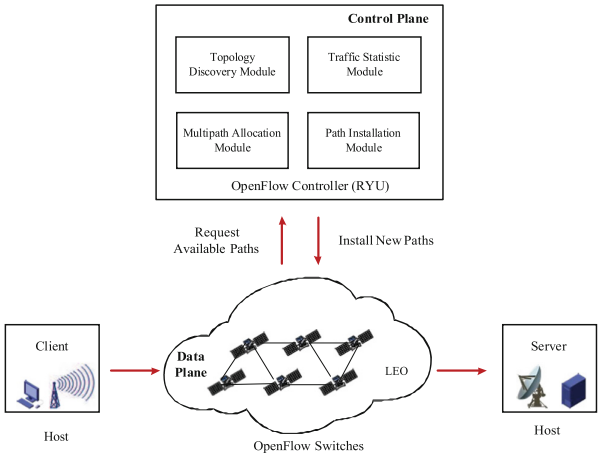

Fig. 1. Architecture of our proposed mechanism.

3.2 Multipath Allocation Module

Subflow path selection scheme plays an important role in satellite transmission performance. Here we analyze the routing strategy of load balancing between links based on k shortest paths. We consider the topology shown in Fig. 2, where the hosts at both ends communicate through satellite nodes such as switches s1 and s2.

There are multiple paths between two user nodes in the network. If there are too many optional paths, choosing the best path can be too inefficient. We use the Depth First Search path find algorithm to get all the paths. The basic principle is as follows: Record the relationship of connection between all nodes in the adjacency matrix. Before backtracking, find the deepest node in the graph first, and then find other possible nodes through the stack. Finally, return a list of all possible paths. Since the DFS algorithm returns an unweighted path list, we need to measure the cost of the path to allocate the subflows to achieve load balancing. We set a MAX_PATHS to indicate the number of the selected the k shortest paths, which is also the number of paths of the

subflow. We calculate the link cost by polling the corresponding switch port using the OSPF mechanism. We use the following cost formula. This formula uses the above-mentioned OSPF mechanism to obtain the link cost. For a path i, bw represents the weight of the bucket, $0 \leq \text{bw(p)} < 10$. The pw represents the path cost and n represents the total number of available paths. When it comes to path cost, we ideally want to find the optimal path first, and then choose the suboptimal one. Correspondingly, the priority of the bucket in the OpenFlow Group table is sorted according to the weight of the bucket. Through this formula, we can predict that the lower the path cost is, the higher the bucket weight will be. We divide the subflow according to the weight of the bucket to achieve the load balancing effect of the path.

$$bw(i) = \left(1 - \frac{pw(i)}{\sum_{j<n}^{j=0} pw(j)}\right) \times 10 \tag{1}$$

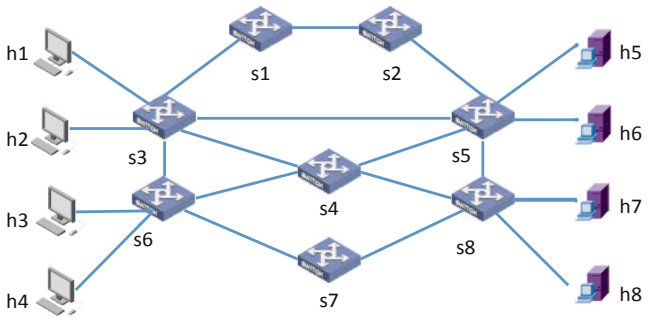

Fig. 2. Topology of switches and hosts.

Algorithm 1 K Shortest Paths Based Load Balancing Algorithm (mptcp-klb)

Input: k, bandwidth, Reference bandwidth
Output: a weight for each path t∈k

J is the total paths from S to T
For each j in {1,2,..., J}
    For each i in j nodes
        cost[j]= Reference bandwidth/ bandwidth(i, i+1)
    End For
End For
Get the k shorted paths from Sorted(J, key=lambda x: cost[x])
For each t in k
    return Weights[t]=1-cost[t]/ $\sum_{k}^{1} cost[t]$
End For

3.3 Path Installation Module

The forwarding rule is generated by the controller according to the results of the multipath allocation module. When the user sends a request to the switch, if the switch sends a request message to the controller every time, enabling the controller to make a routing decision and then returns to the switch, the network performance will get lower. We consider letting the switch itself make a decision that installs OpenFlow rules on the switch by itself, so that the switch will directly forward network traffic [15]. At the same time, we must also consider the continual change of network traffic and topology. As a result, the flow table needs to be dynamically updated to meet the load balancing of the path. When a new flow arrives or the network topology changes, select the best path for the subflow and install the path to the corresponding switch through the previously described module. At the same time, the control network collects global information and updates the flow table configuration in time.

4 Performance Evaluation

In order to evaluate the path load balancing strategy based on MPTCP, we simulate the network topology and communicate with the controller in the Mininet [16] platform, the topology is shown in Fig. 2. Mininet creates virtual networks quickly, running hosts, switches and network links on a single computer. All switches in Mininet are software-based OpenFlow switches. The version of OpenFlow is 1.3. All modules in the controller are implemented on Ryu [17] and all of the softwares listed above run in the VMware virtual machine. Our routing algorithm runs on the Ryu controller. We also implemented k-shortest-paths (mptcp-ksp) [18] and single-path routing (single-tcp) algorithms in Ryu, so that it is easy to compare them with the performance of our algorithm.

We set the link delay between two satellites at 20 ms and bandwidth at 25 Mbps [19]. The link delay and bandwidth between the terminal host and the satellite are respectively set to 5 ms and 15 Mbps. The connection status of the network device is shown in Fig. 2. We assume that the weight of each link is 1. h1 and h7 are selected as observation objects. Firstly, the k-shortest-path algorithm is used to select the three optimal paths, and then the sub-flows are distributed in proportion through the path bw to achieve load balancing. h1 sends TCP packets to h7. For all of the path sets from h1 to h7, we select three paths for transmission. Considering that the paths should be shortest and disjoint, we select three paths, path 1, path 2, and path 3. The bw of them is proportionally allocated as 7:7:6. As is shown in Fig. 3, we test the total number of packets transmitted on three paths with different number of parallel clients. It can be seen that as the number of clients increases, the number of packets forwarded on the three paths all increases accordingly. The proportion of the number of packets transmitted in each path is close to the proportion of bw.

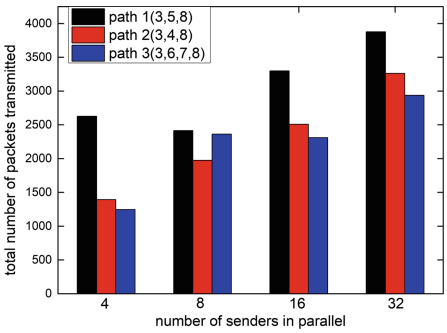

Fig. 3. Total number of packets transmitted with different sender number

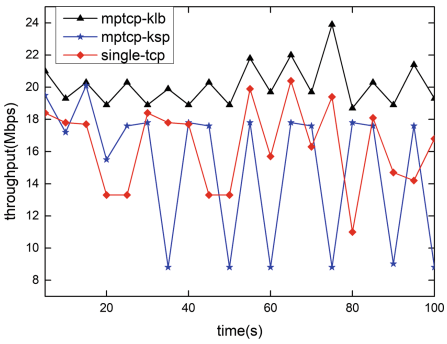

Fig. 4. Sender h1's total throughput

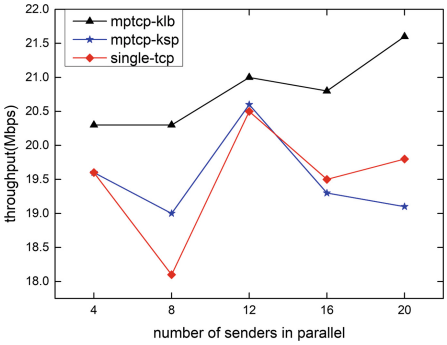

Fig. 5. Sender h1's total throughput with different sender number

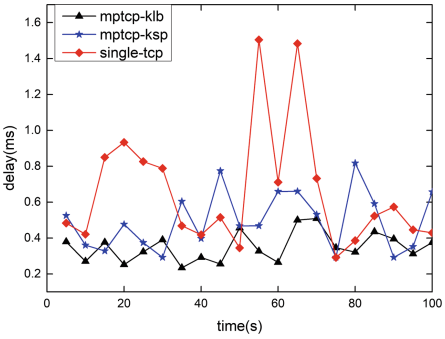

Fig. 6. Delay performance of three algorithms

In Fig. 4, the total throughput of the sender h1 is plotted in the context of using three different algorithms. Because mptcp-ksp and single-tcp do not take the path load into account, the throughput of the system has a certain fluctuation. Mptcp-ksp focus on path selection without congestion control, so it may be more unstable than single-tcp in large number of packets. The overall throughput of mptcp-lb is compared with the other two algorithms. Figure 5 shows the throughput performance of the three algorithms with different number of parallel clients. As we can see, the throughput of the mptcp-ksp and single-tcp schemes rise rapidly when there are 12 parallel clients, and then remain relatively low. This is because the distribution of traffic on the path is not considered, resulting in congestion of the path. Since mptcp-lb allocates subflows according to the load capacity of the path, the throughput of sender h1 has shown an upward trend.

Figure 6 shows the delay performance of the three algorithms. Single-tcp is a single path transmission. Affected by link congestion, its delay performance fluctuates greatly. Relatively speaking, mptcp-lb and mptcp-ksp utilize multipath transmission, so the delay performance of packet transmission can be improved effectively. mptcp-lb has lower latency than mptcp-ksp, which proved that mptcp-lb can realize the rationality of subflow routing arrangement under load balancing.

5 Conclusion

In this paper, we proposed an MPTCP-based load balancing strategy to solve the problem of transmitting data flow in satellite networks. Applying the SDN architecture to the satellite network can obtain the real-time global network topology and state, which improves the utilization and reliability of the path. This strategy can effectively balance the link bandwidth by considering the path cost. We, and proved that the proposed scheme can effectively divide and forward subflow in comparison with other schemes. The simulation results verify that the total system throughput and delay are improved.

Acknowledgements. This research is partially supported by research grants from Science and Technology Project of Jilin province (20180414024GH), and the project in the Education Department of Jilin Province (JJKH20170630KJ).

References

1. SpaceX. https://en.wikipedia.org/wiki/SpaceX
2. SES S.A. https://en.wikipedia.org/wiki/SES_S.A
3. Ford, A., Raiciu, C., Handley, M., et al.: TCP extensions for multipath operation with multiple addresses (2013)
4. Du, P., Nazari, S., Mena, J., et al.: Multipath TCP in SDN-enabled LEO satellite networks. In: 2016 IEEE Military Communications Conference, MILCOM 2016, pp. 354–359. IEEE (2016)
5. Hwang, J., Yoo, J.: Packet scheduling for multipath TCP. In: 2015 Seventh International Conference on IEEE Ubiquitous and Future Networks (ICUFN), pp. 177–179. IEEE (2015). https://doi.org/10.1109/ICUFN.2015.7182529
6. Singh, A., Xiang, M., Konsgen, A., et al.: Enhancing fairness and congestion control in multipath TCP. In: 2013 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC), pp. 1–8. IEEE (2013). https://doi.org/10.1109/WMNC.2013.6549059
7. Krupakaran, K., Sridharan, A.P., Venkatesan, S.M.: Optimized multipath TCP subflows using traceflow, April 2015
8. Baidya, S.H., Prakash, R.: Improving the performance of multipath TCP over heterogeneous paths using slow path adaptation. In: 2014 IEEE International Conference on Communications (ICC), pp. 3222–3227. IEEE (2014). https://doi.org/10.1109/ICC.2014.6883817
9. Raiciu, C., Barre, S., Pluntke, C., et al.: Improving datacenter performance and robustness with multipath TCP. ACM SIGCOMM Comput. Commun. Rev. **41**(4), 266–277 (2011)

10. Zannettou, S., Sirivianos, M., Papadopoulos, F.: Exploiting path diversity in datacenters using MPTCP-aware SDN. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 539–546. IEEE (2016)
11. Rossi, T., De Sanctis, M., Cianca, E., et al.: Future space-based communications infrastructures based on high throughput satellites and software defined networking. In: 2015 IEEE International Symposium on Systems Engineering (ISSE), pp. 332–337. IEEE (2015)
12. Bao, J., Zhao, B., Yu, W., et al.: OpenSAN: a software-defined satellite network architecture. ACM SIGCOMM Comput. Commun. Rev. **44**(4), 347–348 (2014). https://doi.org/10.1145/2619239.2631454
13. Li, T., Zhou, H., Luo, H., et al.: SERvICE: a software defined framework for integrated space-terrestrial satellite communication. IEEE Trans. Mob. Comput. **17**(3), 703–716 (2018)
14. Ferrs, R., Koumaras, H., Sallent, O., et al.: SDN/NFV-enabled satellite communications networks: opportunities, scenarios and challenges. Phys. Commun. **18**, 95–112 (2016)
15. Giotis, K., Argyropoulos, C., Androulidakis, G., et al.: Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Comput. Netw. **62**, 122–136 (2014)
16. Mininet. http://mininet.org/. Accessed 4 Oct 2017
17. Ryu controller. http://osrg.github.io/ryu/. Accessed 4 Oct 2017
18. Zannettou, S., Sirivianos, M., Papadopoulos, F.: Exploiting path diversity in datacenters using MPTCP-aware SDN. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 539–546. IEEE (2016)
19. Taleb, T., Mashimo, D., Jamalipour, A., et al.: Explicit load balancing technique for NGEO satellite IP networks with on-board processing capabilities. IEEE/ACM Trans. Netw. **17**(1), 281–293 (2009)