Received June 20, 2017, accepted July 7, 2017, date of publication July 12, 2017, date of current version August 22, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2726114

SAT-FLOW: Multi-Strategy Flow Table Management for Software Defined Satellite Networks

TAIXIN LI[1,2], (Student Member, IEEE), HUACHUN ZHOU[1,2], HONGBIN LUO[1,2], (Member, IEEE), ILSUN YOU[3], (Senior Member, IEEE), AND QI XU[1,2]
[1]School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
[2]National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing 100044, China
[3]Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea

Corresponding author: Taixin Li (14111040@bjtu.edu.cn)

This work was supported in part by Fundamental Research Funds for the Central Universities under Grant 2016YJS026, in part by the National Basic Research Program of China (973 program) under Grant 2013CB329101, in part by the NSAF of China under Grant U1530118, in part by the National High Technology of China (863 program) under Grant 2015AA015702, and in part by the NSFC under Grant 61232017 and 61422101.

ABSTRACT Software-defined satellite network (SDSN) is a novel framework, which brings software-defined network technologies in the satellite networks. It has great potential to achieve effective and flexible management in the satellite networks. There are two burning issues to be solved for the flow table management in SDSN. First, the ternary content addressable memory (TCAM) space is limited on satellites and the flow table size should be reduced. Second, the frequent handovers will lead to an increase in the flow table size in SDSN. Due to the limited flow table space, a lot of flows will be dropped if the flow table is full during the handover. To address these issues, we first give a description of our focused flow table management problems. Then, we propose *SAT-FLOW*, a multi-strategy flow table management method for SDSN. *SAT-FLOW* considers three key points, limited TCAM space, classified traffic, and handover. *SAT-FLOW* contains two heuristic algorithms, named dynamic classified timeout (*DCT*) algorithm and timeout strategy-based mobility management (*TSMM*) algorithm. *DCT* aims to reduce the flow table size and *TSMM* aims to reduce the drop flows during the handover. We implement *SAT-FLOW* and conduct contrast experiments. The experimental results verify the good performance in terms of transmission quality, *idle_timeout* values distribution, a 15.27%–24.34% decrease in flow table size, an 8.2%–10.4% decrease in drop-flow rate, and a 4.92%–5.7% decrease in table misses for the high priority traffic during the handover.

INDEX TERMS Software defined satellite network, timeout, flow table management, *SAT-FLOW*.

I. INTRODUCTION

In the past few decades, satellite communication technologies have developed rapidly. Satellite networks can provide global efficient broadcast or multicast services anywhere and anytime due to the advantages of global coverage, mobility, and scalability. Now, the satellite communication becomes one of the most important parts of modern communication systems. However, with the development of satellite networks, current and emerging satellite applications become increasingly complex. It is difficult to achieve effective and flexible management due to coarse-grained controls and time-consuming network configuration in the traditional satellite communication networks.

Software Defined Networking (SDN) [1] paradigm and OpenFlow [2] have attracted a lot of attention from both the industry and research institutes. SDN has the characteristic of separating the data plane from the control plane. Software Defined Satellite Networks (SDSN) is an emerging framework that implements SDN technologies in the satellite networks. SDSN is designed to solve the problems, for example, ineffective and inflexible management in the traditional satellite networks. In recent years, there have been some researches, for example [3]–[5], and [6] respectively, focusing on designing SDSN. However, the flow table management for SDSN remains to be studied.

In order to provide high-speed parallel lookup on wildcard patterns, SDN switches usually use Ternary Content Addressable Memory (TCAM) to store their forwarding rules given by a controller [1]. However, it is reported that a commodity switch can store about only 1500 entries [7] because of the high cost and energy consumption of TCAM. Therefore, efficient flow table management is very important to improve the performance of SDN. There have already been researches about flow table management, for example, [8] and [9] aggregate the flow entries by using wildcard rule, thus reducing the flow table size. However, flow table management in SDSN is different from that in terrestrial networks because of the following characteristics of satellite networks.

- The TCAM limit is severer in SDSN. The energy on satellites is limited and the cost of satellite hardware is much higher than that of the terrestrial switch hardware. Therefore, the number of TCAM entries that a satellite can hold is smaller than terrestrial switches. More consideration should be taken on flow table space limit when making flow table management strategies.
- Satellite traffic can be divided into different classes and has different requests for link resources and QoS. Different priorities should be given to different classes of traffic when making flow table management strategies. Thus, the transmission quality of high priority traffic should be guaranteed when the TCAM space is limited.
- The satellite handover occurs frequently, which is different from that in ground wired networks. Rule replacement will occur very frequently in Software Defined Satellites. However, the existing flow entries will still be maintained for a while in the flow table after handover. These unexpired entries occupy the flow table space and are useless because they will not be matched any more. The subsequent flows may be dropped if the TCAM space is limited.

There are two timeout mechanisms in OpenFlow, namely, *idle_timeout* and *hard_timeout* to release the overload flow table. In this paper, we are motivated to focus on flow table management in SDSN by adjusting *idle_timeout* dynamically. We aim to control the flow table size, especially during the handover process. In this way, the performance of transmission in SDSN will be improved. We propose *SAT-FLOW*, a multi-strategy flow table management method for SDSN. *SAT-FLOW* is composed of two heuristic algorithms, named Dynamic Classified Timeout (*DCT*) algorithm and Timeout Strategy-based Mobility Management (*TSMM*) algorithm. *DCT* aims to work out a basic *idle_timeout* value for the table-miss flow taking limited TCAM space and classified traffic into consideration. *TSMM* aims to work out a further *idle_timeout* value based on the result of *DCT* considering link handover in satellite networks. Then, we describe the implementation of *SAT-FLOW* in our prototype, which is supported by a state-level program and introduced in [10]. The prototype adopts Delay Tolerant Network (DTN) [11] for data transmission and OpenFlow for network control. The experimental results show that *DCT* and *TSMM* have a better performance in terms of flow table size, drop-flow rate, table-misses, transmission quality and *idle_timeout* values distribution than other strategies.

To sum up, our main technical contributions in this work are as follows.

- We propose *SAT-FLOW*, which contains two heuristic algorithms, *DCT* and *TSMM*, and considers three key points, limited TCAM space, classified satellite traffic, and satellite link handover. To the best of our knowledge, we are the first ones to study the flow table management problem in SDSN in way of adjusting the *idle_timeout* dynamically.
- We implement the two heuristic algorithms of *SAT-FLOW* in the OpenFlow-based and DTN-based prototype. We deploy *DCT* and *TSMM* in the prototype by installing python modules in the POX controller-based [12] satellite control node and shell scripts in the Open vSwitch-based [13] satellite forwarding nodes.
- We conduct experiments in our prototype. The experimental results show that *SAT-FLOW* can achieve good performance in terms of transmission quality, *idle_timeout* values distribution, a 15.27%-24.34% decrease in flow table size, an 8.2%-10.4% decrease in drop-flow rate, and a 4.92%-5.7% decrease in table-misses for the high priority traffic during the handover.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents a description of our focused flow table management problem. Section IV gives the description and implementation of the *SAT-FLOW* strategy. Section V presents the tests of the proposed heuristic algorithms and the experimental results. In Section VI, we further discuss the study. Finally, Section VII sums up the paper and presents our future work.

II. RELATED WORK

In this paper, we focus on flow table management in SDSN by reducing the flow table size, especially when a handover occurs. In recent years, most of the SDSN related researches are about architecture designing, for example [3], [4], and [5]. Yang *et al.* [6] propose a seamless handover mechanism based on SDSN and conduct physical layer simulation, which shows significant improvement over the existing hard handover and hybrid handover mechanisms. However, [6] mainly focus on handover mechanism while our focus is on flow table size control. Even so, [6] still inspires us a lot. Because handover is one of the most important points that we consider when designing *SAT-FLOW*. Flow table management is a hot topic in SDN and still needs to be explored in depth in SDSN. There have been many researches about flow table management in ground scene. These researches can be mainly divided into three directions.

The first one is designing local policies to process some requests inside switches, thus reducing the number of requests sent to the controller [7], [14]. DIFANE [14] keeps all traffic in the data plane by selectively directing packets through authority switches that store the necessary rules. The controller is responsible for partitioning rules over authority switches. Devoflow [7] separates the management of small flows from the giant flow to handle most ''mice'' (brief and individual) flows in switches and ''elephant'' (long-lived and high-throughput) flows in controller. However, these designs are not suitable for satellite networks. For example, [14] leaves heavy load on forwarding nodes, which are limited in processing and storage resources. In [7], the controller may lose control of some flow that may be sent by high class tenants.

The second one is flow table aggregation by using the wildcard rule for reducing flow table size in a switch [8], [9]. The designing idea in common is that these approaches restructure the OpenFlow wildcard matching rules to aggregate the flows that have some common prefixes. However, it is required to modify the original switch design, raising concerns about compatibility with SDN standards and increased deployment costs.

The third one is assigning proper static/dynamic timeouts to rules for flow eviction based on OpenFlow rule timeout mechanism. This direction is the most relevant one to our work. There are many researches about the first two directions, while the problem of assigning dynamic and proper timeouts still needs to be explored in depth, especially in SDSN. [15] is the first article that explores the impact of the timeout length on performance in SDN. Miss rate and table occupancy are tested when timeout values changed. And Zarek *et al.* [15] proposes a hybrid flow table management method that combines timeouts with explicit controller eviction messages. However, [15] uses a static timeout and leaves dynamically tuned timeouts for future work. Kim *et al.* [16] propose an approach to predict the inter-arrival time of packets in a flow by collecting various traffic network parameters. Then, the controller adjusts the timeout value based on the predicted information to reserve flow table spaces for newly arrived flows in advance. Zhu *et al.* [17] predict the suitable timeout by recording the last expire time and adjust the max timeout to avoid the overflow of the flow table. SmartTime [18] combines an adaptive timeout heuristic to compute idle timeouts with proactive eviction of flow rules and implement the flow management strategy in an OpenFlow controller.

However, these approaches are not suitable for satellite networks because the handover and classified traffic are not considered. In our work, we will consider the limited TCAM space, handover, and classified traffic, all of which are characteristics in satellite networks.

III. PROBLEM DESCRIPTION

For the sake of clear description, we make the definitions as shown in Table 1.

Fig. 1 illustrates the architecture and the transmission process in SDSN. The controller is located in Geostationary Earth Orbit (GEO) satellites and the switches are deployed in Medium Earth Orbit (MEO)/ Low Earth Orbit (LEO) satellites in SDSN. When a packet arrives at the forwarding satellite, the satellite performs lookup in its internal flow tables. If the lookup hits a table entry other than table-miss, the satellite will forward the packet to the next one in a conventional way. Otherwise, the packet is supposed to belong

TABLE 1. Definition of the parameters.

$TCAM_{limit}$	The TCAM space limit
$T(i)_{flow(k)}$	The *idle_timeout* of flow k
$T(h)_{flow(k)}$	The *hard_timeout* of flow k
T_{waste}	The time duration that no flows match the existing flow entry
$T_{flow(k)}$	The flow duration of flow k
N_{entry}	The number of flow table entries
$T_{link(n)}$	The connection duration time of link n
$t^{handover}_{link(n)}$	The time that link n handover occurs
$t^{arrive}_{flow(k)}$	The arriving time of flow k
$T(i)_{init}$	The initial *idle_timeout*
$N_{flow(k)}$	The number of *packet_in* messages of flow k
f_1, f_2, f_3	The decrease factor for the traffic of tos_1, tos_2, and tos_3
$T(i)^{DCT}_{flow(k)}$	The *idle_timeout* of flow k calculated by *DCT* algorithm
$T(i)^{TSMM}_{flow(k)}$	The *idle_timeout* of flow k calculated by *TSMM* algorithm

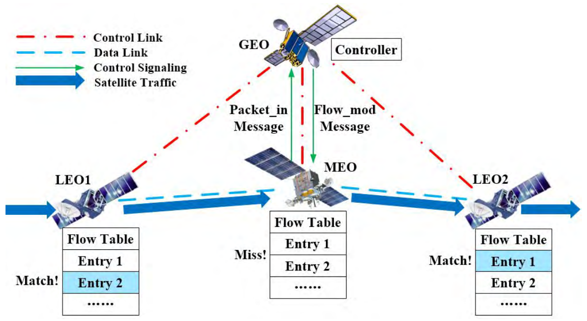

FIGURE 1. Illustration of software defined satellite networks.

to a new flow. In such case, the forwarding satellite requests the control satellite for instructions by sending a *packet_in* message encapsulating the arrived packet. The control satellite determines the respective flow rule and installs it into the flow table located in the forwarding satellite. After that, all packets within the flow are correctly forwarded to their destination without requesting the controller. In Fig. 1, the satellite traffic arrives at LEO1 satellite and matches the entry in the flow table. Then the traffic is forwarded to the next satellite along the path. The traffic arrives at MEO satellite while matches no local entries. Therefore, MEO satellite sends out *packet_in* message to the controller in GEO satellite for forwarding rules. After receiving *flow_mod* message that contains instructions from the controller, MEO satellite forwards the traffic to LEO2. Then, LEO2 satellite forwards the traffic to the next node according to the matched entry.

There are two timeout mechanisms in OpenFlow specification, namely, *hard_timeout* and *idle_timeout*. The flow entry is evicted if the *hard_timeout* is up no matter whether the flow that matches the entry is ongoing or not. While *idle_timeout* is the time period from when there are no active flows that match the entry to when the *idle_timeout* is up. Both of *hard_timeout* and *idle_timeout* can prevent too much flow entries installed in the flow table. While using *hard_timeout* may lead to extra *packet_in* messages if the *hard_timeout* value is shorter than the flow transmission time. Now we will analyse the differences between the *hard_timeout* and the *idle_timeout*.

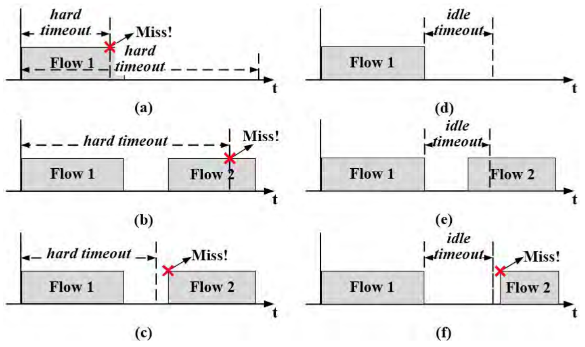

FIGURE 2. Comparison of hard_timeout and idle_timeout.

Fig. 2 illustrates the comparison of the *hard_timeout* and the *idle_timeout*. Fig. 2 (a) and (d) show the situation that there are no repeated flows. If $T\ (h)_{flow(1)} < T_{flow(1)}$, table-miss will be triggered and if $T\ (h)_{flow(1)} > T_{flow(1)}$, there will be TCAM space waste because useless flow entry occupies the flow table space, $T_{waste} = T\ (h)_{flow(1)} - T_{flow(1)}$. Only when $T\ (h)_{flow(1)} = T_{flow(1)}$, neither table-miss nor TCAM space waste will be triggered, while it is very hard to predict flow duration exactly. *idle_timeout* will lead to TCAM space waste in this case, $T_{waste} = T\ (i)_{flow(1)}$. Fig. 2 (b), (c), (e), and (f) show the situation that there are repeated flows. The flow 1 and the flow 2 have the same tuples. In Fig. 2 (b), $t^{arrive}_{flow(2)} \leq T\ (h)_{flow(1)} < t^{arrive}_{flow(2)} + T_{flow(2)}$, table-miss is triggered and $T_{waste} = t^{arrive}_{flow(2)} - T_{flow(1)}$. While in Fig. 2 (e), $t^{arrive}_{flow(2)} - T_{flow(1)} \leq T\ (i)_{flow(1)} < T_{flow(2)} + t^{arrive}_{flow(2)} - T_{flow(1)}$, no table-miss is triggered and $T_{waste} = t^{arrive}_{flow(2)} - T_{flow(1)}$. In Fig. 2 (c), $T_{flow(1)} \leq T\ (h)_{flow(1)} < t^{arrive}_{flow(2)}$, table-miss is triggered and $T_{waste} = T\ (h)_{flow(1)} - T_{flow(1)}$. While in Fig. 2 (f), $T\ (i)_{flow(1)} < t^{arrive}_{flow(2)} - T_{flow(1)}$, table-miss is triggered and $T_{waste} = T\ (i)_{flow(1)}$.

We can conclude that using *hard_timeout* triggers more *packet_in* messages than using *idle_timeout*. And both of *hard_timeout* and *idle_timeout* can lead to TCAM space waste. We must predict the flow duration time if we want to reduce the space waste in the case of using *hard_timeout*. In addition, hard_timeout may cause the situation that the flow entry is evicted while the flow is still active. Therefore, we decide to manage the flow table by allocating proper *idle_timeout*.

In this paper, we have three goals when designing *SAT-FLOW*. Firstly, we aim to work out proper *idle_timeout* value considering TCAM space limit. Secondly, classified traffic should be considered when there is not enough flow table space. Thirdly, we aim to eliminate the effect of link handover on the TCAM space. The analysis of the three goals are as follows.

As we know, OpenFlow rules are more complex than forwarding rules in traditional IP routers. They support fine-grained control of the traffic and more flexible matchings. OpenFlow enabled switches are based on wildcard rules that fit naturally with TCAM. However, using TCAM is expensive and power-hungry. For example, TCAM is 400 times more expensive [19] and 100 times more power-consuming [20] per Mbit than RAM-based storage. Therefore, the limited TCAM space should be considered in OpenFlow enabled networks, especially in SDSN, where resources, such as computing and storage, are limited and cost of hardware is extremely expensive.

The resources in the satellite networks are limited and we cannot allocate enough resources to every tenants. It is necessary to classify the tenants and traffic according to their properties and requirements. Here, we define three Types of Service (ToS), namely $ToS = \{tos_1, tos_2, tos_3\}$. tos_1 identifies the data that needs the highest priority, which means that tos_1 traffic should have the smallest table-miss probability. The tos_1 traffic may be sent by the users in government, military, or the tenants that have paid much to the operators. tos_2 identifies the data that need real-time transmission. The instruction data, or signal data, or point-of-sale data may be marked as tos_2. tos_3 identifies the traffic that is delay-tolerant. The file transmission data, or database querying data can be marked as tos_3. The priority of the three classes of traffic is $tos_1 > tos_2 > tos_3$ when making flow table management strategies.

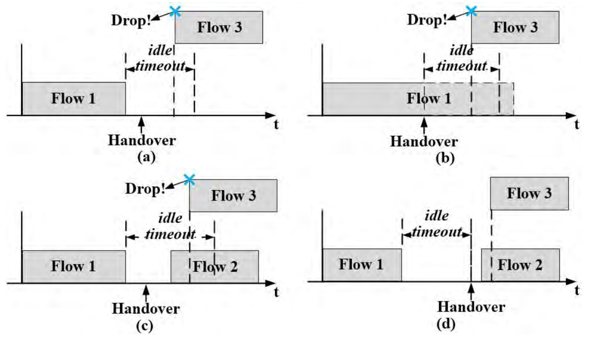

FIGURE 3. Drop-flow in handover scenario.

Finally, we will analyze why handover is an important point that we should consider. Fig. 3 illustrates the drop-flow in the handover scenario. The flow 1 and the flow 2 have the same tuple and the flow 2 is the repeated flow of the flow 1. While the flow 3 is a new flow. When new flows arrive while there is no TCAM space left, the flow will be dropped, namely, drop-flow. In Fig.3 (a) and (b), there are no repeated flows, In Fig. 3 (a), $T_{flow(1)} \leq t^{handover}_{link(n)} < t^{arrive}_{flow(3)}$, $t^{handover}_{link(n)} < t^{arrive}_{flow(3)} < T_{flow(1)} + T\ (i)_{flow(1)}$. If $N_{entry} = TCAM_{limit}$, the flow 3 will be dropped. In Fig. 3 (b), $T_{flow(1)} > t^{handover}_{link(n)}$. A handover occurs when the flow 1 is still active and then no flow 1 packets arrive. If $t^{handover}_{link(n)} < t^{arrive}_{flow(3)} < t^{handover}_{link(n)} + T\ (i)_{flow(1)}$ and $N_{entry} = TCAM_{limit}$, the flow 3 will be dropped. In Fig. 3 (c) and (d), there are repeated flows and

$T_{flow(1)} \leq t^{handover}_{link(n)} \leq t^{arrive}_{flow(2)}$. If $t^{arrive}_{flow(2)} \leq t^{arrive}_{flow(3)} \leq t^{arrive}_{flow(2)} + T(i)_{flow(2)}$, $t^{arrive}_{flow(2)} \leq t^{arrive}_{flow(3)} \leq T_{flow(1)} + T(i)_{flow(1)}$, and $N_{entry} = TCAM_{limit}$, the flow 3 will be dropped. In Fig. 3 (d), if we adjust $T(i)_{flow(1)} \leq t^{handover}_{link(n)} - T_{flow(1)}$, there may be space in the flow table for the flow 3. We can conclude that a handover process may cause an increase in flow table size and a proper *idle_timeout* can reduce drop-flows during handover.

IV. ALGORITHM AND IMPLEMENTATION

Considering the key points in Section III, we propose *SAT-FLOW*. In this section, we will introduce the design and implementation of *SAT-FLOW*.

A. ALGORITHM DESCRIPTION

The descriptions of *DCT* algorithm and *TSMM* algorithm are as follows.

1) DYNAMIC CLASSIFICATION TIMEOUT (DCT) ALGORITHM

DCT takes two key points into consideration, limited TCAM spaces and classified traffic.

a: LIMITED TCAM SPACES

As discussed in Section III, TCAM-based flow table space is limited in OpenFlow enabled networks, especially in SDSN. Therefore, TCAM space is one of the most important parameters in the algorithm. We use $TCAM_{limit}$ as the threshold. If N_{entry} reaches 80% of $TCAM_{limit}$, it means that most of the flow table space is occupied and the increment speed of *idle_timeout* value should be slowed down. If N_{entry} reaches 95% of $TCAM_{limit}$, it means that the flow table is almost used up and the subsequent flows may be dropped due to no more spaces in the flow table. The *idle_timeout* should be reduced to accelerate the expiration of the flow entries.

b: CLASSIFIED TRAFFIC

As discussed in Section III, when it is necessary to reduce the entries, the *idle_timeout* of the flow entries with different priorities is reduced in different proportion. We define f_1, f_2, and f_3 as the decrease factor for the traffic of tos_1, tos_2, and tos_3, $f_1 > f_2 > f_3$.

The process of *DCT* algorithm can be divided to three stages.

Stage 1: Small start ($N_{entry} < TCAM_{limit} \times 80\%$). We set the initial *idle_timeout* a small value (for example, 1s) rather than a big one to probe the upper limit of the flow table space. The *idle_timeout* value increases exponentially in order to probe the upper limit fast.

Stage 2: Limit voidance ($TCAM_{limit} \times 80\% \leq N_{entry} \leq TCAM_{limit} \times 95\%$). The *idle_timeout* value increases additively to avoid reaching the upper limit too fast because the remaining space is not too much.

Stage 3: Fast decrease ($N_{entry} > TCAM_{limit} \times 95\%$). The *idle_timeout* value is reduced immediately according to the decrease factors. We aim to reserve space for the following flows in order to avoid drop-flow.

Algorithm 1 *DCT* Algorithm

```
Input:
    decrease factors $f_1, f_2$, and $f_3$;
    initial idle_timeout $T(i)_{init}$;
Output:
    Appropriate $T(i)^{DCT}_{flow(k)}$ for flow $k$;
1:  for packet_in message arrives do
2:    if packet_in message of flow $k$ is new then
3:      $T(i)_{flow(k)} = T(i)_{init}$;
4:      $N_{flow(k)} = 1$;
5:    else if packet_in message of flow $k$ is not new then
6:      if $N_{entry} < TCAM_{limit} \times 80\%$ then
7:        $T(i)_{flow(k)} = T(i)_{init} \times 2^{N_{flow(k)}}$;
8:      else if $TCAM_{limit} \times 80\% \leq N_{entry} \leq TCAM_{limit} \times 95\%$ then
9:        $T(i)_{flow(k)} = T(i)_{flow(k-1)} + 1$;
10:     else if $N_{entry} > TCAM_{limit} \times 95\%$ then
11:       if ToS of flow $k$ is $tos_1$ then
12:         $T(i)_{flow(k)} = T(i)_{flow(k)} \times f_1$;
13:       else if ToS of flow $k$ is $tos_2$ then
14:         $T(i)_{flow(k)} = T(i)_{flow(k)} \times f_2$;
15:       else if ToS of flow $k$ is $tos_3$ then
16:         $T(i)_{flow(k)} = T(i)_{flow(k)} \times f_3$;
17:       end if
18:     end if
19:     $N_{flow(k)} = N_{flow(k)} + 1$;
20:   end if
21:   $T(i)^{DCT}_{flow(k)} = T(i)_{flow(k)}$
22: end for
23: return $T(i)^{DCT}_{flow(k)}$;
```

The basic procedure of *DCT* algorithm is shown in Algorithm 1.

It accepts the decrease factors f_1, f_2, and f_3, and the initial *idle_timeout* $T(i)_{init}$ as input. When the *packet_in* message arrives at the controller, we judge if the flow information that is encapsulated in the *packet_in* message is new. If so, $T(i)_{flow(k)}$ is set as $T(i)_{init}$ and we begin to count $N_{flow(k)}$. If not, we will judge if flow table space is enough. If it is in **Stage 1**, $T(i)_{flow(k)} = T(i)_{init} \times 2^{N_{flow(k)}}$. If we judge that it is in **Stage 2**, $T(i)_{flow(k)} = T(i)_{flow(k-1)} + 1$. If it is in **Stage 3**, $T(i)_{flow(k)}$ will be decreased by f_1, f_2, and f_3. Then, we will get $T(i)^{DCT}_{flow(k)}$.

2) TIMEOUT STRATEGY-BASED MOBILITY MANAGEMENT (TSMM) ALGORITHM

TSMM takes one key point into consideration, namely satellite link handover.

a: SATELLITE LINK HANDOVER

As discussed in Section III, the handover in satellite networks may cause a sudden increase in flow table size. If the TCAM-based flow table space is limited, some flow may be dropped. Therefore, the *idle_timeout* value should not be bigger than the remaining time of the connection.

Algorithm 2 *TSMM* Algorithm

```
Input:
    result of DCT algorithm: $T(i)^{DCT}_{flow(k)}$;
    connection information of link n: $T_{link(n)}$, $t^{handover}_{link(n)}$;
Output:
    Appropriate $T(i)^{TSMM}_{flow(k)}$ for flow $k$;
 1: for packet_in message arrives do
 2:     Record the arriving time $t^{arrive}_{flow(k)}$ of flow $k$;
 3:     if flow $k$ is forwarded via link $n$ then
 4:         if $t^{handover}_{link(n)} + T_{link(n)} - t^{arrive}_{flow(k)} > T(i)^{DCT}_{flow(k)}$ then
 5:             $T(i)^{TSMM}_{flow(k)} = T(i)^{DCT}_{flow(k)}$;
 6:         else if $t^{handover}_{link(n)} + T_{link(n)} - t^{arrive}_{flow(k)} \leq T(i)^{DCT}_{flow(k)}$
            then
 7:             $T(i)^{TSMM}_{flow(k)} = t^{handover}_{link(n)} + T_{link(n)} - t^{arrive}_{flow(k)}$;
 8:         end if
 9:     end if
10: end for
11: return $T(i)^{TSMM}_{flow(k)}$;
```

The basic procedure of *TSMM* algorithm is shown in Algorithm 2.

It accepts the result of *DCT* algorithm, $T(i)^{DCT}_{flow(k)}$ and the connection information of the links, $T_{link(n)}$ and $t^{handover}_{link(n)}$. When the *packet_in* message arrives at the controller, we record its arriving time $t^{arrive}_{flow(k)}$. Because satellite networks are scheduled, we can get the information about the start time and terminal time of the satellite links. We also know on which link this flow is forwarded from the connection information. Then, we can get the remaining time of this connection by $t^{handover}_{link(n)} + T_{link(n)} - t^{arrive}_{flow(k)}$. We compare the remaining time and $T(i)^{DCT}_{flow(k)}$ and choose the smaller one as $T(i)^{TSMM}_{flow(k)}$.

The *SAT-FLOW* method is composed of *DCT* and *TSMM*. In *SAT-FLOW*, *TSMM* comes after *DCT* and together with the functions of the two algorithms can we achieve the goal of flow table management in SDSN. The time complexity of the two algorithms is $O(n)$.

B. IMPLEMENTATION

The tests of *SAT-FLOW* are conducted in the SDSN prototype, which is based on Kernel-based Virtual Machine (KVM) and OpenStack. The prototype is implemented in five high-performance servers (DELL PowerEdge R720 rack-mount servers). It is necessary to introduce the implementation of SDSN and *SAT-FLOW*.

We adopt the three-layer constellation named *Tr* [21]. We deploy *Tr* in Satellite Tool Kit (STK) [22] and the parameters listed in Table 2 are based on *Tr*.

As introduced in [21], the constellation has 66 LEO layer satellites (LEO1 $\sim$ LEO66), 10 MEO satellites (MEO1 $\sim$ MEO10), and 3 GEO layer satellites (GEO1 $\sim$ GEO3). The constellation can achieve the following design objectives: (1) Higher layer satellites cover all of the lower layer satellites (or ground stations). (2) The number of satellites and orbits is minimal. These design objectives satisfy our need for constellation: hierarchical satellite coverage and simple system. The connections among the satellites are illustrated in Fig. 4.

TABLE 2. Parameters of the three-layer framework.

	Orbit height/km	Running period/h	Number of satellites	Orbit inclination/°
GEO satellites	36000	24	1x3	0
MEO satellites	10390	6	2x5	45
LEO satellites	895.5	12/7	6x11	90

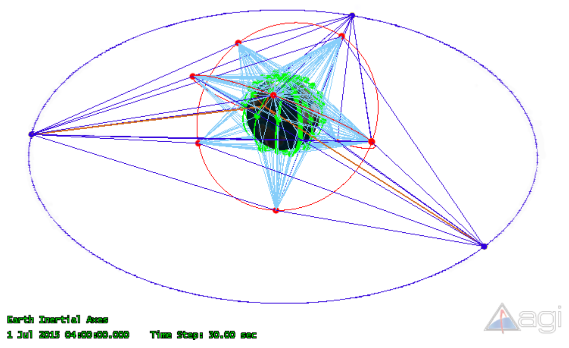

FIGURE 4. Illustration of three-layer constellation.

We set one ground stations, Beijing station ($116°E, 40°N$). One GEO satellite keeps continuous connection to Beijing station. MEO layer satellites keep continuous connections not only to GEO layer satellites, but also to LEO layer satellites. And at the same time, some LEO layer satellites cover the Beijing station continuously. To make emulation practicable and close to a real scenario, two points of settings are considered in the prototype as follows.

a: LINKS

We try to simulate the real delays of satellite links by abstracting parameters from STK. Then, we approach the real delay of satellite links with the help of Linux Traffic Control tool.

b: TRANSMISSION

We implement DTN in the satellite networks to meet the need of high transmission delay with the help of Interplanetary Overlay Network (ION) [23]. DTN is a clean-slate solution in satellite networks. It adopts the concept of store-and-forward and processes packets in a hop-by-hop fashion. DTN is designed for the extreme scenes where end-to-end transmission is not guaranteed [24]. Therefore, DTN has better performance in satellite networks than TCP [25]. DTN utilizes bundles [26] to transmit data and utilizes a convergence layer called Licklider Transmission Protocol (LTP) [27] to provide retransmission mechanism. ION is an implementation of DTN architecture and is designed to enable inexpensive insertion of DTN functionality into embedded systems. We deploy OpenFlow over ION by a method of tunnel.

That is, signaling data are transmitted in bundle tunnel when switches (MEO/LEO satellites) set up connections to controller (GEO satellite) and when controller sends instructions to switches. We run two different sets of ION configuration files on the control link (between GEO satellite and MEO/LEO satellites) and the data link (among MEO/LEO satellites) to separate the control plane and the forwarding plane.

POX controller is embedded on the GEO satellite node. We add a python-based module in the POX code to generate *idle_timeout* values for different flows before they are allocated to the switches. Open vSwitch is embedded on the MEO/LEO satellite nodes. We write and install shell scripts on the MEO/LEO satellite nodes mainly for two purposes. One is to help generate flows and mark the ToS field for different flows at the source node. We define the flows with ToS field modified to 48 as tos_1 traffic, the flows with ToS field modified to 56 as tos_2 traffic, and the flows with ToS field modified to 80 as tos_3 traffic. The other one is to help count real time flow table size at the forwarding nodes and send warnings to the controller when the TCAM utilization reaches 80% and 95%.

V. EXPERIMENTAL EVALUATION

In this section, we will introduce the design and the results of the experiments.

A. EXPERIMENTAL DESIGN

We design five kinds of *idle_timeout* strategies that we compare in the experiments.

S1: Static *idle_timeout* without considering handover: As the name suggests, these strategies use a constant *idle_timeout* value for all the flows. We set three different timeout values, 50s, 70s, and 100s.

S2: Dynamic *idle_timeout* without considering handover: This strategy uses a dynamic but simple police. The idle_timeout value increases exponentially with the arrival times of *packet_in* messages. If the number of flow entries reaches 95% of the TCAM space limit, the *idle_timeout* value of the subsequent flows will be reduced to 1s until the space utilization is less than 95%. This strategy is marked as dynamic-no handover.

S3: *DCT* without considering handover: This strategy only uses Dynamic Classification Timeout (*DCT*) algorithm without considering the handover.

S4: Dynamic *idle_timeout* considering handover: This strategy uses **S2** to generate the *idle_timeout* value. Then the value is re-calculated considering the handover. **S4** is somewhat like *TSMM*, except that the input of **S4** is calculated by **S2** but not *DCT* (**S3**). This strategy is marked as dynamic-handover.

S5: *TSMM*: This strategy uses Timeout Strategy-based Mobility Management (*TSMM*) algorithm. That is, this strategy uses *DCT* to generate the *idle_timeout* value and then re-calculated considering the handover. **S5** is actually *SAT-FLOW*.

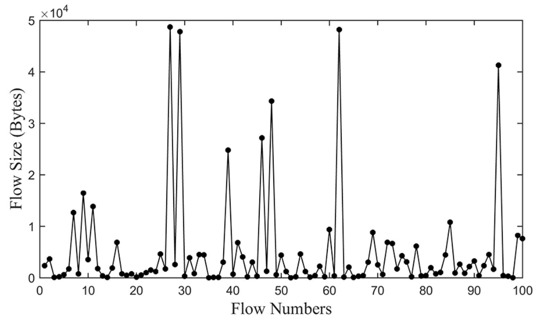

FIGURE 5. Flow sizes.

In the experiments, we set $T(i)_{init} = 1s$, the decrease factor $f_1 = 0.8, f_2 = 0.5$, and $f_3 = 0.1$, respectively. The proportion of tos_1 flows, tos_2 flows and tos_3 flows is 1:2:3. As is shown in Fig. 5, we generate 100 flow sizes that follow the lognormal distribution (mean=7.2374, std=1.9618) [28] to form a flow size set. Every flow chooses a flow size from the set randomly. We send 10 flows per second at a speed of 5 Mbit/s. Different flows are distinguished by the UDP port numbers. We create a set of 600 port numbers and every flow chooses a port number from the set randomly. The experiment time is 1500s. The parameters of link delay that we get from STK are shown in Table 3.

TABLE 3. Parameters of link delay.

	GEO satellites	MEO satellites	LEO satellites	Beijing station
GEO satellites	\	86ms	117ms	\
MEO satellites	86ms	66ms	50ms	\
LEO satellites	117ms	50ms	\	3ms
Beijing station	\	\	3ms	\

We have implemented two topologies to test the algorithms. As is shown in Fig. 6, **Topology 1** is simple and static. **Topology 1** has one GEO satellite, two MEO satellites, four LEO satellites (LEO1 ∼ LEO4) and one ground station. Three classes of traffic are sent from LEO1, LEO2, and LEO3, respectively. Then the traffic is forwarded by MEO1, MEO2, and LEO4 to the station.

Topology 2 is complicated and changing. We focus on the connections between Beijing Station and the LEO satellites. There are 11 LEO satellites that cover the station gradually from 8:00:00 am to 8:25:00 am, totally 1500s. Handover occurs at 8:04:26 (266s), 8:11:20 (680s), 8:12:12 (732s), 8:16:23 (983s), and 8:23:29 (1409s), forming six time durations, **duration 1** to **duration 6**. Fig. 7 illustrates the distances between Beijing station and LEO layer satellites. Based on the shortest distances, we choose LEO2, LEO3, LEO4, LEO5, LEO7, and LEO9 as the relay satellites during the six time durations. There may be multiple selection criteria, but our focus is on a handover scene, rather than how to choose relay satellite. Therefore, we do not consider

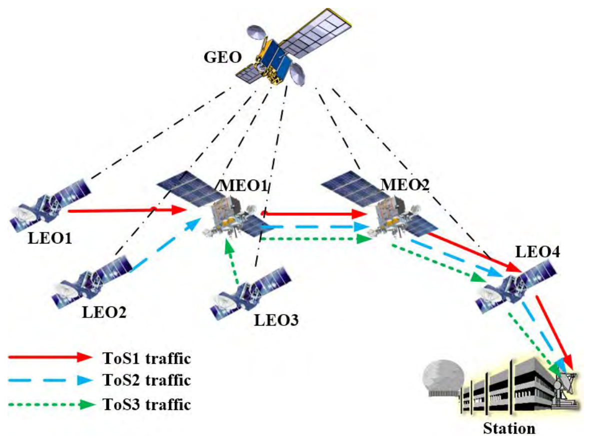

FIGURE 6. Experiment topology 1.

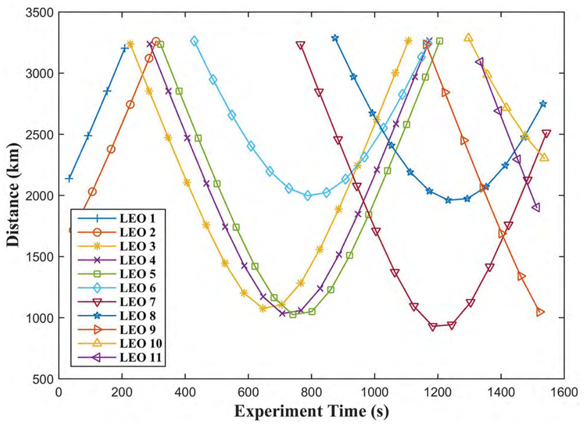

FIGURE 7. Distances between beijing station and LEO satellites in experiment topology 2.

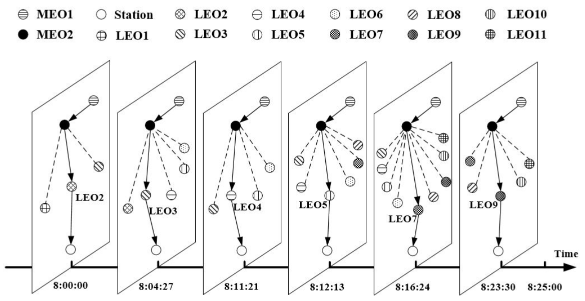

FIGURE 8. Experiment topology 2.

other selection criteria. Then, we choose MEO2 from the ten MEO satellites. MEO2 covers all the relay LEO satellites from 8:00:00 am to 8:25:00 am at the six time durations. Fig. 8 illustrates the connections among the satellites in **Topology 2**. For example, from 8:11:21 to 8:12:12, LEO3, LEO4, LEO5, and LEO6 cover Beijing station and we choose LEO4 as the relay satellite. Traffic is transmitted along the path MEO1$\rightarrow$ MEO2$\rightarrow$ LEO4$\rightarrow$ Beijing station. Then, the handover occurs at 8:12:13 am and the relay LEO satellite is switched to LEO5. Traffic is transmitted along the path MEO1$\rightarrow$ MEO2$\rightarrow$ LEO5$\rightarrow$ Beijing station.

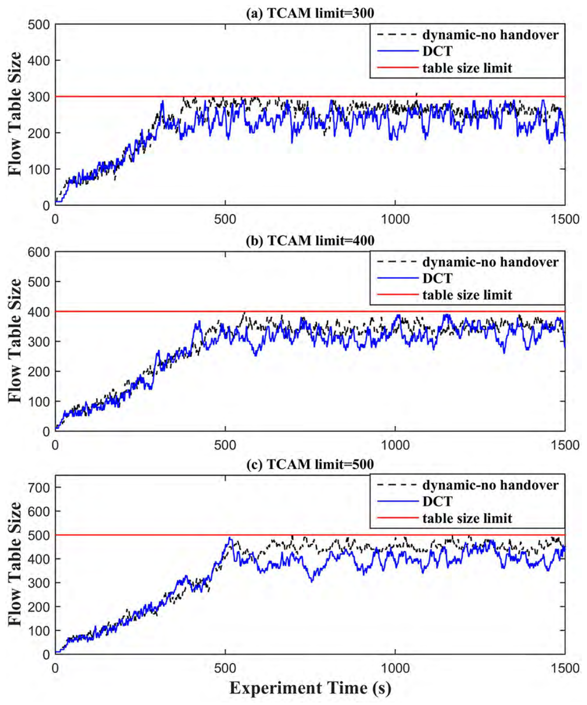

FIGURE 9. Flow table size of S3 vs S2 with different TCAM limit.

B. EXPERIMENTAL RESULTS

We design three sets of experiments. **Experiment 1:** *DCT* algorithm **(S3)** vs dynamic-no handover **(S2)** in **Topology 1** with TCAM space limit. **Experiment 1** is designed to verify the performance of *DCT* algorithm in terms of reducing the flow table size when there is a TCAM limit and no handover. **Experiment 2:** *TSMM* algorithm **(S5)** vs *DCT* algorithm **(S3)** in **Topology 2** without TCAM space limit. **Experiment 2** is designed to verify the performance of *TSMM* algorithm in terms of reducing the flow table size when handover occurs. **Experiment 3:** *TSMM* algorithm **(S5)** vs dynamic-handover **(S4)** vs dynamic-no handover **(S2)** vs static *idle_timeout* **(S1)** in **Topology 2** with TCAM limit=300, 400, and 500, respectively. We design **Experiment 3** to verify the performance of *TSMM* at different TCAM limit. The TCAM limits (300 entries, 400 entries, and 500 entries, respectively) are on the flow table in MEO2 satellite node. We test **five parameters** in the three experiments, the flow table size at MEO2 satellite node, the drop-flow rate at MEO2 satellite node, the normalized number of table-misses of the traffic with different ToS at MEO2, the throughput at Beijing station node, and the cumulative *idle_timeout* values of the flow rules. Then, we compare the three experiments based on the five parameters. The details are as follows.

1) COMPARISON 1: THE FLOW TABLE SIZE

S3 vs S2: The flow table size of **S3** and **S2** with different TCAM limit is shown in Fig. 9. We aim to validate that the *DCT* algorithm can reduce the flow table size under the limited TCAM space conditions. The TCAM space limit is 300, 400, and 500, respectively. We can see from Fig. 9 that,

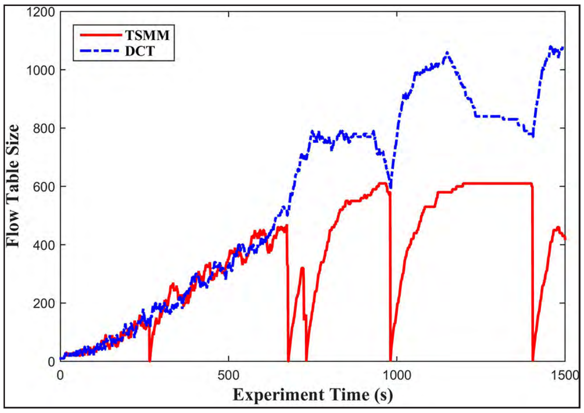

FIGURE 10. Flow table size of S5 vs S3.

in comparison to **S2**, when the TCAM space limit is 300, the average flow entry number is reduced 9.05% by **S3** (**S3**: 211, **S2**: 232). When the TCAM space limit is 400, the average flow entry number is reduced 9.22% by **S3** (**S3**: 307, **S2**: 340). When the TCAM space limit is 500, the average flow entry number is reduced 9.39% by **S3** (**S3**: 328, **S2**: 362). The flow table size increases slowly at the beginning. The *idle_timeout* value is small because of the **Stage 1**, so the flow entries are evicted quickly as the new flow entries are installed. At a result, the total number of flow entries increases slowly during the first several hundred seconds.

We can conclude that the **S3** has an advantage in controlling the flow table size under limited TCAM space conditions in comparison to **S2**. The main difference between **S3** and **S2** is that **S3** has the **Stage 2**. When the remaining TCAM space is not too much, the **Stage 2** can ensure that the flow entries do not increase too fast. In addition, **Stage 2** starts before it is too late to control the flow table size. Therefore, **S3** does well in reducing the flow table size. By analysing the experimental data, we find that the bigger the TCAM space limit is, the more obvious effect **S3** has on the flow table size.

S5 vs S3: Fig. 10 shows the flow table size of **S5** and **S3** when the satellite links handover occurs. We aim to validate that the *TSMM* algorithm can reduce the flow table size under the handover conditions. We can see from Fig. 10 that the flow entries are evicted as soon as the handover occurs by implementing **S5**. In this way, the flow entry number is reduced and reaches 600 at most when there is no TCAM space limit. However, only using **S3** performs not so well in **Topology 2**. The flow entry number reaches 1100 at most. The average flow entry number is reduced 37.7% by **S5** (**S5**: 352, **S3**: 565). We analyze the experimental data in detail and find that: in comparison to **S3**, the average flow entry number is reduced 0.75% by **S5** in **duration 1**, 2.61% in **duration 2**, 71.28% in **duration 3**, 37.53% in **duration 4**, 40.66% in **duration 5**, and 68.24% in **duration 6**.

We will give some analyses about Fig.10. **(1)** The maximum flow table size reaches 600 because we send 600 kinds of different flows in our experiment. That is to say, there is a flow entry for every different flow. **(2)** In **duration 1**,

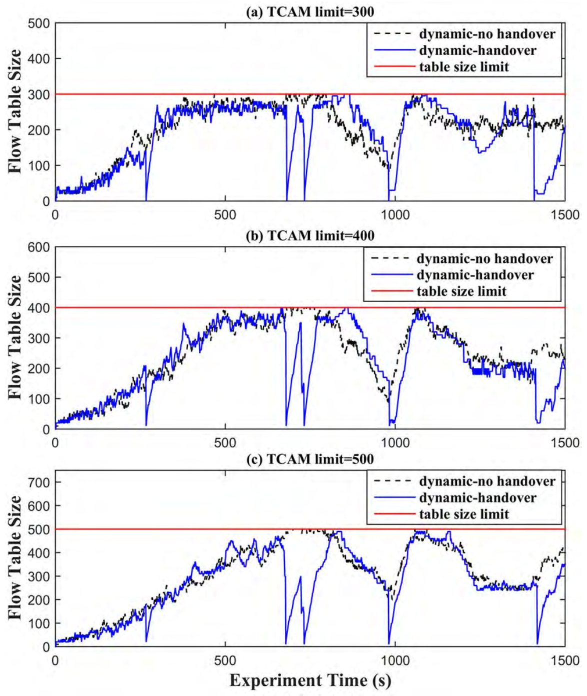

FIGURE 11. Flow table size of S4 vs S2.

the average flow table size of **S5** and **S3** are nearly the same because there is no handover occurs. **(3)** In **duration 2**, the average flow table size of **S5** is slightly smaller than that of **S3**. Because the flow entries still remain in the flow table after handover without **S5**. The percentage of decrease is not big because the *idle_timeout* values are relatively small at the beginning. **(4)** In **duration 3** to **duration 6**, the flow table size of **S3** increases quickly after handover occurs. Because the flow entries are still maintained after handover and the *idle_timeout* values are very big. The installation of new flow entries will lead to rapid increase in the flow table size. **(5)** In **duration 2** to **duration 6**, the flow table size of **S3** decreases after handover occurs. Because the flow entries that belong to the former connection begin to expire. We can conclude that implementing **S5** has an advantage in control the flow entry number under handover conditions in comparison to only using the **S3**. That is to say, the mechanism of evicting the flow entries that belong to the former connection can help to reduce the TCAM space occupation.

S4 vs S2: The flow table size of **S4** and **S2** is shown in Fig. 11. In comparison to **S2**, when the TCAM space limit is 300, the average flow entry number is reduced 6.9% by **S4** (**S4**: 189, **S2**: 203). When the TCAM space limit is 400, the average flow entry number is reduced 8.54% by **S4** (**S4**: 225, **S2**: 246). When the TCAM space limit is 500, the average flow entry number is reduced 11.18% by **S4** (**S4**: 270, **S2**: 304).

We can see that the curve trends of **S4** and **S2** in Fig. 11(a), (b) and (c) are almost the same. Because the first part of **S4** is actually **S2**. That is to say, both **S4** and **S2** have the same mechanism to generate the initial *idle_timeout*. The difference is that the flow entries are evicted as soon as the

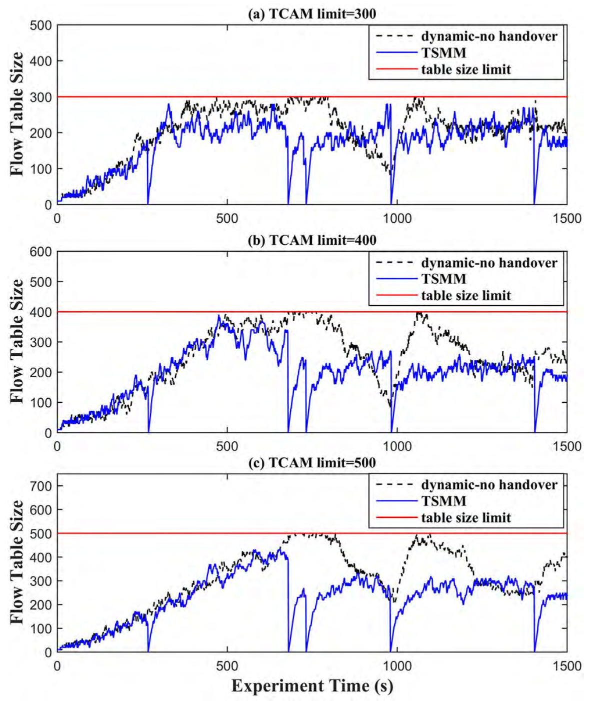

FIGURE 12. Flow table size of S5 vs S2.

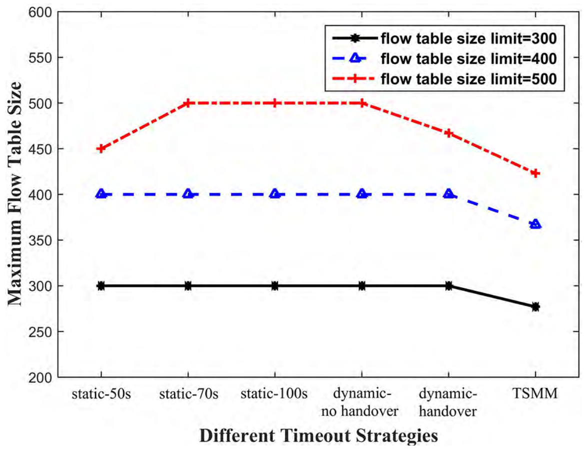

FIGURE 13. Maximum flow table size of different timeout strategies in Experiment 3.

handover occurs by using **S4**. By analysing the experimental data, we find that most of drop-flows of the **S2** appears after handover. Because the old flow entries are unexpired and still maintained in the flow table. The new arrived flows cause a sudden increase in flow table size. Then, some of the new flows are dropped because the TCAM space is not enough. In addition, we can conclude that the higher the upper limit is, the bigger the proportion of decrease of the average flow entry number is.

S5 vs S2: The flow table size of **S5** and **S2** is shown in Fig. 12. We can see that the flow entries are evicted as soon as the handoff occurs. **(1)** When the TCAM limit is 300 (as is shown in Fig. 12 (a)), the average flow entry number is reduced 15.27% by **S5** (**S5**: 172, **S2**: 203). The detail analysis is that, the flow entry number is reduced 2.31% by **S5** in **duration 1**, 17.35% in **duration 2**, 53.12% in **duration 3**, 10.62% in **duration 4**, 8.94% in **duration 5**, and 24.29% in **duration 6**. **(2)** When the TCAM limit is 400 (as is shown in Fig. 12 (b)), the average flow entry number is reduced 19.10% by **S5** (**S5**: 199, **S2**: 246). In details, the flow entry number is reduced 2.14% by **S5** in **duration 1**, 6.09% in **duration 2**, 58.68% in **duration 3**, 27.48% in **duration 4**, 22.72% in **duration 5**, and 32.36% in **duration 6**. **(3)** When the TCAM limit is 500 (as is shown in Fig. 12 (c)), the average flow table size is reduced 24.34% by **S5** (**S5**: 230, **S2**: 304). In details, the flow table size is reduced 2.96% by **S5** in **duration 1**, 5.02% in **duration 2**, 63.06% in **duration 3**, 35.39% in **duration 4**, 25.37% in **duration 5**, 42.77% in **duration 6**.

We can see that comparing to Fig. 10, the tendency of **duration 1** for **S5** and **S2** is the same as Fig. 10. However, **duration 2** to **duration 6** are different in Fig. 12 (a). Because **S5** is in **Stage 2** and **Stage 3**, and **S2** reaches the upper limit when the TCAM space limit is 300. As a result, the proportion of decrease of the average flow entry number in Fig. 12 (a) is smaller than that of Fig. 10. Similarly, the upward trend which is the same as Fig. 10 ends at **duration 2** and **duration 3** when the TCAM limit is 400 and 500. In addition, we can conclude that the proportion of decrease of the average flow entry number is bigger when the TCAM limit is high. When there is no TCAM limit (as is shown in Fig. 10), we get the biggest decrease proportion (37.7%). At last, we can conclude that **S5** has an advantage in control the flow table size under the conditions of limited TCAM space and satellite links handover.

S5 vs S4 vs S2 vs S1: The maximum flow table size **Experiment 3** is shown in Fig. 13. **(1)** When the TCAM space limit is 300, the maximum flow table sizes of **S1**, **S2**, and **S4** are 300. That is, the flow entry number of **S1**, **S2**, and **S4** reaches the TCAM space limit. However, the maximum flow table size of **S5** is 277 and is less than the TCAM space limit. **(2)** When the TCAM space limit is 400, the maximum flow table sizes of **S1**, **S2**, and **S4** reach the upper limit of TCAM space. Only **S5** is an exception (maximum flow table size is 364). **(3)** When the TCAM space limit is 500, **S5**, **S4**, and one case of **S1** (static *idle_timeout* is 50s) do not reach the upper limit. The maximum flow table size of **S5** (423) is smaller than that of **S4** (460) and **S1** (451). The percentage of decrease for **S5** is 7.67%, 9%, and 15.4% respectively relative to the TCAM space limit (300, 400, and 500). We can conclude that **S5** dose better in decreasing the maximum flow table size under the conditions of limited TCAM space and satellite links handover in comparison to the other strategies. In addition, the percentage of decrease is bigger when the TCAM space limit is higher.

2) COMPARISON 2: THE NORMALIZED NUMBER OF TABLE-MISSES

Fig. 14 (a), (b), and (c) illustrates the normalized number of table-misses of **S5**, **S4**, **S2**, and **S1** when the TCAM space is limited and satellite links handover occurs. We aim to verify the performance of the **Stage 3** in **S5** in terms of the

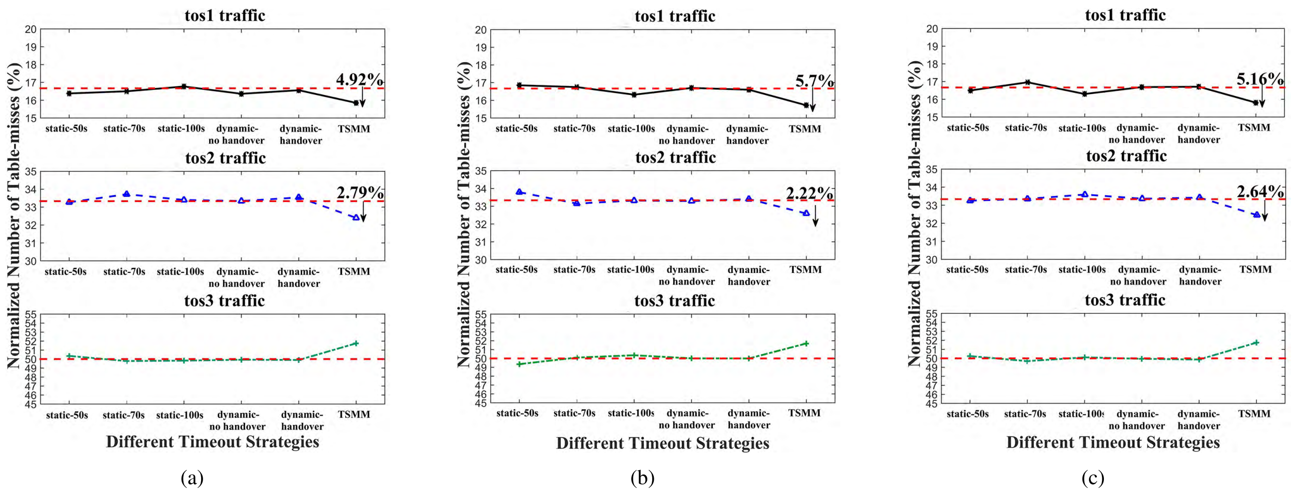

FIGURE 14. Normalized number of table-misses of different tos traffic for S5 vs S4 vs S2 vs S1. (a) TCAM limit = 300. (b) TCAM limit = 400. (c) TCAM limit = 500.

table-misses. Here, we take **S5** as an example to show how to calculate the normalized number of table-misses for different ToS traffic. We count the sum of *packet_in* messages for **S5**, and the number of *packet_in* messages for tos_1 traffic, tos_2 traffic, and tos_3 traffic in the tests for **S5**, respectively. Then the *packet_in* messages for tos_1 divided by the sum is the normalized value of tos_1 traffic for **S5**. **(1)** When the TCAM limit is 300 (as is shown in Fig. 14 (a)), the normalized numbers of table-misses of tos_1 traffic and tos_2 traffic for **S5** are decreased by 4.92% and 2.79%. **(2)** When the TCAM limit is 400 (as is shown in Fig. 14 (b)), the normalized numbers of table-misses of tos_1 traffic and tos_2 traffic for **S5** are decreased by 5.7% and 2.22%. **(3)** When the TCAM limit is 500 (as is shown in Fig. 14 (c)), the normalized numbers of table-misses of tos_1 traffic and tos_2 traffic for **S5** are decreased by 5.16% and 2.64%.

We can see from Fig. 14 that the normalized value of tos_1 traffic, tos_2 traffic, and tos_3 traffic for **S1**, **S2**, and **S4** stay at 16.67%, 33.33%, and 50%. Because the proportion of tos_1 flows, tos_2 flows, and tos_3 flows that are sent at the source node is 1:2:3. If there are no strategies about classified traffic and the number of flows in the experiment is big enough, the proportion of table-misses for tos_1 traffic, tos_2 traffic, and tos_3 traffic should be 1:2:3 in one trial. In addition, because the decrease factors $f_1 > f_2 > f_3$, the proportion of flow entries maintained in the flow table for tos_1 and tos_2 traffic is bigger than that of tos_3. Then, we can see that the proportions of table-misses of tos_1 traffic and tos_2 traffic for **S5** are smaller than 16.67% and 33.33% while the proportion of table-misses of tos_3 traffic is bigger than 50% after normalization. We can conclude that the classified decrease factors in **Stage 3** can reduce table-misses for high priority traffic.

3) COMPARISON 3: THE DROP-FLOW RATE

The drop-flow rate of **S5**, **S4**, **S2**, and **S1**, which is caused by the limited TCAM space is shown in Fig. 15. Here, we count

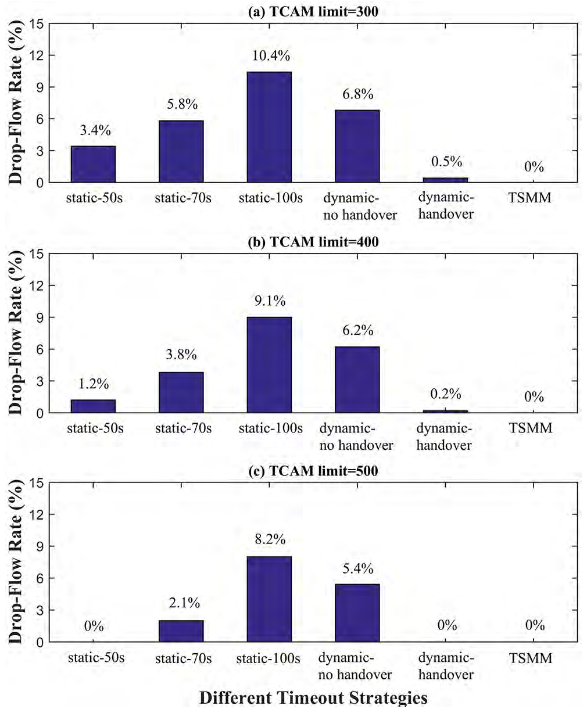

FIGURE 15. Drop-flow rate of S5 vs S4 vs S2 vs S1.

the number of drop-flows in each trial and calculate the drop-flow rate for each strategy respectively. We can see from Fig. 15 that when the TCAM space limit is 300 and 400, the drop-flow rate of **S5** is the smallest (0%). When the TCAM space limit is 500, **S5**, **S4**, and one case of **S1** (static *idle_timeout* is 50s) cause no drop-flows. We can say that if the TCAM space limit is high and the static *idle_timeout* is small, there will be no drop-flows either. But a large number of *packet_in* messages will be triggered in this case. **S4** has a low drop-flow rate because **S4** use a flow entries eviction mechanism considering handover. As the experimental data

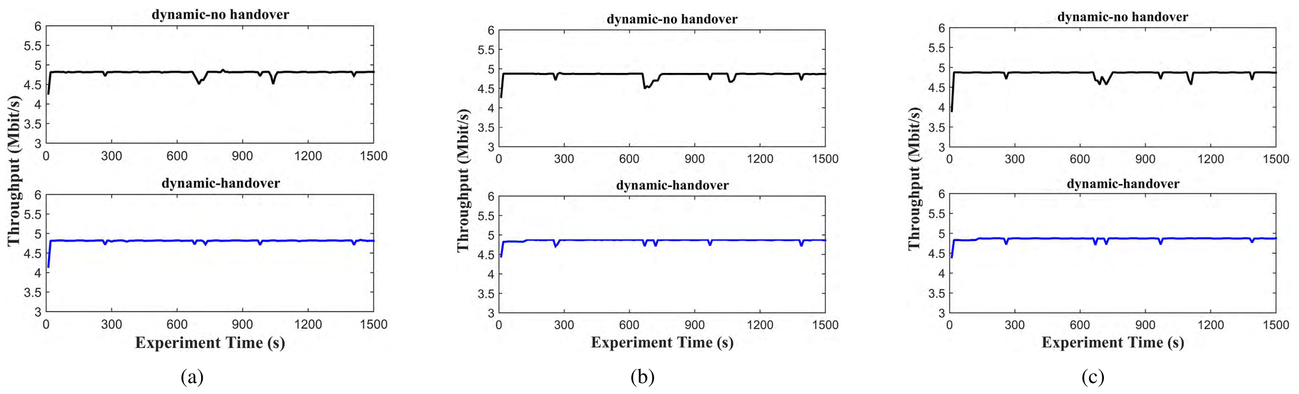

FIGURE 16. **Throughput of S4 vs S2. (a) TCAM limit = 300. (b) TCAM limit = 400. (c) TCAM limit = 500.**

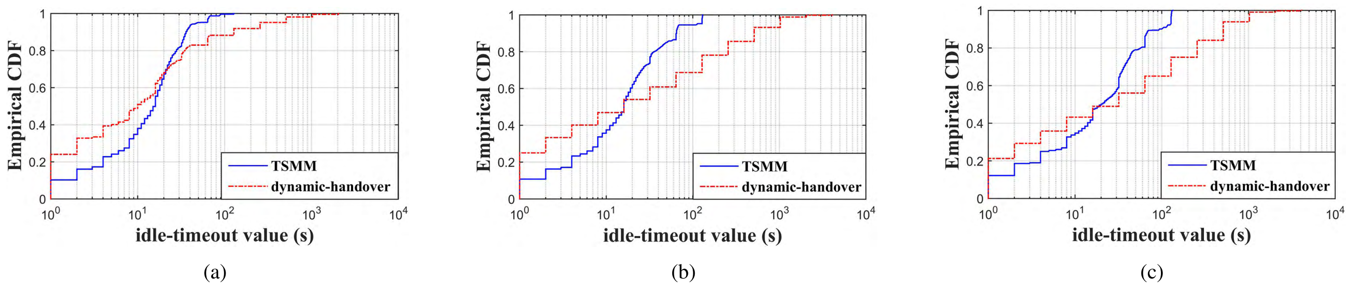

FIGURE 17. **Idle_timeout value of S2 vs S5. (a) TCAM limit = 300. (b) TCAM limit = 400. (c) TCAM limit = 500.**

show, most of the drop-flows are caused by handover. Therefore, **S4** also has a good performance in terms of drop-flow rate. When the static *idle_timeout* is set to 100s, the drop-flow rate is the biggest (10.4%, 9.1%, and 8.2%) among the rest strategies. Because the flow entries are evicted after a long time, the old flow entries occupied the flow table and cause not enough space for the subsequent flows. The drop-flow rate of **S4**, **S2** and **S1** is smaller when the TCAM limit is high than that when the TCAM limit is low. We can conclude that the drop-flow rate can be reduced 8.2%-10.4% at most by using **S5**.

4) COMPARISON 4: THE THROUGHPUT

The throughput of **S4** and **S2** is shown in Fig. 16 (a), (b), and (c), respectively. The experimental data are captured by wireshark at the Beijing station node. We can see that the throughput stays at about 4.9 Mbps. The transmission quality is effected slightly only when satellite link handover occurs when using **S4**. However, the throughput significantly decreases after the second, third, and fourth handover when using **S2**. Because a lot of flows are dropped at MEO2 after handover due to limit TCAM space and the Beijing station node receive less flows. The difference between **S4** and **S2** is that **S4** considers handover. Therefore, we can say that the flow entries eviction mechanism considering handover in **S4** and **S5** can improve the transmission quality.

5) COMPARISON 5: THE idle_timeout VALUE

The cumulative *idle_timeout* values of the flow rules for **S2** and **S5** is shown in Fig. 17 (a), (b), and (c), respectively. Here, we collect the experimental data from the POX controller at the GEO satellite node. It is clear that the *idle_timeout* values of **S5** have a concentrated distribution. Comparing the three cases, the *idle_timeout* values in Fig. 17 (c) are evenly distributed while in Fig. 17 (a), more *idle_timeout* values are big. The ranges of *idle_timeout* values of **S5** in Fig. 17 (a), (b), and (c) are almost the same. However, the largest values of **S2** increase with the TCAM limit increases. Comparing **S2** and **S5**, the proportion of small *idle_timeout* values (1s-15s) for **S2** is bigger than that of **S5**. The largest *idle_timeout* values of **S2** (about 2×10^3s-4×10^3s) are far bigger than those of **S5** (about 10^2s-2×10^2s). We can say that the *idle_timeout* values of **S5** stay in a reasonable range, that is, not too small and not too large. Too small *idle_timeout* values will lead to *packet_in* messages being triggered frequently while too large *idle_timeout* values will lead to large flow table size.

We can conclude from **Comparison 1**, **Comparison 2**, **Comparison 3**, **Comparison 4** and **Comparison 5** that the performance of flow table management in SDSN can be improved by implementing *SAT-FLOW* in terms of flow table size, drop-flow rate, table-misses, transmission quality during handover, and *idle_timeout* values distribution.

VI. FURTHER DISCUSSION

In this paper, we have proposed *SAT-FLOW* to achieve flow table management by adjusting the *idle_timeout* dynamically. There are some directions that could be further explored. We list some important ones as follows.

Firstly, more considerations on the table-misses. In this paper, our focus is mainly on reducing the flow table size and the drop-flows. We only try to reduce the table-misses for the high priority traffic. However, a lot of table-misses mean too many *packet_in* messages. The control link delay is very high (about 85ms to 117ms) in SDSN. Therefore, the transmission delay will be too high if there are too many *packet_in* messages. The balance between flow table size and table-miss has to be studied necessarily. We can solve this problem by adding a game theory-based algorithm to reach the equilibrium point between flow table size and table-misses.

Secondly, multiple GEO satellites coordination. In this paper, one GEO satellite covers the MEO/LEO satellites from 8:00:00 am to 8:25:00 am continuously. However, there may be the case that more than one GEO satellite participates in the control task, for example, when the transmission path is too long. Therefore, it is necessary and challenging to investigate the coordination mechanism among the GEO satellites. Preliminarily, the coordination mechanism should include an eastbound and westbound communication protocol that handles the flow entries synchronization and strategies negotiation.

Thirdly, generalized model for the flow table management problem. We have modeled the OpenFlow-based and DTN-based SDSN system in our previous work. However, the model did not contain analysis about flow table size or table-miss. We may expect to give a generalized model to analyze the flow table management problem.

VII. SUMMARY AND FUTURE WORK

In this paper, we studied the flow table management problem in SDSN. A multi-strategy flow table management method for SDSN, *SAT-FLOW*, is proposed to solve this problem. *SAT-FLOW* is composed of two heuristic algorithms, *DCT* and *TSMM*. Then, we introduced the implementation of *SAT-FLOW* in the prototype. Finally, the experimental results show that *SAT-FLOW* can achieve the goal of flow table management in SDSN. Our future work will follow three directions. Firstly, we plan to expand the scale of the experiments and design more proper scenarios to valid *SAT-FLOW*. We will extend the current work to a integrated space-terrestrial scenario. Secondly, a thorough investigation on the *SAT-FLOW* is desired, such as the overhead and cost in a large-scale deployment scenarios. Finally, we will do our best to search for real satellite traffic trace and test *SAT-FLOW*. In this way, we will make the conclusion more convincing.

REFERENCES

[1] F. Hu, Q. Hao, and K. Bao, “A survey on software-defined network and OpenFlow: From concept to implementation,” *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.

[2] A. Lara, A. Kolasani, and B. Ramamurthy, “Network innovation using OpenFlow: A survey,” *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, 1st Quart., 2014.

[3] J. Bao, B. Zhao, W. Yu, Z. Feng, C. Wu, and Z. Gong, “OpenSAN: A software-defined satellite network architecture,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 347–348, 2014.

[4] L. Bertaux *et al.*, “Software defined networking and virtualization for broadband satellite networks,” *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 54–60, Mar. 2015.

[5] R. Ferrús *et al.*, “SDN/NFV-enabled satellite communications networks: Opportunities, scenarios and challenges,” *Phys. Commun.*, vol. 18, pp. 95–112, Mar. 2015.

[6] B. Yang, Y. Wu, X. Chu, and G. Song, “Seamless handover in software-defined satellite networking,” *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1768–1771, Sep. 2016.

[7] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling flow management for high-performance networks,” *Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.

[8] S. Luo, H. Yu, and L. M. Li, “Fast incremental flow table aggregation in SDN,” in *Proc. IEEE 23rd Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2014, pp. 1–8.

[9] B. Leng, L. Huang, X. Wang, H. Xu, and Y. Zhang, “A mechanism for reducing flow tables in software defined network,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 5302–5307.

[10] T. Li, H. Zhou, H. Luo, Q. Xu, and Y. Ye, “Using SDN and NFV to implement satellite communication networks,” in *Proc. IEEE Int. Conf. Netw. Netw. Appl. (NaNA)*, Jul. 2016, pp. 131–134.

[11] S. Burleigh *et al.*, “Delay-tolerant networking: An approach to interplanetary Internet,” *IEEE Commun. Mag.*, vol. 41, no. 6, pp. 128–136, Jun. 2003.

[12] *POX*, accessed on Mar. 2017. [Online]. Available: http://sdnhub.org/tutorials/pox/

[13] *Open vSwitch*, accessed on Mar. 2017. [Online]. Available: http://www.openvswitch.org

[14] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, “Scalable flow-based networking with DIFANE,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 351–362, Oct. 2010.

[15] A. Zarek, Y. Ganjali, and D. Lie, “OpenFlow timeouts demystified,” Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2012.

[16] T. Kim, K. Lee, J. Lee, S. Park, Y.-H. Kim, and B. Lee, “A dynamic timeout control algorithm in software defined networks,” *Int. J. Future Comput. Commun.*, vol. 3, no. 5, pp. 331–336, 2014.

[17] H. Zhu, H. Fan, X. Luo, and Y. Jin, “Intelligent timeout master: Dynamic timeout for SDN-based data centers,” in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 734–737.

[18] A. Vishnoi, R. Poddar, V. Mann, and S. Bhattacharya, “Effective switch memory management in OpenFlow networks,” in *Proc. 8th ACM Int. Conf. Distrib. Event-Based Syst.*, 2014, pp. 177–188.

[19] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, “Infinite cacheflow in software-defined networks,” in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, 2014, pp. 175–180.

[20] E. Spitznagel, D. Taylor, and J. Turner, “Packet classification using extended TCAMs,” in *Proc. 11th IEEE Int. Conf. Netw. Protocols*, Nov. 2003, pp. 120–131.

[21] F. Long, *Satellite Network Robust QoS-Aware Routing*. Berlin, Germany: Springer, 2014.

[22] *Satellite Tool Kit (STK)*, accessed on Mar. 2017. [Online]. Available: http://www.agi.com

[23] S. Burleigh, “Interplanetary overlay network: An implementation of the DTN bundle protocol,” in *Proc. 4th IEEE Consum. Commun. Netw. Conf.*, Jan. 2007, pp. 222–226.

[24] B. Feng, H. Zhou, G. Li, H. Li, and S. Yu, “SAT-GRD: An ID/Loc split network architecture interconnecting satellite and ground networks,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 484–489.

[25] C. Caini, P. Cornice, R. Firrincieli, and D. Lacamera, “A DTN approach to satellite communications,” *IEEE J. Sel. Areas Commun.*, vol. 26, no. 5, pp. 820–827, Jun. 2008.

[26] K. L. Scott and S. Burleigh, *Bundle Protocol Specification*, document RFC 5050, 2007.

[27] M. Ramadas, S. Burleigh, and S. Farrell, *Licklider Transmission Protocol—Specification*, document RFC 5326, 2008.

[28] Q. Shao, “Measurement and analysis of traffic in a hybrid satellite-terrestrial network,” Ph.D. dissertation, School Eng. Sci., Simon Fraser Univ., Burnaby, BC, Canada, 2004.

TAIXIN LI (S'17) received the B.S. degree in telecommunications engineering from Beijing Jiaotong University (BJTU), China, in 2013, where he is currently pursuing the Ph.D. degree in telecommunications and information system. He was with the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. His research interests include next generation Internet, service function chain, software-defined networking, delay tolerant networking, and satellite networking.

HUACHUN ZHOU received the B.S. degree from the People's Police Officer University of China in 1986, and the M.S. degree in telecommunication automation and the Ph.D. degree in telecommunications and information system from Beijing Jiaotong University (BJTU) in 1989 and 2008, respectively. In 1994, he joined the Institute of Automation Systems, BJTU, where he is currently a Lecturer. From 1999 to 2009, he was a Senior Engineer with the School of Electronic and Information Engineering, BJTU, and with the Network Management Research Center, BJTU. Since 2009, he has been a Professor with the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. He has authored over 40 peer-reviewed papers and he is the holds 17 patents. His main research interests are in the area of mobility management, mobile and secure computing, routing protocols, network management, and satellite network.

HONGBIN LUO (S'06–M'07) received the B.S. degree from the Beijing University of Aeronautics and Astronautics, China, in 1999, and the M.S. degree(Hons.) and the Ph.D. degree in communications and information science from the University of Electronic Science and Technology of China, in 2004 and 2007, respectively. In 2007, he joined the School of Electronic and Information Engineering, Beijing Jiaotong University, where he is currently a Professor. From 2009 to 2010, he was a Visiting Scholar with the Department of Computer Science, Purdue University. He has authored over 50 peer-reviewed papers in leading journals (such as IEEE/ACM Transactions on Networking and the IEEE Journal on Selected Areas in Communications) and conference proceedings. His research interests are in the wide areas of network technologies, including routing, Internet architecture, and optical networking. He has served as a TPC member at the IEEE GLOBECOM, the IEEE ICC, the IEEE HPSR, and many others. In 2014, he received the Second Class National Invention Award and the National Science Fund for Excellent Young Scholars from the National Natural Science Foundation of China. He is currently an Associate Editor of the IEEE Communications Letters, *KSII Transactions on Internet and Information Systems*.

ILSUN YOU (SM'13) received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, South Korea, in 1997 and 2002, respectively, and the Ph.D. degree from Kyushu University, Japan, in 2012. From 1997 to 2004, he was with the THIN Multimedia Inc., Internet Security Company Ltd., and Hanjo Engineering Company Ltd. as a Research Engineer. He is currently an Associate Professor with the Department of Information Security Engineering, Soonchunhyang University, Asan, South Korea. He is a fellow of the IET. He has served or is currently serving as a main organizer of international conferences and workshops, such as IMIS, MobiWorld, MIST, SeCIHD, AsiaARES, and so forth. He is the Editor-in-Chief of the *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* and the *Journal of Internet Services and Information Security*. He is also on the Editorial Board of *Information Sciences*, the *Journal of Network and Computer Applications*, the IEEE Access, *Intelligent Automation and Soft Computing*, and the *International Journal of Ad Hoc and Ubiquitous Computing*. His main research interests include mobile networks and Internet security, Internet of Things security, and formal security analysis.

QI XU received the B.S. degree in telecommunications engineering from Beijing Jiaotong University (BJTU), China, in 2014, where he is currently pursuing the Ph.D. degree in telecommunications and information system. He was with the National Engineering Laboratory for Next Generation Internet Interconnection Devices, BJTU. His research interests include energy saving, next generation Internet, service function chain, and satellite networking.