Flow Scheduling in OBS Networks Based on Software-Defined Networking Control Plane

Wan Tang[1], Fan Chen[1], Min Chen[2], Guo Liu[1]
[1] College of Computer Science, South-Central University for Nationalities
Wuhan 430074, China
[e-mail: tangwan@scuec.edu.cn; chenfan_wh@outlook.com; liuguo@mail.scuec.edu.cn]
[2] School of Comupter Science and Technology, Huazhong University of Science and Technology
Wuhan 430074, China
[e-mail: minchen2012@hust.edu.cn]

Received October 12, 2015; revised October 24, 2015; accepted November 13, 2015; published January 31, 2016

Abstract

The separated management and operation of commercial IP/optical multilayer networks makes network operators look for a unified control plane (UCP) to reduce their capital and operational expenditure. Software-defined networking (SDN) provides a central control plane with a programmable mechanism, regarded as a promising UCP for future optical networks. The general control and scheduling mechanism in SDN-based optical burst switching (OBS) networks is insufficient so the controller has to process a large number of messages per second, resulting in low network resource utilization. In view of this, this paper presents the burst-flow scheduling mechanism (BFSM) with a proposed scheduling algorithm considering channel usage. The simulation results show that, compared with the general control and scheduling mechanism, BFSM provides higher resource utilization and controller performance for the SDN-based OBS network in terms of burst loss rate, the number of messages to which the controller responds, and the average latency of the controller to process a message.

Keywords: Optical burst switching (OBS); Software-defined networking (SDN); OpenFlow; Unified control plane (UCP); Flow scheduling

This work is supported in parted by the National Natural Science Foundation of China (NSFC) (No. 61103248 and No. 61572220), China Ministry of Science and Technology (MOST), the International Science and Technology Collaboration Program (Project No. 2014DFT10070), and the State Key Laboratory of Advanced Optical Communication Systems & Networks, China.

1. Introduction

Optical fibers are ideal transfer media for the growing traffic demands in metro and long-haul networks, providing vast capacity and offering steady long-reach transmission [1]. As the most mature optical switching technology, optical circuit switching (OCS) can provide guaranteed quality of service (QoS), but the wavelength utilization is inadequate [2]. Being the ideal optical switching, optical packet switching (OPS) can achieve high utilization; however, the immature optical processing technologies make it impractical. Fortunately, optical burst switching (OBS) [3] is proposed as a compromise between OCS and OPS, taking advantage of optical network performances and using the burst control packet (BCP) for advance reservations. Therefore, OBS is regarded as a more promising optical switching technology.

However, nowadays, commercial IP/optical multilayer networks are managed and operated separately, leading to inefficient network operation, resource waste, and high capital and operational expenditure [4, 5]. In order to solve those problems, network operators need to seek a unified control plane (UCP) that combines the management and operation of multiple layers [6]. A distributed control plane, called generalized multi-protocol label switching (GMPLS), is proposed to address the separate management in IP/optical multilayer networks, but it is overcomplicated. The GMPLS-based UCP proposed in [7] can provide unified management for OBS networks and wavelength switched optical networks (WSONs), but the information delivery among different nodes is unstable and complex due to the distributed nature of the GMPLS protocol [8]. Meanwhile, the experimental validation and evaluation of a GMPLS-based UCP is provided to process a multi-protocol label switching transport profile (MPLS-TP) in WSON [9]. Nevertheless, the hardware flexibility is insufficient because the standardization of GMPLS lags behind its hardware development [8]. Therefore, GMPLS is incompetent in offering a promising UCP for the future optical network.

Software-defined networking (SDN) [10], a novel network architecture proposed in 2009, extracts the control plane from the data plane of physical hardware and provides a programmable mechanism, operating and managing most parts of the network using the central controller(s). Providing high scalability and flexibility for SDN, OpenFlow is the most common communication protocol between the control layer and the forwarding layer in SDN architecture [11]. In an OpenFlow-based SDN, the flow table, maintained in network devices (e.g. OpenFlow-enabled switches), decides the forwarding method of packets and is generated and configured by the central controller. Regarded as a promising UCP for future optical networks, the OpenFlow-based SDN can operate and manage IP/optical multilayer networks with different switching granularities, and provides high capacity and scalability and low power consumption for optical transport networks [3,12].

The distributed control plane (e.g., GMPLS) is overcomplicated and requires high performance on each node of the network. Conversely, the central control plane, such as the SDN, transfers large amounts of computations from the nodes to the central controller and reduces the functional requirement of each node. In addition, central control can improve the BCP forwarding rate. A general control and scheduling mechanism for SDN-based OBS networks is proposed in [4, 5]. Nevertheless, the source edge-node sends a Packet-In message to the central controller after assembling a burst, which means that the central controller only processes one burst when responding to a Packet-In message. If the network offered load is high, the central controller and each OBS node have to process a large number of messages per second, ultimately resulting in low network resource utilization, i.e., needing more bandwidth between the nodes and the controller, and enlarging the average latency for the controller to process a message.

The application-based network operations (ABNO), an architecture published by the Internet Engineering Task Force (IETF), provides a way for different network technologies and use cases [12, 13]. The performance of the ABNO architecture for multilayer use cases with control plane and with OpenFlow in the optical layer was validated by A. Aguado et al. The ABNO architecture as a flexible SDN approach provides the support with non-OpenFlow networks [14].

To address the issue of network resource utilization in the SDN-based OBS, in this paper, we propose a burst-flow scheduling mechanism (BFSM) for SDN-based OBS networks. In BFSM, the source edge-node buffers a certain number of bursts with the same source and destination, and these bursts are composited into a burst-flow. The source edge-node sends only one Packet-In message, containing the information of a burst-flow, to the controller to process the bursts. Meanwhile, by introducing an extension of flow table structure based on OpenFlow specification 1.0.0 [15] and improving the network applications in the controller proposed in [3], a resource scheduling algorithm for BFSM is also proposed in our work.

The rest of this paper is organized as follows. In Section 2, we discuss the problem of the general control and scheduling mechanism for SDN-based OBS networks. Section 3 describes the BFSM, including the signaling procedure, the extension of the flow table structure in the OBS nodes, and the network applications provided by the controller. Section 4 introduces the resource scheduling algorithm applying in BFSM. Section 5 evaluates the performance of the BFSM via simulation. Finally, a conclusion is drawn in Section 6.

2. Problem Description

Compared with electrical transport, optical transport has the advantages of high capacity, low power consumption, and long-haul transmission. Not only will the optical network offer a central control UCP, but it will also provide a high channel utilization rate and scalability if introducing the OpenFlow-based SDN [11]. The OpenFlow-based SDN has been applied to optical wavelength networks [16, 17], but the work is still at an early stage and too complex to implement. Because prevailing electrical switching technologies cannot be directly applied to the optical SDN, Patel et al. presented a software defined optical network (SDON) architecture by spliting the control plane from the data plane. In addition, he discussed the key technologies of SDON including physical hardware technologies, common interface, and controller technologies [5].

In an SDN-based network, the controller is the scheduling and management center, and plays an extremely important role. The performance of the OpenFlow controller is evaluated in terms of the number of messages that a controller responds to per second, and the average latency of the controller to process a message [18, 19, 20]. The most time-consuming processes of a controller are those of reading the arrival OpenFlow messages from the network interface cards (NICs) and communicating with the OpenFlow-enabled switches [21, 22].

A field trial of an OpenFlow-based UCP for multilayer multigranularity optical switching networks was undertaken by Liu et al. to verify and evaluate the end-to-end latency, overall feasibility and efficiency of the SDN-based OBS [8]. Zhang et al. introduced a general control and scheduling procedure [4]. The source OBS edge-node sends out a Packet-In message to the controller after assembling a burst, and the controller computes an end-to-end path for the BCP of the burst. The controller inserts flow entries into all nodes along the BCP path except the source, and inserts a flow entry into the source edge-node until receiving the

Loading-Completion messages from other nodes along the path. A control and scheduling procedure without an acknowledgement mechanism is proposed [5], in which the nodes along the calculated routing path need not send the completion messages to the controller. The mechanisms discussed above lead to low network resource utilization, especially, the one proposed by Patel et al. which will result in burst loss if any related node does not process its flow entries on a timely basis.

If the number of Packet-In messages per unit time can be decreased while the number of bursts processed is unchanged, the required bandwidth between the nodes and the controller, and the average latency of the controller to process a message will be reduced. In [4, 5], the sending of one Packet-In message to the controller corresponds to one burst, meaning that the number of Packet-In messages is equal to the number of bursts. Based on the one-to-one corresponding mechanism (OTOCM), the controller has to process a large number of Packet-In messages and consume amounts of time in reading the arrival OpenFlow messages and communicating with OBS nodes. All of these hinder improvement in controller performance. Therefore, in this work, we propose a scheduling mechanism named BFSM to address the issue.

3. BFSM Base on SDN

In this section, we describe the key features of BFSM, and present the processing sequence of BFSM in SDN-based OBS networks.

3.1 Key Features Involved in BFSM

The key features of BFSM include the mechanism of assembling the burst-flow, the extension of the flow table structure, and the modified network applications in the SDN controller.

(1) *Assembling the burst-flow mechanism*

The control granularity of resource reservation in BFSM is the burst-flow, while the transmission granularity of BFSM is still the burst. The bursts with the same destination are considered to belong to the same burst-flow. In assembling the burst-flow, each SDN-based OBS node has one burst-flow buffer used to store the same-destination bursts and record the time of the burst aggregation. When the burst-flow buffer is filled by the same destination bursts, or the assembling time (i.e., the time for burst aggregation) reaches the threshold value, all of the bursts are assembled into a burst-flow. After launching the burst-flow, another burst-flow will be assembled in the burst-flow buffer.

(2) *Extended structure of the flow table*

In this work, we modify the construction of the flow table defined in OpenFlow specification 1.0.0 to meet the burst-flow scheduling requirements in BFSM. A flow table consists of several flow entries, and the main components of the improved flow entry in a flow table are illustrated in **Fig. 1**.

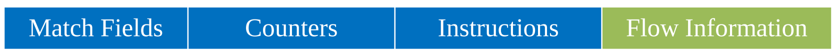


Fig. 1. Main components of a flow entry in the extended flow table

The Flow Information field is the extended part of a flow entry, and it contains some information of a burst-flow, such as the number of total bursts, the number of bursts that have reserved resources successfully, and the packet identities (IDs) of the bursts. Because there is insufficient resource for all bursts in a burst-flow when the offered load is high, the stored flow information will make the controller handle the burst-flow effectively. According to the information, the bursts in a burst-flow can be handled accurately, and the flow table will be updated in time as long as the last burst has been processed.

(3) *Network applications*

In [3], the SDON architecture (shown in **Fig. 2**) and the corresponding controller technologies are presented. Various functionalities of the control plane are installed as network applications in the controller, including Topology Storage, Routing, Resource Allocation, Access Control, etc. The controller stores the topological information in the Topology Storage component after receiving the features replied from each node. Routing is responsible for routing the burst-flows and stores the route table. Resource Allocation runs the resource scheduling algorithm to schedule burst-flows and allocate them channel resource, in order to achieve high resource utilization and avoid the high burst loss rate. Finally, Access Control manages the OpenFlow messages and takes charge of the communication with OpenFlow-enabled switches.

In our work, as shown in **Fig. 3**, we add Burst Buffer Management (BBM) in the Network Applications layer. It is one of the most important parts of BFSM and is used to manage the

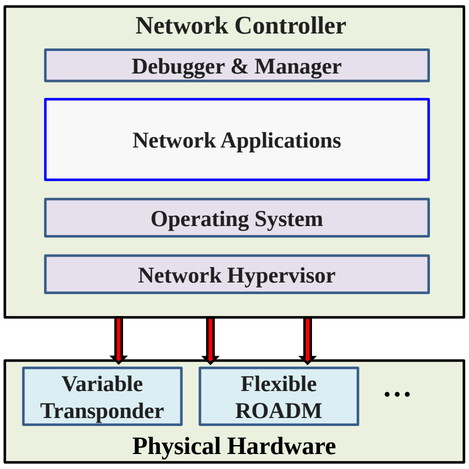


Fig. 2. SDON architecture [3]

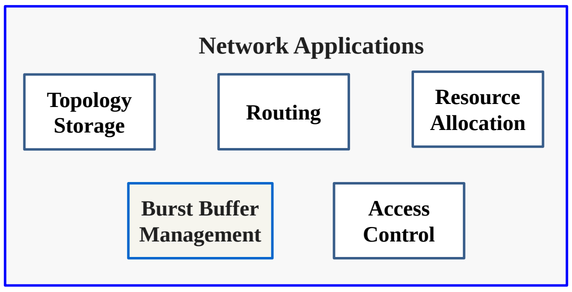


Fig. 3. Network applications in the controller

burst-flow buffers in the source OBS nodes. The source node sends out status messages to the BBM, containing the information of the burst and the burst-flow buffer. Meanwhile, BBM sends control messages to the source node for specifying the size of a burst-flow, setting the time threshold for buffering bursts, obtaining the remainder space in the burst-flow buffer, and so on.

3.2 Signaling and Processing Sequences

A UCP of the OBS network should meet the requirements of BCP switching and resource reservation. In the SDN-based OBS networks applying OpenFlow, a central controller processes the routing and resource scheduling, and then inserts appropriate flow entries into all OBS nodes on the selected routing path. Each OBS node should configure the flow table.

In traditional OpenFlow-based networks, if an arrival packet matches with the existing flow entries of the flow table in the OpenFlow switch, the corresponding action of the matching flow entry will be performed. However, if there is no matching flow entry, the switch will send the packet to the central controller for further processing. However, the switching granularity is the optical burst which is aggregated by packets with the same destination, and the bursts propagate in wavelength channels without any Optical/ Electronic (O/E) translation. The signaling procedures of the OTOCM increase the number of messages for the controller to respond to, the average latency of the controller to process each message, and the bandwidth occupancy of the control channels between the controller and the switches.

An OpenFlow-enabled OBS node is component of two parts: edge-node and core-node, which undertake different jobs. An edge-node assembles IP packets into bursts, sends the Packet-In message to the controller, and sends the BCP onto the control channel after being inserted with a burst-flow entry. The core node may have multiple physical ports of control channels and corresponding switching fabric ports for data channels [4].

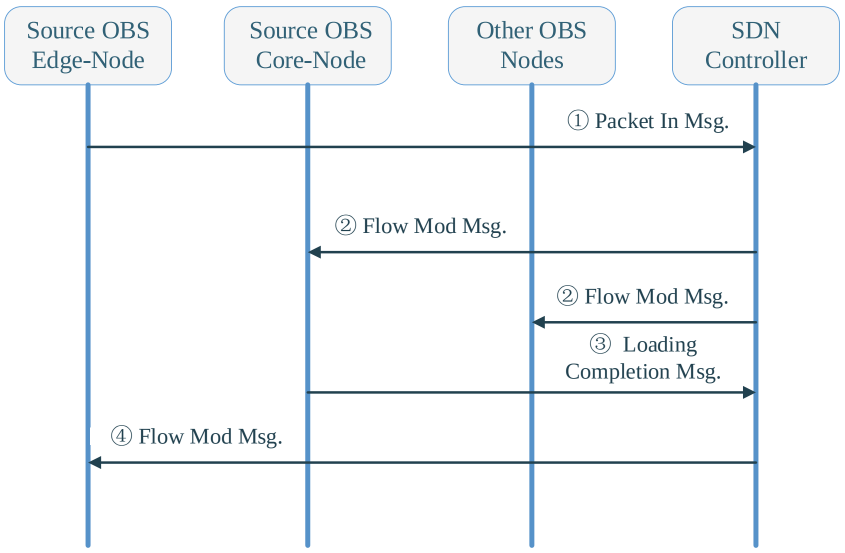


Fig. 4. Signaling procedures of BFSM between the controller and the OBS nodes

To solve the problems raised by the OTOCM mechanism, in the proposed BFSM, only one Packet-In message containing the information of a burst-flow is sent to the controller for resource reservation. **Fig. 4** describes the detailed signaling procedures of the BFSM between the controller and the OpenFlow-enabled OBS nodes.

The signaling procedures applied in the SDN-based OBS network presented in **Fig. 4** correspond to the Steps 1 to 4 depicted in **Fig. 5**. Comparing to the OTOCM proposed in [4, 5], in BFSM, a Packet-In message contains the information of a burst-flow instead of one burst, and the controller can process more bursts while responding one Packet-In message.

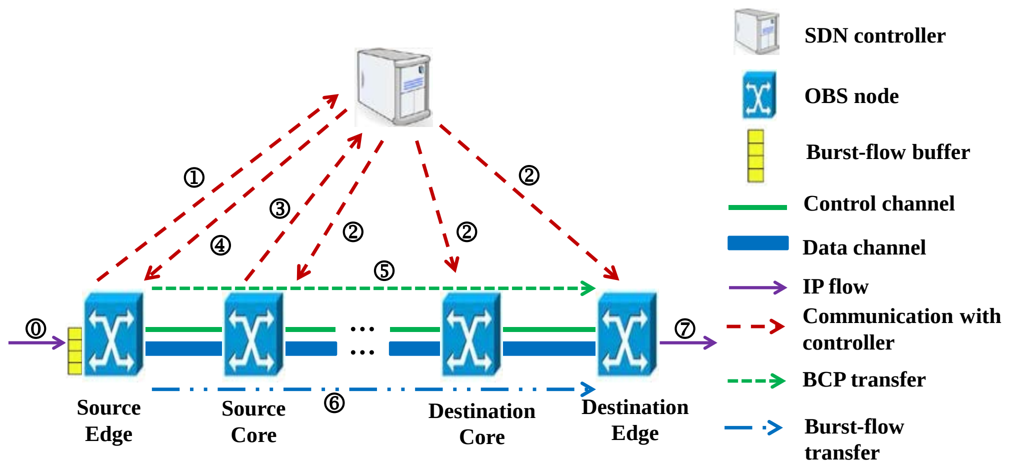


Fig. 5. Processing sequences of BFSM in the SDN-based OBS networks

The BFSM in the SDN-based OBS network works as follows.

Step 0: Generate a burst-flow

The source OBS edge-node aggregates packets belonging to a flow with the same destination into bursts and creates their corresponding BCPs. Then, the same-destination bursts are stored in the burst-flow buffer. When the burst-flow buffer is full, or all bursts of a flow have been assembled in the buffer, the bursts in the buffer will comprise a burst-flow.

Step 1: Send a Packet-In message

The source edge-node encapsulates the BCP of the first burst in a Packet-In message and sends it to the controller to request an end-to-end optical path for the burst-flow. This BCP contains the burst-flow information, including the number of bursts, burst size, etc.

Step 2: Insert flow entries into the OBS nodes except the source

The controller allocates resource and calculates an end-to-end optical virtual path for the burst-flow. The corresponding flow entries are encapsulated in a Flow-Mod message which is sent by the controller to add, delete or modify the entities of the flow table in the switch. The Flow Information field stores the resource reservation information (including the number of bursts, the number of bursts that have reserved wavelength successfully, etc.) of each burst in the burst-flow. The Flow-Mod message is sent to all OBS nodes along the selected path except the source edge-node.

Step 3: Send a Loading-Completion message

The OBS node informs the controller with a Load-Completion message to confirm that a new burst-flow entry is loaded. The source core-node of the optical virtual path sends a Loading-Completion message to the controller when the flow entry is loaded into it. In order to decrease the number of messages to which the controller responds, only the source core-node takes this step.

Step 4: Insert the flow entry into the source

Receiving the Loading-Completion message, the controller encapsulates BCP and flow entry in a Flow-Mod message, and sends the message to the source edge-node.

Step 5: Send BCPs of the burst-flow

The source edge-node updates its flow table and extracts the BCP of the first burst from the Flow-Mod message. After sending the extracted BCP on the control channel, the source node injects other BCPs of the burst-flow into the control channel in order. All BCPs are transported along the selected path, and the Packet IDs of their corresponding bursts are stored in each node on arrival in advance. If the resource reservation for a burst fails, its corresponding BCP would be discarded.

Step 6: Send bursts in the burst-flow

Once the BCP of a burst has set off, the corresponding burst is released to the data channel by the source edge-node after the offset time, and reaches the destination node transparently. Only the bursts having reserved successfully can enter into the data channel, and the other bursts are dropped at the source edge-node.

Step 7: Disassemble bursts and delete the corresponding flow entries

The destination edge-node disassembles bursts into packets, and all nodes along the selected path delete the corresponding flow entries.

4. Reservation with More Usage Algorithm

In this section, the reservation with more channel usage (RMCU) algorithm is presented, and some examples are given to illustrate the operation process of the RMCU. In the controller, RMCU is loaded in Resource Allocation to schedule burst-flows and allocate channel resource to them.

4.1 Algorithm Description

The RMCU aims to reserve more bursts of a burst-flow in one channel in order to make full use of channel resource and avoid high burst loss rate. In the flow table of an OBS node, a one-to-one relationship exists between the flow and its entry with one output port, and all bursts of a burst-flow transmit through the same data channel without any optical buffer. Therefore, the arrival time of each burst is predictable. The RMCU firstly contrasts the available channel resource during the time between the first burst and the last arriving burst. Then, the channel having the most vacant resource for the burst-flow will be selected. When the controller makes reservations on the last link, it applies the RMCU algorithm to modify the previous link reservation information. The old information is inconsistent with the latest information of the link reservation. That is to say, if the last link and its previous one are both reserved successfully, the resource reserved by the bursts in the previous link would be released, and the corresponding reservation information is modified to be Failure.

In **Table 1**, the notations for describing the RMCU are given. The leisure horizon is the last time for the channel to be scheduled for the burst-flow. Because of the interval time between bursts in a burst-flow, the channel resource has not been reserved completely, which generates some gaps between burst reservations. The RMCU regards the resource within the gaps as available leisure resource elements and stores them in the gap lists for new reservations.

Table 1. Notation list

Notation	Description	Notation	Description
C	set of channels in a link	L_i	gap list of the *i*-th channel
C_i	the *i*-th channel in a link	t_i	the leisure horizon of the *i*-th channel
t_s	beginning of reservation time	t_e	end of reservation time
B	reservation information in current link	B'	reservation information in previous link
N	the number of bursts in a burst-flow	n_i	the number of available leisure resource elements in the gap list of the *i*-th channel

Next, we describe how the RMCU works to reserve the wavelength bandwidth on the *i*-th link of a routing path of a burst-flow.

```
Algorithm: RMCU

Step 1: Calculate $C_i$ available resource, and select the appropriate channel for transporting
        If (the leisure horizon $t_i$ precedes the burst-flow arrival time $t_s$)
            Select the channel with the minimum gap between $t_i$ and $t_s$.
        Else
            Calculate $n_i$ in the $L_i$ between $t_s$ and $t_e$, and select the channel with maximum $n_i$.
Step 2: Reserve resource for the bursts in the burst-flow
        If (the resource storing in $L_i$ is unused)
            Reserve $N$ bursts in the burst-flow, and update B and $t_i$.
            Store the available gaps in the $L_i$.
        Else
            Access $L_i$ to find the available resource element for each burst in the burst-flow.
            If (there is available resource)
                Reserve bursts, and update $B$.
            If (the leisure horizon $t_i$ precedes the subsequent burst arrival time)
                Reserve all subsequent bursts in the burst-flow, and update B and $t_i$.
            Store the available gaps in $L_i$.
Step 3: If (the last link is to make reservations)
        Modify $B'$ which is inconsistent with $B$, and release the corresponding reservation resource.
```

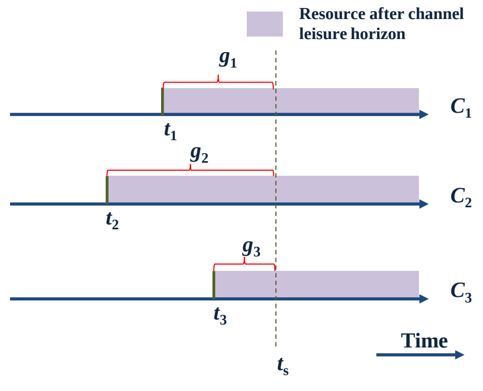
The RMCU can reuse the wavelength bandwidth for transmitting bursts as much as possible, and sometimes reallocate the resources to other burst-flows. The channels with more available resource will be reused preferentially. If the channel leisure horizon precedes the arrival time of any burst of the other burst-flows, the RMCU can successfully reserve bandwidth for the subsequent bursts.

4.2 Illustration Examples

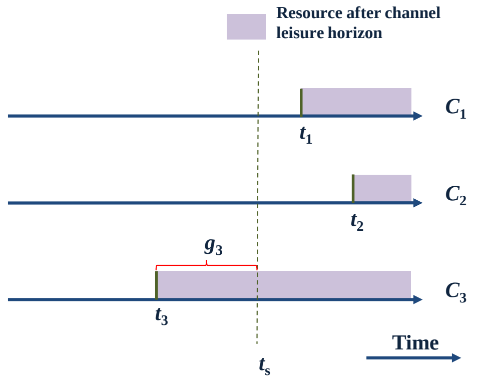
In the BFSM-based OBS networks applying the RMCU algorithm, the channels selected for transporting burst-flows have the minimum gap between t_s and the channel leisure horizon, or have the most available resource in the gap list. Meanwhile, the RMCU stores the available gaps between burst reservations in the gap list for the next allocation, and makes new reservations within them. Therefore, the RMCU can reduce the waste of channel resource and allocate more resource for burst-flows, which improve the resource utilization and reduce the burst loss rate.

We use some examples to illustrate the operation of the proposed RMCU. Assume a link has three channels that have different channel leisure horizons, i.e., t_1, t_2 and t_3, respectively. g_i is the gap between t_i and t_s.

For example, in **Fig. 6(a)**, three channel leisure horizons are less than the burst-flow arrival time t_s, and g_3, the gap between the third channel leisure horizon t_3 and t_s, is the smallest. Therefore, the third channel C_3 is selected for transmitting the burst-flow. However, when only one channel leisure horizon is less than t_s, the channel is the only choice for the burst-flow (shown in **Fig. 6(b)**). Meanwhile, a new reservation can be made within the gap g_3, and RMCU stores it in the gap list.



(a) Three channel leisure horizons before t_s



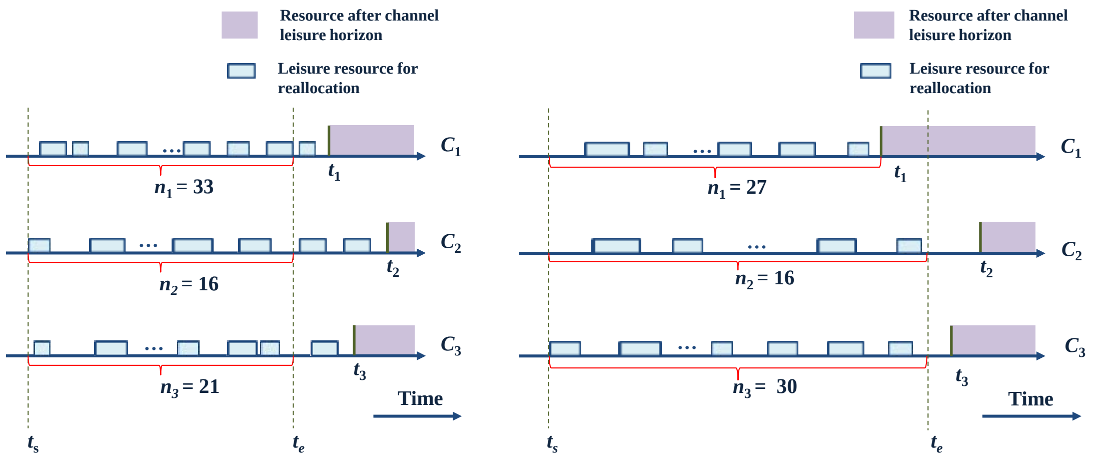
(b) One channel leisure horizon before t_s

Fig. 6. The channel leisure horizons before the burst-flow arrival time t_s

In **Fig. 7**, for a burst-flow, t_s is less than the three channel leisure horizons, which results in making use of the leisure resource element stored in the gap lists. As shown in **Fig. 7(a)**, for the time period between t_s and t_e, the channel resource was reserved. Then, the RMCU chooses the first channel C_1 which has maximum leisure resource elements (n_1=33) in the gap list between t_s and t_e for a burst-flow. If the channel leisure horizon t_1 is less than t_e (**Fig. 7(b)**), the third channel C_3 having the maximum leisure resource elements (n_3=30) will be selected for the burst-flow.

5. Simulation and Evaluation

In this section, the performance of the proposed RMCU is compared with the classical OBS and OTOCM via simulation in terms of burst loss rate, the number of Packet-In messages that a controller responds to per second, and the average latency of the controller to process a message. The latter two are main performance indexes of the SDN controller [18].



(a) Three channel leisure horizons after t_e (b) Two channel leisure horizons after t_e

Fig. 7. The channel leisure horizons after the burst-flow arrival time t_s

5.1 Simulation Setting

The simulation platform is a six-node ring SDN-based OBS network with one controller, and the simulation was run on a 64-bit Windows operating system containing eight physical cores from Intel Xeon(R) E5-2620 processors, and 7.64 GB RAM. The parameter settings of the simulation [24] are listed in **Table 2**.

Table 2. Simulation Setting

Parameter	Value
burst size	1 Mbit
transmission rate of data channel	8.6×10^9 bps
data channel propagation delay	5×10^{-5} s
propagation delay of data channel	1.3×10^9 bps
transmission rate of control channel	
control channel propagation delay	4.35×10^{-5} s
transmission rate between controller and OBS node	
offset time	1×10^{-6} s
the number of generated bursts	5×10^4

The numbers of busts in a burst-flow (i.e. the size of a burst-flow) are different in the three simulation scenarios. Once the number of bursts in the burst-flow buffer reaches the set burst-flow size, those bursts will be assembled together into a burst-flow. Each scenario ran ten times with different seeds. The simulation results given later are the average values of the ten simulation runs.

Cbench [25] is the standard tool for evaluating OpenFlow controller performances by emulating OpenFlow-enabled switches [18, 19, 20, 23]. Cbench generates Packet-In messages and sends them to the controller to measure the main performance indexes of the SDN controllers [18, 19, 20] including the number of Packet-In messages that a controller responds to per second, and the average latency of the controller to process a message. Therefore, in our work, the two main performance indexes are used for performance evaluation. In future, 5G scenario [26] will be considered for further enhanced user's quality of experience (QoE) [27].

5.2 Simulation Results and Analysis

Before describing the simulation scenarios, some constraints in the simulation are set:

- The controller should respond to each received Packet-In message;
- No collision and interference happens during burst propagation;
- Each OBS node can be both source and destination;
- Sending and receiving burst-flows in each OBS node are independent;
- There is no optical buffer and O/E conversion during the burst propagation.

(1) *Scenario 1: comparison with the classical OBS*

In the classical OBS networks, the burst scheduler makes a reservation for every burst. In order to make sure that the controller in the BFSM-based case makes a reservation for each burst, the size of the burst-flow buffer is equal to the size of one burst in this scenario.

Fig. 8 plots the burst loss rates between the BFSM-based OBS and classical OBS networks. In the simulation, the resource scheduling algorithm used in both cases was First-Fit (FF), a classical scheduling scheme of OBS networks. Their burst loss rates are almost the same when the offered load is low (between 0.1 and 0.3). However, if the offered load is higher than 0.3, the burst loss rate of the BFSM-based case is better than that of the classical OBS due to the central control of the SDN. It is worth noting that the advantage of the BFSM-based OBS is increased as the offered load increases. This proves that the SDN architecture provides network-wide scheduling ability and can avoid channel congestion better than the non-SDN case. Meanwhile, in the BFSM case, a burst should be discarded at the source node if its required resource cannot be reserved successfully at any node on its path, and the treatment can improve the channel resource utilization.

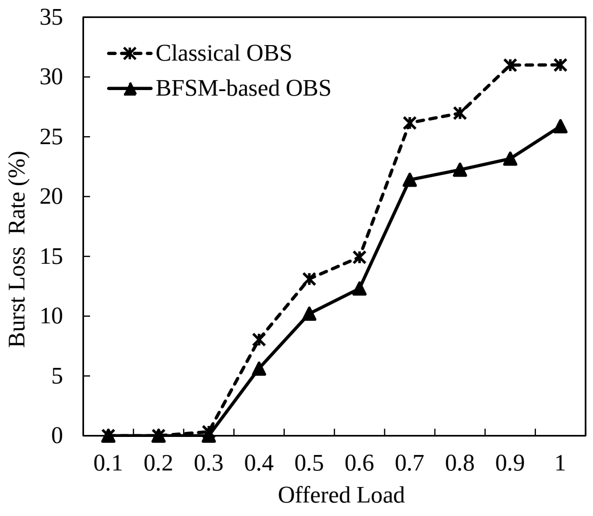


Fig. 8. Burst loss rates of two SDN-based and non-SDN OBS networks

Because there is no optical buffer and O/E conversion, the nodal processing delay in the intermediate node can be ignored. In the BFSM-based OBS, the average hop count and average end-to-end delay are 1.765 and 3.037×10^{-4} s, respectively. The numerical results indicate that the average end-to-end delay is almost equal to the product of the average hop multiplied by the delay of one hop ($=1.72 \times 10^{-4}$ s), and the BFSM hardly increases the end-to-end delay for bursts.

(2) *Scenario 2: comparison with the OTOCM-based OBS*

For Packet-In messages, the throughput is calculated by the number of Packet-In messages that a controller responds to per second, and the average latency is calculated by the round-trip time (RTT) from the source OBS edge-node to the controller. The size of the burst-flow in the BFSM case is, with uniform probability, between 20 and 40.

In the OTOCM-based OBS, the source node sends a Packet-In message to the controller after assembling one burst, and the control granularity of the resource reservation is the burst. However, in the BFSM-based OBS, the control granularity of the resource reservation is the burst-flow. FF is also the resource scheduling algorithm applied in the BFSM-based and OTOCM-based OBS cases in this scenario.

Table 3 shows that the maximum number of Packet-In messages in the BFSM-based OBS is three orders of magnitude less than that of the OTOCM.

Table 3. Maximum number of Packet-In messages (offered load = 1)

Control mechanism	Messages per second
OTOCM-based OBS	142931
BFSM-based OBS	4730

The response latencies are shown in **Table 4**. The BFSM-based OBS has the minimum response latency at 189.2 μs, and that of the OTOCM-based OBS is at 190 μs, because the BFSM-based OBS sends fewer Packet-In messages to the controller per second. That is, the fewer Packet-In messages that are processed, the lower the process time cost for the controller. Meanwhile, if the number of Packet-In messages is smaller, the bandwidth demand to the controller is lower.

Table 4. Response latency

Control mechanism	Minimum response latency	Average response latency
OTOCM-based OBS	189.995 μs	191.476 μs
BFSM-based OBS	189.268 μs	190.055 μs

In the BFSM-based OBS, the average number of bursts in a burst-flow is 30. As shown in **Table 4**, we can calculate that the average response latency of a burst, that is the average processing latency of the BFSM-based OBS, is 6.322 μs, about 30 times less than that of the OTOCM-based OBS (=190 μs).

Therefore, the BFSM-based OBS sends fewer Packet-In messages to the controller for processing and has lower average latency not only for one Packet-In message but also for one burst. In addition, the required bandwidth to the controller is decreased in the BFSM-based OBS. Compared with the OTOCM-based OBS network, the BFSM-based OBS network generates fewer Packet-In messages per second to which a controller responds, and the average latency of the controller to process each message is less, while the traffic between the nodes and the controller is less too.

(3) *Scenario 3: impact of the burst-flow size*

In this scenario, we consider the impact of the burst-flow size in the BFSM applying the RMCU. The number of bursts in each burst-flow is set to be 10 (the small size), 50 (the median size) or 500 (the large size), respectively.

Fig. 9 shows the contrasted burst loss rates of different sized burst-flows. A burst-flow with the larger size has the lower burst loss rate. When the offered load is higher than 0.6, more bursts in the small size burst-flow are discarded. We can see from **Fig. 9** that the larger the burst-flow size is, the fewer the number of bursts in the BFSM-based OBS networks are discarded. This means that the larger sized case can provide better burst transport service. Meanwhile, the larger sized burst-flow results in a smaller number of burst-flows, and fewer Packet-In messages generated by the source OBS node, which decreases the workload of the controller. That is to say, the time cost of the controller to process the messages received from the OBS nodes and the required bandwidth between the nodes and the controller will decrease.

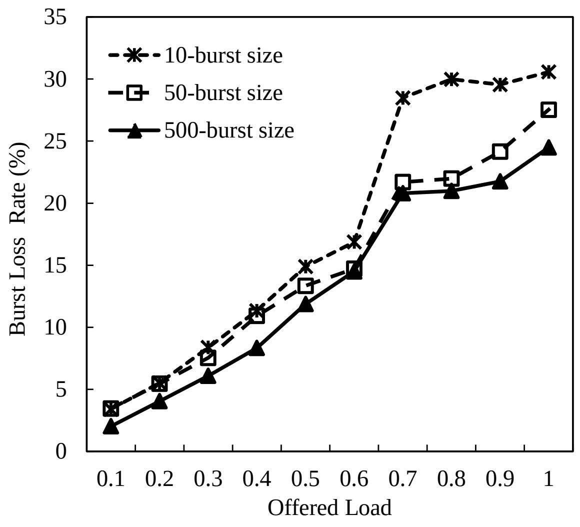


Fig. 9. Burst loss rates for different burst-flow sizes

6. Conclusion

To address the issue of the underutilization of network resource in SDN-based OBS networks, we presented a burst-flow scheduling mechanism BFSM in this paper. Our main contributions are: 1) extending the structure of the flow table, 2) adding a BBM application in the central controller, 3) giving the process and signaling procedure of the burst-flow, and 4) proposing a resource scheduling algorithm taking into account channel usage. The simulation results show that the proposed mechanism can provide good controller performance for the SDN-based network. In addition, when the burst-flow size is larger, more bursts can reach their destinations, and resource utilization is higher. Due to adding the burst-flow buffer in the source edge-node, extending the flow table, and increasing the burst-flow assembling time, the space and time consumption will increase to some extent. Therefore, how to define the sizes of burst-flow buffer and the flow information is our next work. Overall, BFSM offers optimal controller performance and a good solution of UCP for OBS networks.

References

[1] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: challenges and research advances," *IEEE Network*, vol. 28, no. 6, pp. 6-11, Nov. 2014. Article (CrossRef Link)
[2] T. Coutelen, H. Elbiaze, and B. Jaumard, "Performance comparison of OCS and OBS switching

paradigms," in *Proc. of 7th International Conference on Transparent Optical Networks*, vol. 1, pp. 212-215, Jul. 3-7, 2005. Article (CrossRef Link)

[3] C. Qiao and M. Yoo, "Optical burst switching (OBS) – a new paradigm for an optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69-84, 1999. Article (CrossRef Link)

[4] D. Zhang, L. Liu, L. Hong, H. Gou, T. Tsuritani, J. Wu, and I. Morita, "Experimental demonstration of OBS/WSON multi-Layer optical switched networks with an OpenFlow-based unified control plane," in *Proc. of 16th Optical Networks Design and Modelling (ONDM)*, pp. 1-6, Colchester, United Kingdom, Apr. 17-20, 2012. Article (CrossRef Link)

[5] A. N. Patel, P. N. Ji, and T. Wang, "QoS-aware optical burst switching in OpenFlow based software-defined optical networks," in *Proc. of 17th Optical Network Design and Modeling (ONDM)*, pp. 275-280, Brest, France, Apr. 16-19, 2013. Article (CrossRef Link)

[6] X. Ge, B. Yang, J. Ye, G. Mao, C.-X. Wang, and T. Han, "Spatial spectrum and energy efficiency of random cellular networks," *IEEE Transactions on Communications*, vol. 63, no. 3, pp. 1019 - 1030, Mar. 2015. Article (CrossRef Link)

[7] H. Guo, Y. Yin, T. Tsuritani, P. Huang, G. Zheng, J. Wu, T. Otani, M. Suzuki, and J. Lin, "Experimental demonstration of interworking GMPLS with OBS networks," in *Proc. of European Conf. on Optical Communications (ECOC)*, Sep. 16-20, pp. 1-2, Berlin, Germany, 2007. Article (CrossRef Link)

[8] L. Liu, D. Zhang, T. Tsuritani, R. Vilalta, R. Casellas, L. Hong, I. Morita, H. Guo, J. Wu, R. Martínez, and R. Muñoz, "Field trial of an OpenFlow-based unified control plane for multilayer multigranularity optical switching networks," *Lightwave Technology*, vol. 31, no. 4, pp. 506-514, Feb. 2013. Article (CrossRef Link)

[9] R. Martinez, R. Casellas, and R. Muñoz, "Experimental validation/evaluation of a GMPLS unified control plane in multi-layer (MPLS-TP/WSON) networks," in *Proc. of Optical Fiber Communication Conf. and Exposition (OFC/NFOEC)*, pp. 1-3, Los Angeles, USA, Mar. 4-8, 2012. Article (CrossRef Link)

[10] H. Farhady, H. Y. Lee, and A. Nakao, "Software-defined networking: A survey," *Computer Networks*, vol. 81, pp. 79-95, Apr. 2015. Article (CrossRef Link)

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, Apr. 2008. Article (CrossRef Link)

[12] D. King, A. Farrel, and N. Georgalas, "The role of SDN and NFV for flexible optical networks: Current status, challenges and opportunities," in *Proc. of 17th Int. Conf. on Transparent Optical Networks (ICTON)*, pp.1-6, Budapest, Hungary, Jul. 5-9, 2015. Article (CrossRef Link)

[13] D. King and A. Farrel, "A PCE-based architecture for application-based network operations," *IETF RFC 7491*, Mar. 2015. Article (CrossRef Link)

[14] A. Aguado, V. López, J. Marhuenda, ó. Dios, and J. Fernández-Palacios, "ABNO: a feasible SDN approach for multivendor IP and optical networks," *Journal of Optical Communications and Networking*, vol. 7, no. 2, pp. A356-A362, Feb. 2015. Article (CrossRef Link)

[15] ONF, "OpenFlow Switch Specification Version 1.0.0," Dec. 2009. Article (CrossRef Link)

[16] L. Liu, T. Tsuritani, I. Morita, H. Guo, and J. Wu, "OpenFlow-based wavelength path control in transparent optical networks: A proof-of-concept demonstration," in *Proc. of 37th European Conf. on Optical Communications (ECOC)*, Geneva, Switzerland, Sep. 18-22, 2011. Article (CrossRef Link)

[17] L. Liu, T. Tsuritani, I. Morita, H. Guo, and J. Wu, "Experimental validation and performance evaluation of OpenFlow-based wavelength path control in transparent optical networks," *Optics Express*, vol. 19, no. 27, pp. 26578-26593, Dec. 2011. Article (CrossRef Link)

[18] D. Erickson, "The Beacon OpenFlow controller," in *Proc. of ACM SIGCOMM workshop on Hot Topics in Software Defined Networking (HOT SDN)*, pp. 13-18, Hong Kong, China, Aug. 18, 2013. Article (CrossRef Link)

[19] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. of the 2nd USENIX conf. on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, San Jose, USA,

Apr. 24, 2012. Article (CrossRef Link)

[20] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advance study of SDN/OpenFlow controllers," in *Proc. of the 9th Central & Eastern European Software Engineering Conf. in Russia (CEE-SECR)*, Moscow, Russia, October, 2013. Article (CrossRef Link)

[21] A. Shalimov and R. Smeliansky, "On bringing software engineering to computer networks with software defined networking," in *Proc. of Spring/Summer Young Researchers' Colloqium on Software Engineering (SYRCoSE)*, pp. 17-22, Kazan. Russia, May 2013. Article (CrossRef Link)

[22] P. Ivashchenko, A. Shalimov, and R. Smeliansky, "High performance in-kernel SDN/OpenFlow controller," in *Proc. of Open Networking Summit Research Track*, USENIX, Santa Clara, USA, March, 2014. Article (CrossRef Link)

[23] D. Turull, M. Hidell, and P. Sjodin, "Performance evaluation of OpenFlow controllers for network virtualization," in *Proc. of 14th High Performance Switching and Routing (HPSR)*, pp.50-56, Vancouver, Canada, Jul. 1-4, 2014. Article (CrossRef Link)

[24] X. Ge, G. Zhu, and Y. Zhu, "On the testing for alpha-stable distributions of network traffic," *Computer Communications*, vol.27, no.5, pp.447-457, Mar. 2004. Article (CrossRef Link)

[25] Cbench, "Cbench Controller Benchmarker," [cited 2015 June 9]. Article (CrossRef Link)

[26] K. Zheng, L. Zhao, J. Mei, M. Dohler, W. Xiang, and Y. Peng, "10 Gb/s HetSNets with millimeter-wave communications: access and networking-challenges and protocols," *IEEE Communications Magazine*, vol. 53, no.1, pp. 222-231, Jan. 2015. Article (CrossRef Link)

[27] K. Zheng, X. Zhang, Q. Zheng, W. Xiang, and L. Hanzo, "Quality-of-experience assessment and its application to video services in LTE networks," *IEEE Wireless Communications*, vol. 22, no.1, pp. 70-78, Feb. 2015. Article (CrossRef Link)

Wan Tang received the B.S. and M.S. degrees in Computer Application Technology from South Central University for Nationalities, Wuhan, China, in 1995 and 2001, respectively, and received a Ph.D. degree in Communication and Information Systems from Wuhan University, China, in 2009. She is currently an associate professor in the College of Computer Science of South-Central University for Nationalities. From 2001 to 2002, she worked as a visiting scholar at the Department of Computer Engineering, Chonbuk National University, South Korea. Furthermore, from 2012 to 2013, she worked as a visiting scholar at the Department of Computer Science and Engineering, SUNY at Buffalo, USA. Her research interests include protocols for optical/wireless networks, software defined networking, network security, etc.

Fan Chen is an M.S. student at the College of Computer Science, South-Central University for Nationalities. He received a B.S. degree in the Department of Computer Science from Donghu College, Wuhan University in 2013. His research interests lie in optical switching technologies, software-defined networking, data center networks, etc.

Min Chen is a professor in School of Computer Science and Technology at Huazhong University of Science and Technology (HUST). He is Chair of IEEE Computer Society (CS) Special Technical Communities (STC) on Big Data. He was an assistant professor in School of Computer Science and Engineering at Seoul National University (SNU) from Sep. 2009 to Feb. 2012. He was R&D director at Confederal Network Inc. for half a year. He worked as a Post-Doctoral Fellow in Department of Electrical and Computer Engineering at University of British Columbia (UBC) for three years. Before joining UBC, he was a Post-Doctoral Fellow at SNU for one and half years. He received Best Paper Award from IEEE ICC 2012, and Best Paper Runner-up Award from QShine 2008. He serves as editor or associate editor forInformation Sciences, Wireless Communications and Mobile Computing, IET Communications, IET Networks, Wiley I. J. of Security and Communication Networks, Journal of Internet Technology, KSII Trans. Internet and Information Systems, International Journal of Sensor Networks. He is managing editor for IJAACS and IJART. He is a Guest Editor for IEEE Network, IEEE Wireless Communications Magazine, etc. He is Co-Chair of IEEE ICC 2012-Communications Theory Symposium, and Co-Chair of IEEE ICC 2013-Wireless Networks Symposium. He is General Co-Chair for the 12th IEEE International Conference on Computer and Information Technology (IEEE CIT-2012) and Mobimedia 2015. He is General Vice Chair foTridentcom 2014. He is Keynote Speaker for CyberC 2012, Mobiquitous 2012 and Cloudcomp 2015. He is a TPC member for IEEE INFOCOM 2014. He has more than 260 paper publications, including 100+ SCI papers, 50+ IEEE Trans./Journal papers, 6 ISI highly cited papers and 1 hot paper. He has published a book on IoT: OPNET IoT Simulation (2015) with HUST Presss, and a book on big data: Big Data Related Technologies (2014) with Springer Series in Computer Science. His Google Scholars Citations reached 5,200+ with an h-index of 34. His top paper was cited 640 times, while his top book was cited 420 times as of Aug 2015. He is an IEEE Senior Member since 2009. His research focuses on Internet of Things, Mobile Cloud, Body Area Networks, Emotion-aware Computing, Healthcare Big Data, Cyber Physical Systems, and Robotics, etc.

Guo Liu is an M.S. student at the College of Computer Science, South-Central University for Nationalities. He received a B.S. degree in the College of Computer Science from South-Central University for Nationalities in 2013. His research interests lie in software-defined networking, traffic engineering in data center networks, etc.