IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 13, 2018, accepted May 21, 2018, date of publication May 30, 2018, date of current version June 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2842058

Implementation of Multipath Network Virtualization With SDN and NFV

QINGTIAN WANG, (Student Member, IEEE), GUOCHU SHOU, YAQIONG LIU, (Member, IEEE), YIHONG HU, ZHIGANG GUO, AND WEI CHANG, (Student Member, IEEE)

Beijing Key Laboratory of Network System Architecture and Convergence, School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Guochu Shou (gcshou@bupt.edu.cn)

This work was supported in part by National Nature Science Foundation of China under Grant 61471053 and in part by the Fundamental Research Funds for the Central Universities under Grant 2018RC03.

ABSTRACT Multipath networks have been extensively studied to improve throughput and alleviate network congestion. However, multipath networks, with the exception of equal-cost multi-path networks, are difficult to implement because they schedule network resources at cross layers and require complex signaling mechanisms to monitor network status (e.g., round trip time, bandwidths, and link state). Virtualization, in contrast, provides network isolation with protocols and resources and simplifies the process of coupling the interaction between signaling mechanisms, which makes multipath implementation practical. This paper proposes a multipath network virtualization scheme that implements software-defined networking (SDN) and network function virtualization (NFV). Multipath networks can be deployed in the virtualized environment. The proposed scheme attains a summary of network resources, such as network topology and link bandwidth, and can schedule these resources for selecting and spreading flow over multiple paths. In addition, virtualization provides computing and storage resources for flow splitting, tag adding, and packet reordering, among other applications. Specifically, the tags are designed for forwarding and packet reordering. Our experiment builds the platform based on open platform for NFV and SDN on the practice testbed. To verify its functionalities, we first evaluate the overhead of each component and then implement and compare five state-of-the-art multipath models on the proposed platform. Our scheme is compatible with these multipath models. Moreover, the experimental results show that the model with flow splitting has superior performance with respect to load balancing.

INDEX TERMS Multipath, virtualization, NFV, SDN, flow splitting, packet reordering.

I. INTRODUCTION

Recent online applications, such as social networking, high-definition video streaming, and augmented reality (AR),have high demands for throughput, bandwidth, quality of service (QoS) and end-to-end delay. For example, AR applications require a minimum bandwidth of 10Mb/s to ensure that the video has sufficient information and rely on latencies under 20ms [1]. However, most internet communications are established over a single path mechanism, which is subject to constraints on throughput, load balancing and delay, making it difficult to meet the expected communication performance. Therefore, a multipath network mechanism is required to spread the communication traffic over multiple paths. Compared to a single path mechanism, a multi-path approach increases the available bandwidth, ensures high-quality network services, and guarantees quality of experience (QoE) [2], [3].

Although multipath models have been studied for years, many multipath models are difficult to implement in practice. The only widely used multipath routing model is the Equal-Cost Multi-Path (ECMP) model [10], which spreads traffic over multiple equal cost paths using hash functions. However, ECMP cannot dynamically forward the flow according to the changing network status, which leads to load unbalancing. Other multipath network models, such as FLARE [4], LBPF [5], LDM [6], are rarely implemented in practice because these models relay on the accurate network status. For example, the FLARE model requires information about packet inter-arrival time and path delay; the LBPF model counts the lengths of the input queues and each packet's

arrival time; the LDM model is based on path utilization and hop-count. This information, however, is difficult to obtain due to the complex signaling mechanisms in the traditional multipath network environment. Furthermore, heterogeneous switches/routers in the traditional multipath network lead to packets that are out-of-order because the different hardware-based devices cause delays in changing the forwarding process. Moreover, the computing and storage capabilities of these switches and routers are not sufficient to satisfy the multipath network requirements for flow splitting and packet reordering. The infrastructure of traditional networks is controlled by a large group of network operators, whereby the complicated signaling mechanisms and protocols obtain the available bandwidth and congestion status of the path. Thus, it is difficult to construct a multipath network.

To implement multipath routing in practice and solve the above-described issues, one potential solution is to deploy virtualized multipath network with software-defined networking (SDN) and network function virtualization (NFV). Network virtualization abstracts the physical layer resources to the virtual layer and forms a global view of network resources. The control layer can manage network resources efficiently without complex signaling mechanisms at the cross layers. We now briefly introduce SDN and NFV. SDN is a network paradigm that been widely adopted as a mean to enable network virtualization [7]. This approach decouples the control plane and data plane and consolidates the control functions into a controller. The network controller determines the path of the network flow across a network of switches using the OpenFlow protocol. Additionally, SDN has a simple network design, making it easy to deploy with the multipath network mechanism. NFV, in contrast, aims to address the issues with traditional hardware-based networks by virtualizing the network functions and solves the problems that traditional network switches/routers have difficulty managing or for which these components require physical changes [8]. Furthermore, NFV makes it easy to expand network size, reduce energy consumption, and decrease the waiting period for new service deployment. NFV standards have been well developed by various committees, such as European Telecommunications Standard Institute (ETSI), Huawei and Ericsson [9].

In this paper, we propose a multipath virtualization network implementation scheme that takes advantage of both SDN and NFV. Specifically, we apply SDN for its efficient management, control separation and global network resource scheduling. Regarding computing and storage resources, we build upon NFV by implementing network functions in servers. In the proposed scheme, the control plane and data plane are designed to implement a multipath network. In the experiment, we build the test platform based on Open Platform for NFV (OPNFV) and SDN. We evaluate state-of-the-art multipath network models that forward traffic based on flow splitting, network status and hashing. The results show that flow splitting alleviates network congestion on a multipath network, and the path selected with network status facilitates load balancing. This paper makes the following contributions:

- We propose a multipath network virtualization scheme that contains control plane and data plane, and five state-of-art multipath models are evaluated in the scheme.
- We propose a flexible flow splitting mechanism in the virtualized multipath network environment, which has not been achieved in the traditional hardware-based multipath network.
- To prevent the flow units from the same flow being forwarded to one path, we presented a tag-based forwarding mechanism.
- We conduct a testbed based on OPNFV in practice.

The rest of paper is organized as follows. In Section II, we review the related literature in multipath network implementations. Section III introduces the implementation of multipath network virtualization and the corresponding components. We present our experiments and analysis in Section IV. Section V concludes this paper.

II. RELATED WORK

In this section, we first review the studies of multipath network in the traditional network, i.e., IP and wireless networks, where packets are forwarded *via* IP protocols at cross layers. We then outline the network virtualization schemes and the attempts to apply multipath with SDN or NFV.

In the traditional IP network, multipath networks can be established using several different approaches. Prabhavat *et al.* [10] discuss five multipath cases in different networks scenarios: multipath routing at source node with multiple interfaces, multipath routing at gateway node with multiple interfaces, multipath at source node with multiple wireless interfaces, inverse multiplexing over multiple parallel point-to-point narrowband links and multipath routing over wireless mesh network or mobile ad hoc. Singh *et al.* [11] point that multipath networking is constructed at cross layers in the traditional network (i.e., IP network). Specifically, this approach establishes an end-to-end connection at the transport layer, forward flows or packets over the network layer, and reserves link bandwidths at the link layer. De Coninck *et al.* [12] implement multipath Transmission Control Protocol (TCP) on Android smartphones and analyze how popular smartphone applications interact with multipath TCP under different network conditions with both WiFi and cellular networks.

Regarding multipath network virtualization schemes. He *et al.* [13] applied network virtualization to remove the differences between the infrastructures of heterogeneous networks to consider fiber-wireless hybrid access networks (FiWi). The authors proposed a viable solution to establish multipath access in the FiWi network through the flexible use of virtual networks with network virtualization. There are also works that consider multipath network in an SDN architecture, such as [14]–[17]. Subedi *et al.* [15] presented an adaptive multipath routing

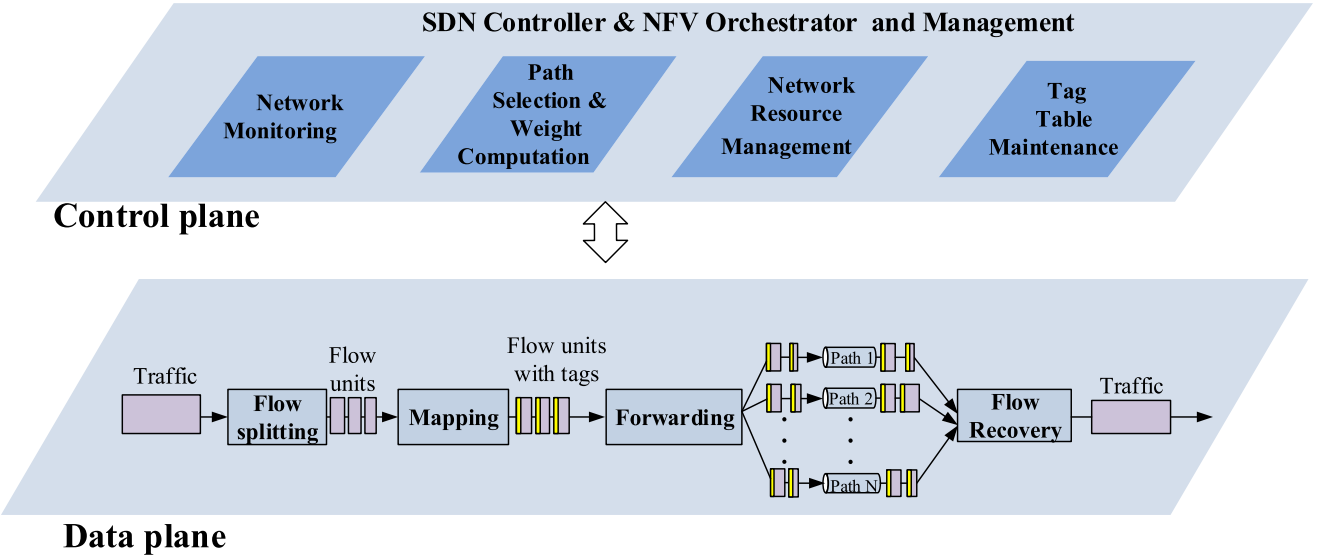

FIGURE 1. Illustration of implementation of multipath network virtualization.

architecture in physical layer networks. This architecture adapts to link and path failures to compute multiple paths. To avoid out-of-order delivery, Subedi *et al.* [15] also proposed selecting weight probability with caching instead of Weighted Round Robin (WRR) and per-packet link selection. Banfi *et al.* [16] proposed a multipath SDN architecture to select alternate paths in parallel to aggregate capacity and provide higher throughput by enabling the use of multiple paths in a physical or network layer topology. They discussed a solution using path selection, packet scheduling, and packet reordering under a multipath SDN architecture and verified that the capacity aggregation benefits provided by the architecture are similar to what Multi-Path TCP (MPTCP) achieves. Basit *et al.* [17] proposed a novel approach to build a conducive inter-networking ambiance for future data-centric applications. The approach employed a cross-layer coordinated multipath forwarding scheme attached to SDN to ensure improved reliability and throughput.

NFV releases network device virtualization. Therefore, interests has been shown in studies on implementing technology to enable multipath network. Nguyen *et al.* [18] developed a multipath routing solution with NFV-based routing. The objective was to minimize the maximum link utilization and the network resources utilization efficiency. Pham and Pham [19] analyzed and optimized the load balancing problem by using multipath routing in NFV. The authors also verified that the responsive ability of a network system was improved based on NFV.

The previous studies focused on multipath network configurations in a traditional IP network or used SDN for its flexible forwarding and rapid deployment abilities to address path selection and improve network throughput and reliability. These studies also implemented NFV to optimize link utilization. However, the previous did not consider jointly implementing SDN and NFV to establish a multipath network architecture. Therefore, the present paper proposes a multipath network virtualization implementation scheme that considers the challenge of implementing a multipath network in an IP network architecture.

III. THE SCHEME OF IMPLEMENTATION OF MULTIPATH NETWORK VIRTUALIZATION

In this section, we present the proposed multipath network virtualization implementation scheme [20], as shown in Fig.1. This scheme contains a control plane and data plane. The control plane consists of a Network Monitoring unit to monitor the network bandwidth and delay; a Path Selection & Weight Computation component to choose the paths; a Network Resource Management component to manage the network capabilities; and a Tag Table Maintenance unit to map the multipath selection to the protocol that is implemented on the devices, e.g., OpenFlow protocol. In the data plane, the traffic is first split into flow units through Flow splitting, and then multipath tags are added *via* Mapping. The multiple paths' tags are determined by the control plane that specifies the flow paths. The flow units are then forwarded to the designated paths from the Forwarding components corresponding to their tags. Finally, at the destination node locating, flow recovery restores the flow units from the multiple paths. Next, we discuss these components and their functions in detail.

A. CONTROL PLANE

1) NETWORK MONITORING

In SDN, the controller does not need to discover the nodes, i.e., OpenFlow switches, actively, because the nodes are connected to the controller and voluntarily announce their existence. Moreover, the controller supports the LLDP [21] based link discovery. However, the controller does not monitor the network status, such as bandwidth and delay.

Therefore, the controller does not obtain the network status for planning flow paths. Thus, the controller must monitor the network status using a network monitoring component.

The Network Monitoring component monitors the delay *via* Round-Trip Time (RTT), which is derived from the well-known Internet Control Message Protocol (ICMP). However, additional resources such as Virtual Machines (VMs) are required on each node to enable the ICMP. The available bandwidth can be obtained based on the method proposed in [22]. Specifically, the current load $b_i(t)$of link at time t can be polled using the counters in the node, which measure bandwidth using flow counters as follows,

$$b_i(t) = \frac{n_i(t) - n_i(t-T)}{T},$$

where $n_i(t)$ is the counter value and T is the polling period. The available bandwidth a_i(t) is calculated by total capacity minus current load, i.e.,

$$a_i(t) = c_i - b_i(t),$$

where c_i is total capacities of a link. The Network Monitoring component periodically updates and stores the information about the network topology, delay and bandwidth. This information is also shared with other components in the control plane.

2) NETWORK RESOURCE MANAGEMENT

In the traditional IP or the SDN-only network, the computing and storage capabilities are subject to the hardware limitations of the switches and routers, which makes it difficult to process flow. Therefore, we propose replacing the physical networking devices on the servers with a powerful Central Processing Unit (CPU) and hard-drives that adapting NFV. As suggested in [23], the NFV uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks to establish network connections. Thereby, the network resource management component is capable of scheduling the network bandwidth resources using the SDN controller and computing and storage resources using NFV management and orchestration (MANO). The SDN controller schedules the network resources in two steps.

- *Step-1:* Based on the bandwidth and delay between the nodes, which are stored in the network monitoring component, the SDN controller selects multiple paths to construct a multipath network and allocates bandwidth to each path using Path Selection & Weight Computation.
- *Step-2:* The SDN controller informs the nodes how to distribute the flow and forward to the paths *via* packet-out message.

The NFV MANO contains a Virtualized Infrastructure Manager (VIM) centralized manages the computing and storage resources that distributed at all the independent nodes in data plane. To do this, VIM creates Virtual Machines(VMs) according to the requirements of computing and storage resources from a functional component, and then the component is carried in a VM.

3) PATH SELECTION & WEIGHT COMPUTATION

This component selects the paths for the flow whenever a new connection is established from the source to the destination node in the data plane. Based on the network topology information obtained from the network monitoring components, we apply Yen's K shortest paths (KSP) [24] algorithm to calculate the disjoint paths for this flow from its source to the destination node. Next, according to the application specified requirements, a path selection algorithm can be embedded to further select one or multiple paths to forward the flow. For example, by embedding the LDM [6] path selection algorithm, the path with larger available bandwidth and shorter delay will be selected with high priority, thus guaranteeing the QoS requirement.

After the path selection, challenges remain for path weight computation. When a flow is split into several flow units, the path weight is that flow unit in proportion to the flow, specifically, determining the number of flow units distributed in each path. Path weight computation can be embedded with state-of-art algorithms. For example, by embedding weighted multipath routing (WMR) [25], the path with higher available bandwidth will be allocated with higher weight, thus alleviating network congestion.

The selected paths and weight of each path are sent to the data plane of the source node *via packet-out* message. We also separate background and requested flow using *set-queue* action in the OpenFlow table which assigns a queue id for each packet. The queue id determines which queue is attached to a port used for forwarding packets. Therefore, the background traffic and requested flow are separated into different queues (or ports) and prevent packet loss from the requested flow if the background traffic uses significant bandwidth on one link.

4) TAG TABLE MAINTENANCE

Tag table maintenance stores the information that identifies the specific path that a flow unit forward to and the order rule for flow units and packets (e.g., sorting in ascending order). Tags help the mapping component in the data plane to determine the flow units forwarding. Tags are derived from the Path Selection & Weight Computation. A new tag is generated when a connection request arrives. Moreover, the relationship between tags and original header of a flow is sent to Flow Splitting and Flow Recovery components of data plane.

The design of the tag is based on commonly observed phenomena of flows: (*i*) a flow is forwarded over multiple paths, (*ii*) different flows with different priorities, and (*iii*) a serial number is applied to represent the order of flow units. Therefore, the tag of each flow unit indicates where that unit originates from, its priority and to which path it should be forwarded.

TABLE 1. Tag format.

Tag Identifier	Flow ID	Path ID	Packet Sequence Number	Priority	Time to live (optional)

The tag format is shown in Table 1. The format contains the Tag Identifier, Flow ID, Path ID, Packet Sequence Number, Priority and an optional Time to Live field. In this paper, the tag identifier is a custom binary character to identify the tag. In particular, the size of Flow ID, Path ID, Packet Sequence Number are various according to the number of flows and network scale. Flow ID is used for distinguishing flows. Path ID represents the path that the flow unit be forwarded to. Packet Sequence Number(PSN) records the order of packets. Priority represents the forwarding order of the flow unit. Time to Live is an optional that limits the life of a tag in the control plane.

B. DATA PLANE

1) FLOW SPLITTING

The traditional method to split flows is called the traffic splitting, which is based on the smallest splitting unit. There are three levels of traffic splitting on the packet-level, flow-level and subflow-level, respectively [10]. In the state-of-art traffic distributing approaches, hashing is based on flow-level, and Round Robin (RR) is based on packet-level. In subflow-level traffic splitting, packets going to the same destination are split into subflows. The packets of the subflow have the same destination, but packets heading to the same destination may be carried in different subflows. However, in these traffic splitting methods, the larger flow may be forwarded to a higher load path after traffic splitting, which causes the length of the queue to increase and results in load unbalancing. Therefore, we propose a novel flow splitting approach based on NFV that splits the flow into several flow units of various sizes. These flow units can be forwarded into the path corresponding to its load capacity, which balances the queue and the path load. In the proposed virtualization scheme, flow splitting is implemented in the source node, where the number and size of the traffic units are determined using the Path Selection & Weight Computation component in the control plane. The procedure of the proposed flow splitting approach is as follows.

- *Step-1:* When the mixed traffic arrives at the source node, it is buffered and separated into flows according to their source and destination addresses.
- *Step-2:* The flow splitting component distinguishes each flow and sends its size to the control plane.
- *Step-3:* In the control plane, the Path Selection & Weight Computation component calculates the number of selected paths and their weight. Then, the control plane sends a *packet-out* message to the source node containing the path tag and their weights.
- *Step-4:* After receiving the *packet-out* message, the Flow Splitting determines the first and last packet of each flow units according to the calculated path weight. Note that, the weight can vary.
- *Step-5:* Then, the flow units are sent to the Mapping component to wait for the tags to be added.

2) MAPPING

Mapping distinguishes flow units by matching the path tags determined by the control plane. The original header of packets in the flow unit is replaced with a simplified tag embedded in the OpenFlow header. To this end, the OpenFlow 1.3.1 specification for SDN allows additional headers by using the *push* operation [26]. For each flow, a set of tags are added corresponding to the flow units, which are removed when the flow arrives at the destination node.

3) FORWARDING

In the traditional multipath network, flow units derived from the same flow share an identical source and destination address and are therefore forwarded to the same path. This constraint makes the multipath network impractical. To solve this issue, we implement a tag-based forwarding approach. According to this approach, the forwarding component detects the priority of the flow units based on the Priority field in the tag header and forwards them accordingly. Each flow unit is forwarded into the path corresponding to its Path ID. Therefore, in the proposed flow unit forwarding mechanism, the flow units are forwarded based on their tag instead of their destination address, allowing flow units to reach the destination using different paths.

4) FLOW RECOVERY

This component is employed in the destination node and recovers the flow units to form the original flow based on the Path ID of the tags. Specifically, it receives the relationship between tags and original header from the Tag Table Maintenance of control plane via the packet-out message. It then stores the said relationship into a storage space. Upon receiving a flow unit, it then search and replace the flow unit header to the one that stored in the storage space. Next, the flow recovery component notifies the tag table maintenance component to remove the stored path via the packet-in message and tag information regarding the recovered flow.

By using the tags to recovery flows, this component also solves the packet reordering issue, where the different delay of the forwarding path may cause the recovered flow to be out-of-order and unreadable. As shown in [27], in flow-level traffic splitting, if the splitting percentages change or a path fails, a flow is likely to be forwarded into a different path than before, which causes out-of-order packets during the transition. For the same reason, in packet-level traffic

splitting, packets arriving may out-of-order due to the various paths' delay in multiple paths.

We now present our tag based packets reordering in the flow recovery component. As shown in Algorithm 1, after the destination node receives flow units, they are stored in the same buffer. Then, the flow recovery component requests the order rule of flow units and packets *via* packet-in message from the Tag Table Maintenance. Next, the Tag Table Maintenance component replies *via* a packet-out message with the order of Path ID and Packet Sequence Number(PSN). Then, the flow units with the same Flow ID are sent to the one buffer. Subsequently, the packets of each flow unit are reordered by the Packet Sequence Number. Finally, the flow is recovered by sorting the flow units by Path ID.

Algorithm 1 Packet Reordering

1: Flow recovery requests the order rule of traffic units as well as packets from Tag Table Maintenance and receives the reply.
2: **while** *destination node receive the flow units* **do**
3: *send them to the buffer*
4: **end while**
5: **while** *flow recovery check the tags of flow units* **do**
6: **if** *Flow ID* $= N$ **then**
7: *send the packets to the buffer of* $N-th$ *path*
8: **end if**
9: *reordering the packets by the PSN*
10: **end while**
11: **while** *packets have been reordering* **do**
12: *recovery the flow by Path ID*
13: **end while**
14: *Send the traffic to destination network*

IV. EXPERIMENTS AND PERFORMANCE EVALUATION

In this section, we first build the experiment platform on the practice testbed and evaluate the functional components of data plane, and then implement the state-of-art multipath models on the built platform. We then analyze of the models' comparison results.

A. EXPERIMENTAL ENVIRONMENT

Recall that we conducted multipath network virtualization experimental platform using SDN and NFV technologies. Specifically, we used OpenFlow switches that support the OpenFlow 1.3.1 protocol for flows/packets forwarding. Three industrial-level x86 servers were deployed with Open Platform for NFV (OPNFV), one of which acted as the control node with an SDN controller and NFV orchestrator, while the other two servers were computing nodes to simulate source and destination nodes. The devices were connected *via* fiber links. An OPNFV infrastructure platform typically consists of an OpenStack that manages VMs, and an SDN controller that manages the control plane and distributes commands to the data plan [28]. On each node, four VMs were created for flow splitting, mapping, forwarding and recovery, respectively. In addition, the four components are implemented with C++. On the computing nodes, VMs ran *Iperf* Client and Server were established to emulate the traffic, and additional VMs running OpenVSwitch (OVS) were deployed as needed for forwarding. In the experiments, we deployed an additional six severs as the source and destination nodes, three of which generated random background traffic. The configuration parameters for the experiment are shown in Table 2.

TABLE 2. Hardware and software configuration.

Device	Configuration
Server	Operating system:ubuntu 14.04 Intel Xeon processor @2.9GHz 64GB RAM Virtual Switch:OpenvSwitch SDN Controller: OpenDayLight(Lithium-SR3) NFV platform:OPNFV(Arno)
Switch	Switch model: Centec V350 OpenFlow specification 1.3.1 L2 to L4 complete matching fields 8*1Gbase-T 12*10Gbase-X

Multipath models were embedded in the SDN controller. After mapping, the SDN controller sent OpenFlow tables to the source node, destination node and forwarding nodes. The experimental topology is shown in Figure 2. In this topology, *FU* denotes the split Flow Unit. We use *OF m,n* to represent the *n-th* switch in *m-th* path, for example, *OF 1,1* represents the first switch in the first path. We use *S,n* and *D,n* to represent the *n-th* port of the source node and the destination node, respectively. The source and destination nodes are specified in the experiment.

We conduct two sets of experiments, we first evaluate the overhead of each component, and we then implement and compare five state-of-art multipath models in our proposed scheme. In order to evaluate the overhead of each component, we first evaluated the processing time of Flow Splitting, Mapping and Flow Recovery components. We then show the round trip time (RTT) of Forwarding component. In particular, a Data Plane Development Kit (DPDK) [33] was combined in the forwarding component to reduce the forwarding time. We varied the size of packet and evaluated the RTT between source node and destination node. Flow recovery component was evaluated by varying the out-of-order ratio. In general, there were 30,000 packets in a flow with 900Mb in our experiments. Therefore, we test 30,000 packets in the experiment and the out-of-order ratio varied from 10%-90%. Specifically, the out-of-ratio is determined as per [32].

To verify the functionalities of the proposed multipath network virtualization implementation scheme, we implemented five state-of-art multipath models in our experiment: Direct Hashing (DH) [29], Primary Number Modulo-N Load Balance (PMN-LB) [30], Load Balancing for Parallel Forwarding (LBPF) [5], Load Distribution over Multipath (LDM) [6] and Weighted Multipath Routing (WMR) [25]. Based on the forwarding mechanism,

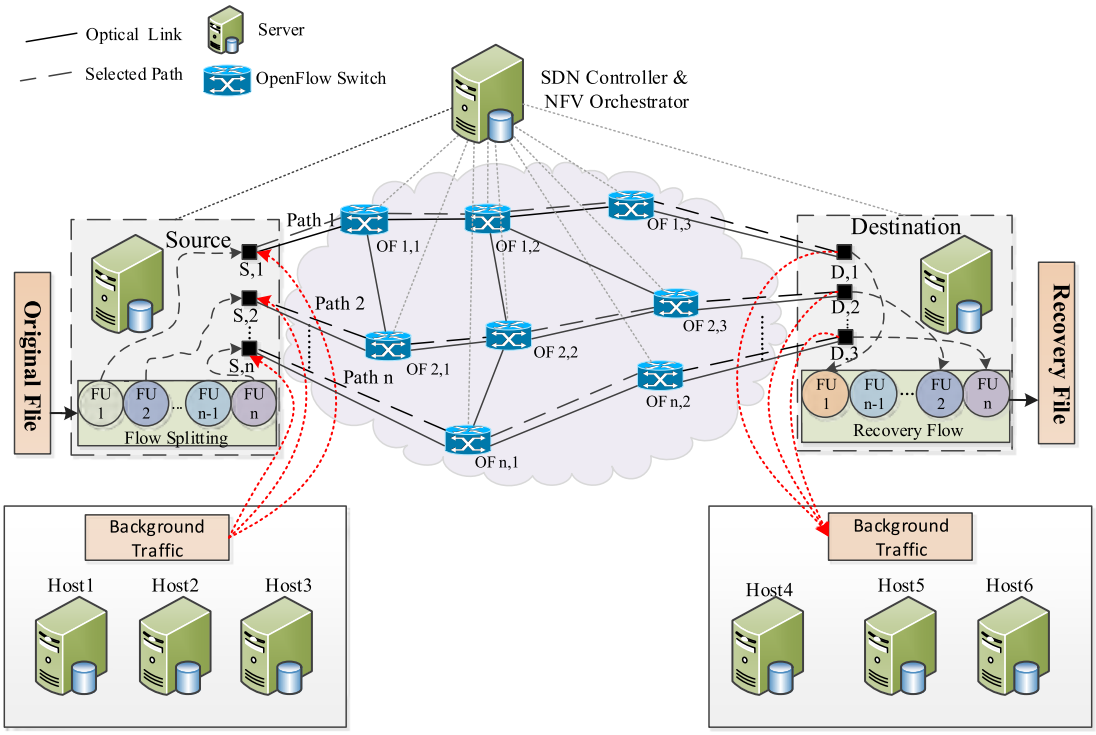

FIGURE 2. Experimental topology.

these models were classified into three categories: (1) hash, i.e., DH and PMN-LB; (2) network conditions, i.e., LBPF and LDM; and (3) flow splitting, i.e., WMR. We now briefly introduce the process of these models:

- Direct Hashing (DH)
  The DH forwards the flow based on modulo-k hash algorithm, where k is the number of paths, and each flow is assigned to a path according to a hashing value H mod k.
- Primary Number Modulo-N Load Balance (PMN-LB)
  The PMN-LB assigns incoming flows to the selected next-hop based on hash algorithm. The hashed value of flow is divided by the number of multiple paths. If a next-hop points the router is remainder, this next-hop selected to forward packet. The remainder is the index for a array of next-hops excluded next-hops not existing. Otherwise, the prime number not less than the number of available multiple paths divides the hashed value and a next-hop is selected using this remainder. If the both above calculations are failed, the hashed value is then divided by the number of available multiple paths and repeat.
- Load Balancing for Parallel Forwarding (LBPF)
  The LBPF selects the path for a flow according to a hash function when the system has a low load. If the system has less available bandwidth, the load adaptor may decide to override the decisions of the hash splitter that uses the packet's destination address as input to the hash function. The load adapter checks each passing packet to identify flow, and sets the high-rate flows to be forwarded to the path with the shortest queue.
- Load Distribution over Multipath (LDM)
  The objective of LDM is to enhance the network utilization and the network performance by adaptively splitting traffic load among multiple paths. The routing forward the traffic at the flow forwarding units, and the traffic proportioning reflects both the length and the utilization of a path.
- Weighted Multipath Routing Algorithm
  This algorithm spreads the elephant flows which take up a large amount of network bandwidth across multiple links based on the dynamically flow splitting to reduce network congestion. The algorithm calculates the path weight based on link load, where the higher link load results in a lower path weight.

In our experiments, the input traffic was forwarded over a multipath network with three disjoint paths, and each path had an input queue with 2Gigabit(Gb) buffer size. To implement these models in the proposed experiment platform with identical process, we assumed that if the available bandwidth satisfied the requested demand, then the request was accepted. Otherwise, the traffic was sent to a buffer to wait to be forwarded. Each fiber had a link capacity of 1Gb. The available bandwidth of each link was influenced by the background traffic. Particularly, as in [31], there were three types of background traffic packet with size of 44 Bytes, 576 Bytes and 1500 Bytes, and the distribution of these packets are 50%, 25% and 25%, respectively.

To simulate mixed traffic, we transmitted eight files at a time with sizes from 400Mb to 1.3Gb. Initially, the available bandwidth of the three paths are constantly 200Mbps, 300Mbps and 400Mbps. After the size of traffic changed the load of the multipath network, the available bandwidth of these paths changed with a fixed ratio, i.e., 2 to 3 to 4. We use *Wireshark* to capture all the packets from the traffic and to calculate the delay and link utilization. The results were generated from fifty runs. We observed the following metrics:

Multipath Network Delay represents the maximum delay over multiple paths in the multipath network. The delay includes the time consumed for transmission and queuing.

Delay standard deviation shows the variation of the delay over multiple paths.

Multipath network utilization is the transmission rate divided the total available bandwidth in the multipath network, where the rate is determined by the total files size divided by the transmission delay of multipath network.

B. OVERHEAD OF COMPONENTS

In our experiments, a flow had a size ranging from 400Megabits(Mb) to 1300Mb and was split into three flow units. We evaluated the processing time of flow splitting and mapping. we also demonstrated the processing time of a 900Mb flow that been split into three to nine flow units with mapping tags. From Fig. 3, we can see that the processing time of flow splitting and mapping is increasing from 13.8ms when the size of flow is 400Mb to 46.6ms when the size of flow is 1300Mb. This is because the computing ability of cpu, writing and reading ability of interface of these two components are stable, every bit has almost same processing time. Thus their processing time is increase with the size of flow. Fig. 4 shows that the processing time is around 32.5ms with the increasing of flow unit number from three to nine. This is because, the size of flow is a constant regardless the number of flow unit it been split to. Therefore, additional processing time occurs in these two components with increasing size of flow.

Fig. 5 shows that RTT is increasing from 0.26ms when the size of packet is 64Byte to 0.48ms when the size of packet is 512Byte. This is because packets are serial processing in the Forwarding component. Therefore, the RTT is increased with larger packet size. We noted there is a fluctuation from 64Byte to 128Byte, because DPDK processes short size packet faster than long size packet.

Flow recovery component was evaluated by varying the out-of-order ratio. From Fig. 6, we can see that when out-of-ratio increase from 10%-70% the reordering time slightly fluctuate around 90ms. The reordering time is 92ms when the reordering ratio is 80% to 94ms when the reordering ratio is 90%. The reordering time moderately fluctuate

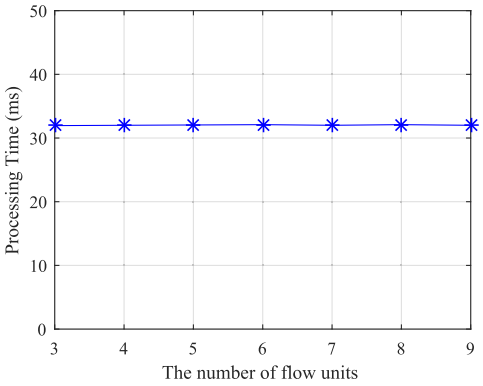

FIGURE 4. Processing time of the number of flow unit.

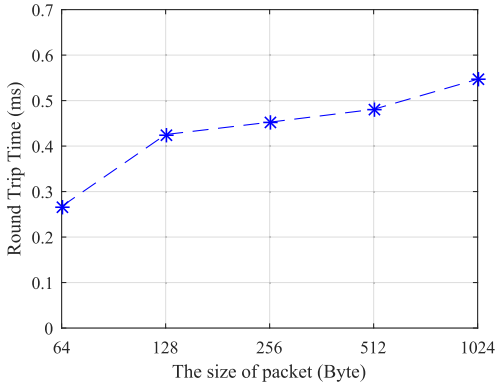

FIGURE 5. Forwarding time.

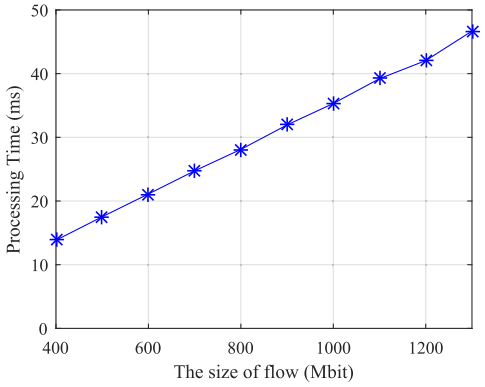

FIGURE 3. Processing time of flow splitting and mapping.

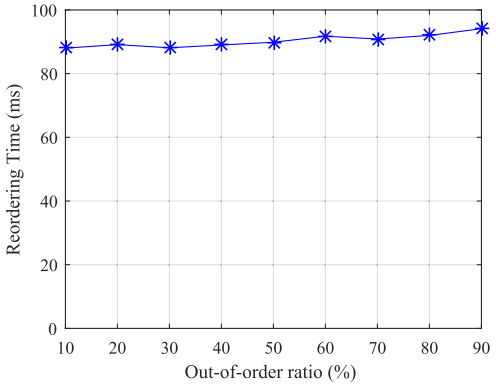

FIGURE 6. Packet reordering time.

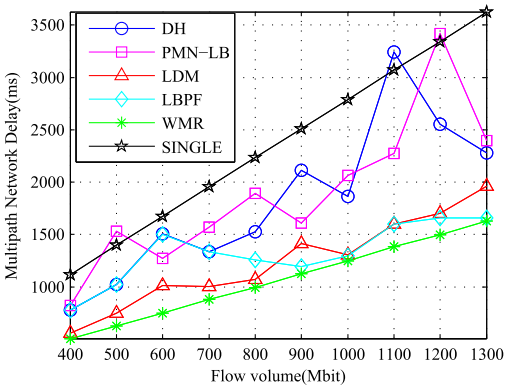

FIGURE 7. Performance comparison of multipath network delay.

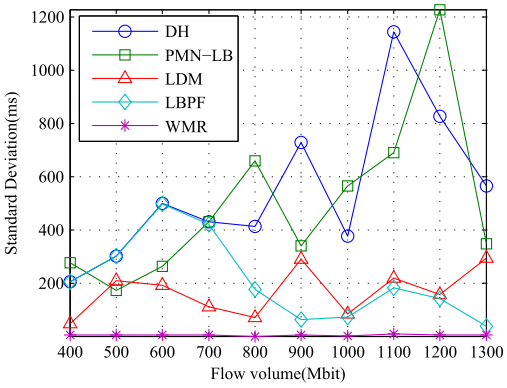

FIGURE 8. Performance comparison of standard deviation.

from 10%-90% in the experiment. This is because the number of packets are constant and all packets are sorted by Algorithm 1, therefore, the around time it takes.

From the evaluation of these four components, flow splitting, mapping and flow recovery introduce additional delay, i.e., processing time and reordering time. Therefore, the additional delay which is introduced from our scheme is about 120ms when the size of a flow is 900Mb.

C. MODELS PERFORMANCE COMPARISON

We first evaluated the differences in multipath network delay among the DH, PMN-LB, LDM, LBPF and WMR multipath models, and the single path delay that eight files transmit on 400Mbps available bandwidth of the path as baseline; see Fig. 7. The multipath models fluctuated except WMR, particularly, and DH and PMN-LB because each file size is different. That is, the size of the input flow is random; at 500Mbps and 1200Mbps, the PMN-LB has the larger delay than single path, and at 1100Mbps DH also has the larger delay than single path, because the DH and PMN-LB forward flow is based on inherent hash rules, and thus, the large flow was forwarded to the high load path at a low load, which caused the delay to increase. Additionally, the LDM had a lower delay than the DH and PMN-LB models. The reason is that the LDM chose the path with fewer hops and link utilization. Therefore, it avoided forwarding larger flows to the long-length queue. The LBPF had the same multipath network delay as the DH when the size was 400Mb-700Mb because when the path was not queued, LBPF and DH forwarded flows using the same hash function. However, the delay of LBPF was lower than that of DH at 800Mb because LBPF forwarded the new flow to an idle path or a path with a shorter queue. At 900Mb-1.3Gb, the delay of LBPF was lower than that of DH because LBPF splits the newly arrived flow when queuing occurs on the path, resulting in a lower queue delay. At 400Mb-800Mb, LBPF and LDM multipath networks were not queued, and the multipath network delay of LDM was lower than that of the LBPF. LDM and LBPF multipath network delays were the same or LDM was greater than LBPF at 900Mb-1.3Gb because LDM cannot split the flow, and the queued delay of LDM increased queuing on the multipath network. The WMR had a lower linear delay increase because the WMR splits each flow according to the available bandwidth of each path avoiding or decreasing the queue. Therefore, the delay of WMR was the lowest of the compared models.

As shown in Fig. 8, we tested the delay for each path and calculated the standard deviation of the delay for each path in the multipath network. The standard deviation reflected the deviation of the available bandwidth allocation of the models in each path. Fig. 8 shows that DH and PMN-LB had a larger standard deviation than the other three models because the hash-based algorithms only forwarded arriving traffic using the default rules. At 400Mb-800Mb, LDM had a lower standard deviation than LBPF but fluctuated at 900Mb because LDM forwarded flows to the path utilizing the current link. If there was a large difference between the size of the flows and three paths all had high link utilization, LDM forwarded the larger flow to the path with the lowest path utilization, increasing the path queue length. In the range of 900Mb-1.3Gb, although the size of flow exceeded the available bandwidth provided by the multipath network, the LDM still allocated the flow to the path with the lower link utilization. However, LBPF at 800Mb had a lower standard deviation than DH. The flow avoided being forwarded to the queued path because LBPF forwards the flow to an idle path. At 900Mb-1.3Gb, the standard deviation of LBPF was lower than that of LDM because LBPF splits the flow into packets based on the length of the queued sequence, with the shorter queued sequence receiving more packets. WMR had satisfactory performance due to WMR splitting the flow according to the availability of each path in proportion to the total available bandwidth. Therefore, the standard deviation of WMR was zero.

Fig. 9 shows the multipath network delay measured with 900Mb of traffic under various network loads. The X-axis is the multipath network load utilization from 50% to 95%. The Y-axis is the logarithmic coordinates represents various multipath network delay. In the figure, the load increases, and the delay of the five multipath models also shows

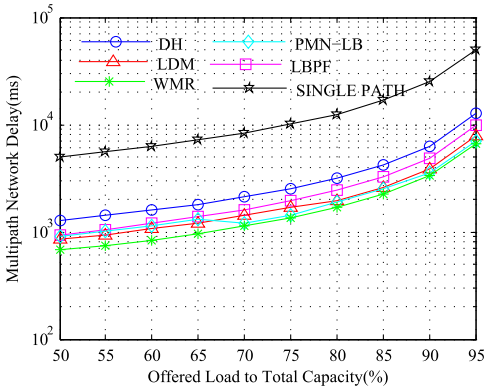

FIGURE 9. Multipath network delay with various load.

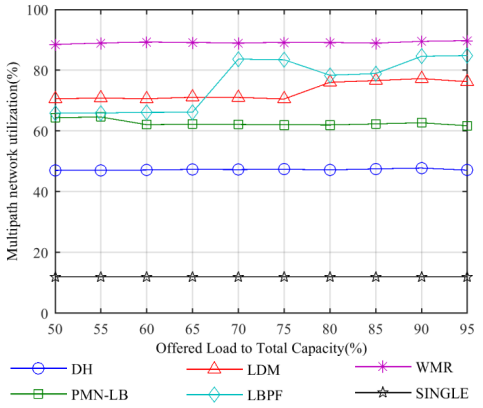

FIGURE 10. Performance comparison of multipath network utilization.

an increasing trend, and single path has the largest delay. At 50%, the delay of single path is about 5,000ms and five multipath models are all below than 1,600ms. With the increasing of offered load to total capacity, the trend of delay increasing of multipath models are much smaller than single path. DH had the longest delay compared to the other four multipath models. The shortest delay was in WMR, and the PMN-LB had a longer delay than LDM. However, when the load utilization was 70%, the delay of LBPF was shorter than that of LDM because LDM had the higher utilization of paths. If the next flow was larger, it was forwarded to the lower utilization path, causing an increase in the queuing delay. When LBPF had a larger load, the splitting mechanism was enabled, reducing the overload of the path and decreasing the delay.

Multipath network utilization reflects the utilization of the available bandwidth of a multipath network. As shown in Fig. 10, multipath models had superiority network utilization than single path, WMR had the highest utilization rate of approximately 90% among the models, single path had the lowest utilization rate of approximately 11.9%. DH and PMN-LB had lower utilization rates than the other three models, at almost 45%, 65% respectively. The utilization rate of LBPF was greater than that of DH, at 50%-65%, because LBPF forwards the flow to the queued shortest path. The utilization of LDM to the network was greater than that of LBPF at 50%-65%, and at 70%-95%,LBPF was utilized more than LDM because at 50%-65%, the length of the queue of each path is zero, whereas at 70%-95%, the length of the queue increases, and LBPF starts to split the flow to balance the length of the queue. Additionally, the idle time of the path decreased, and multipath network utilization increased.

V. CONCLUSION AND FUTURE WORK

This paper proposes and implements a scheme of multipath network virtualization with SDN and NFV. The control plane and data plane are incorporated in the scheme. The control plane managed the data plane using four functional components. Multipath was implemented in the data plane. Flow splitting under network virtualization is also presented in this paper. With the software design and NFV technology, nodes provide powerful computing and storage capabilities to implement multipath network flexibly. We also used an experimental platform based on OPNFV. The state-of-art models were tested on the platform, and the results were analyzed. A key future work is to apply Service Function Chaining (SFC) of NFV to meet additional requirements on services, such as load balancing. One can also apply Virtual Network Functions(VNFs) auto-scaling and auto-healing features in NFV to improve computing and storage resource utilization.

REFERENCES

[1] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui, "Future networking challenges: The case of mobile augmented reality," in *Proc. Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 1796–1807.

[2] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, Oct. 2006.

[3] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 6824, IETF, Fremont, CA, USA, 2013.

[4] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 51–62, Apr. 2007.

[5] W. Shi, M. H. MacGregor, and P. Gburzynski, "Load balancing for parallel forwarding," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 790–801, Aug. 2005.

[6] J. Song, S. Kim, M. Lee, H. Lee, and T. Suda, "Adaptive load distribution over multipath in NEPLS networks," in *Proc. ICC*, Anchorage, AK, USA, May 2003, pp. 233–237.

[7] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, "Network virtualization: Technologies, perspectives, and frontiers," *J. Lightw. Technol.*, vol. 31, no. 4, pp. 523–537, Feb. 15, 2013.

[8] M. Chiosi *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *Proc. SDN OpenFlow World Congr.*, Darmstadt, Germany, 2012, pp. 5–7.

[9] Ericsson. (Feb. 2014). *Telefonica and Ericsson Partner to Virtualize Networks, Press Release*. [Online]. Available: http://www.ericsson.com/news/1763979

[10] S. Prabhavat, H. Nishiyama, N. Ansari, and N. Kato, "On load distribution over multipath networks," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 3, pp. 662–680, 3rd Quart., 2012, doi: 10.1109/SURV.2011.082511.00013.

[11] S. K. Singh, T. Das, and A. Jukan, "A survey on Internet multipath routing and provisioning," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2157–2175, 4th Quart., 2015, doi: 10.1109/COMST.2015.2460222.

[12] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "Observing real smartphone applications over multipath TCP," *IEEE Commun. Mag.*, vol. 54, no. 3, pp. 88–93, Mar. 2016, doi: 10.1109/MCOM.2016.7432153.
[13] S. He, G. Shou, Y. Hu, and Z. Guo, "Performance of multipath in fiber-wireless (FiWi) access network with network virtualization," in *Proc. MILCOM*, San Diego, CA, USA, Nov. 2013, pp. 928–932.
[14] E. J. Kitindi, S. Fu, Y. Jia, A. Kabir, and Y. Wang, "Wireless network virtualization with SDN and C-RAN for 5G networks: Requirements, opportunities, and challenges," *IEEE Access*, vol. 5, pp. 19099–19115, 2017.
[15] T. N. Subedi, K. K. Nguyen, and M. Cheriet, "OpenFlow-based in-network layer-2 adaptive multipath aggregation in data centers," *Comput. Commun.*, vol. 61, pp. 58–69, May 2015.
[16] D. Banfi, O. Mehani, G. Jourjon, L. Schwaighofer, and R. Holz, "Endpoint-transparent multipath transport with software-defined networks," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, Dubai, United Arab Emirates, Nov. 2016, pp. 307–315.
[17] A. Basit, S. Qaisar, S. H. Rasool, and M. Ali, "SDN orchestration for next generation inter-networking: A multipath forwarding approach," *IEEE Access*, vol. 5, pp. 13077–13089, 2017.
[18] T.-T.-L. Nguyen, T.-M. Pham, and H. T. T. Binh, "Adaptive multipath routing for network functions virtualization," in *Proc. 7th Symp. Inf. Commun. Technol.*, New York, NY, USA, 2016, pp. 222–228.
[19] T.-M. Pham and L. M. Pham, "Load balancing using multipath routing in network functions virtualization," in *Proc. IEEE RIVF Int. Conf. Comput. Commun. Technol., Res., Innov., Vis. Future (RIVF)*, Hanoi, Vietnam, Nov. 2016, pp. 85–90.
[20] Q. Wang, J. Xue, G. Shou, Y. Liu, Y. Hu, and Z. Guo, "Implementation of multipath network virtualization scheme with SDN and NFV," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.
[21] *IEEE Standard for Local and Metropolitan Area Networks—Station and Media Access Control Connectivity Discovery*, IEEE Standard 802.1AB, 2009.
[22] P. Megyesi, A. Botta, G. Aceto, A. Pescapè, and S. Molnár, "Available bandwidth measurement in software defined networks," in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, New York, NY, USA, 2017, pp. 651–657.
[23] *Network Functions Virtualisation: Architectural Framework*, document GS NFV 002 v1.2.1, ETSI, Dec. 2014.
[24] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, pp. 712–716, Jul. 1971.
[25] J. Liu, J. Li, G. Shou, Y. Hu, Z. Guo, and W. Dai, "SDN based load balancing mechanism for elephant flow in data center networks," in *Proc. Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Sydney, NSW, Australia, Sep. 2014, pp. 651–657.
[26] *OpenFlow Switch Specification, Version1.3.1*. Accessed: Sep. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdnresources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf
[27] K. C. Leung, V. O. K. Li, and D. Yang, "An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 522–535, Apr. 2007.
[28] *OPNFV, Version Arno*. Accessed: Jun. 2015. [online]. Available: https://www.opnfv.org/software
[29] C. Villamizar, *OSPF Optimized Multipath (OSPFOMP)*, document Internet draft draft-ietf-ospf-omp-02.txt, Feb. 1999.
[30] J. Kim, B. Ahn, and J. Kim, "Multiple path selection algorithm using prime number," in *Proc. 10th IEEE Singapore Int. Conf. Commun. Syst. (ICCS)*, Singapore, Oct. 2006, pp. 1–5.
[31] C. Fraleigh *et al.*, "Packet-level traffic measurements from the sprint IP backbone," *IEEE Netw.*, vol. 17, no. 6, pp. 6–16, Nov. 2003.
[32] *Packet Reordering Metrics*, document RFC 4737, IETF, 2006.
[33] *DPDK, Version 16.04*. Accessed: Apr. 12, 2016. [Online]. Available: https://dpdk.org/rel

QINGTIAN WANG received the B.E. and M.S degrees from Zhengzhou University, Zhengzhou, China. He is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. His research interests are mainly network virtualization, mobile edge computing, and multipath networks.

GUOCHU SHOU is currently a Professor with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His research interests include access network and edge computing, fiber and wireless network virtualization, network construction and routing, and mobile Internet and applications.

YAQIONG LIU received the double bachelor's degrees in computer science and engineering and financial management from Tianjin University, China, in 2009, and the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2016. She is currently a Lecturer with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include data mining, networking and computing, spatial query processing, GIS, location-based services, and image animation.

YIHONG HU is currently an Associate Professor with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. Her research interests include fiber and wireless communications, network virtualization, and cloud computing.

ZHIGANG GUO is currently a Professor with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His research interests are mainly 5G wireless communications and networks and cognitive computing.

WEI CHANG received the B.E. degree in communication engineering from Tianjin University in 2014. She is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. Her research interests are mainly edge computing and network planning.