Design and Implementation of an SD-WAN VPN System to Support Multipath and Multi-WAN-Hop Routing in the Public Internet

Steven Lee [1], Kwan-Yee Chan [2], and Ting-Yun Chen [2]

[1]National Chung Cheng University
[2]Affiliation not available

October 30, 2023

Abstract

We present the design of a multipath multi-WAN-hop SD-WAN (MMS) system to realize an overlay network on top of the public internet. The MMS includes an SD-WAN system controller (SSC) and MMS gateways (MMSGs), one for each branch. The SSC is responsible for configuring the routing paths for the whole system. The MMSG uses low-cost access networks such as PON, xDSL, PLC, cable modems, and even LTE/5G to access the public internet. We propose an IP address swapping technique to realize multihop routing in the public internet. In addition, we implement IP over MPTCP (IPoMP) in MMSGs, in which a flow between a pair of branches is mapped into multiple MPTCP subflows to exploit multipath routing.

Design and Implementation of an SD-WAN VPN System to Support Multipath and Multi-WAN-Hop Routing in the Public Internet

Steven S. W. Lee, Kwan-Yee Chan, and Ting-Yun Chen

Abstract—**The software-defined wide area network (SD-WAN) is a new virtual private network (VPN) technology that enables an enterprise to interconnect all of its geographically distributed branch campuses through a low-cost public internet. Conventionally expensive leased lines are deployed to fulfill the requirement for communication in a multicampus enterprise. The SD-WAN VPN takes advantage of the flexible programmability of software-defined networking (SDN) to provision routing paths and perform traffic control to reduce network costs by offloading traffic from expensive leased lines to the public internet. In this paper, we present the design of a multipath multi-WAN-hop SD-WAN (MMS) system to realize an overlay network on top of the public internet. The MMS includes an SD-WAN system controller (SSC) and MMS gateways (MMSGs), one for each branch. The SSC is responsible for configuring the routing paths for the whole system. The MMSG uses low-cost access networks such as PON, xDSL, PLC, cable modems, and even LTE/5G to access the public internet. We propose an IP address swapping technique to realize multihop routing in the public internet. In addition, we implement IP over MPTCP (IPoMP) in MMSGs, in which a flow between a pair of branches is mapped into multiple MPTCP subflows to exploit multipath routing. To evaluate our MMS system, we implement an experimental network. Compared to the conventional IP-based VPN that uses IP-in-IP tunneling, the proposed IPoMP-based MMS system can significantly enhance network throughput for a multicampus enterprise.**

Index Terms — **Software-Defined Networking (SDN); Software-Defined WAN (SD-WAN); Virtual Private Network (VPN); IP over MPTCP (IPoMP); Multipath Routing; Overlay Network; IP Address Swapping**

I. Introduction

THE software-defined wide area network (SD-WAN) applies software-defined networking (SDN) technology to reduce network building costs and enhance flexibility in controlling the network. The SD-WAN can be classified into two categories. One directly employs SDN switches to construct a WAN. The other applies SDN to realize gateways and edge routers to facilitate efficient traffic control between enterprise branches or datacenters. The former is usually deployed and operated by large organizations or telecommunication network operators who own their WANs. They take full advantage of the flexibility of SDN to provide network services to their customers. The latter does not change the existing WAN. Instead, the focus is on establishing a virtual private network (VPN) on top of the public internet by using low-cost, commercially available internet access technologies.

All the authors are with the Department of Communications Engineering, National Chung Cheng University, Chiayi, Taiwan (e-mail: ieeswl@ccu.edu.tw; kwanyee86@gmail.com; tingyun0313@gmail.com).

The major cloud service providers have their own dedicated WANs. Google [1], Microsoft [2], and Facebook [3] have applied SDN technology to their networks. SWAN [4], B4 [5], and BwE [6] implement SDN-based traffic engineering to improve the performance of the inter-data center WAN. Grace [7] introduces APIs for customers by abstracting WAN connections based on the connection types, bandwidth, latency sensitivity, and policy-related information. Grace also develops an effective conflict detection algorithm considering both resource reservation and safety guarantees.

Telecommunication network operators can use SDN technology to provide VPNs for their customers. The SDxVPN [8] is an SDN-based VPN solution that enables a network service provider to provide VPN services. The core network for SDxVPN is an MPLS network owned by the service provider. The system applies SDN in provider edge devices. With the help of SDN, SDxVPN achieves flexible control, enabling efficient use of the MPLS network resources. Another example of using SDN to manage an MPLS-based VPN can be found in [9]. In [10], SDN is used to enhance policy-based routing and load balancing for Ethernet VPN (EVPN)-based data centers.

In addition to applying the SDN-based control paradigm to manage a WAN, another kind of SD-WAN technology focuses on the design of gateways or edge routers [11][12] . The targets of these kinds of products use common low-cost access networks such as xDSL, cable modems, PLC, and PON to provide low-cost VPN services among branches of an enterprise or an organization. The article in [13] lists the definition of the SD-WAN provided by the research firm Gartner. An SD-WAN must be able to support multiple connection types. It needs to be able to perform dynamic path selection for load sharing and resiliency purposes. In addition, the controller should be able to configure and manage the whole system.

In this paper, our goal is to design an SD-WAN VPN system that can enable an enterprise to interconnect all of its geographically distributed branches through a low-cost public internet. Figure 1 depicts the system architecture. The whole system includes an SD-WAN system controller (SSC) and multiple multipath multi-WAN-hop SD-WAN gateways (MMSGs), one for each branch. The controller performs path planning and provisioning, assigns traffic classification and prioritization, and collects statistical data for the whole SD-WAN VPN through configuring and controlling the MMSGs.

In the example shown in Fig. 1, the enterprise consists of three branches, each of which is equipped with an MMSG. The gateway connects to the public internet through standard low-cost access networks such as xDSL, PON, PLC, cable modems, and even LTE/5G. If a leased line is available for the branch, the leased line is by default used to deliver high-priority interbranch traffic. To increase the bandwidth utilization of the expensive leased line, an MMSG could transmit low-priority traffic through the leased line if some bandwidth remains. The SDN switch inside the MMSG guarantees traffic delivery based on strict priority; hence, the low-priority traffic will not block the transmission of high-priority traffic at any time.

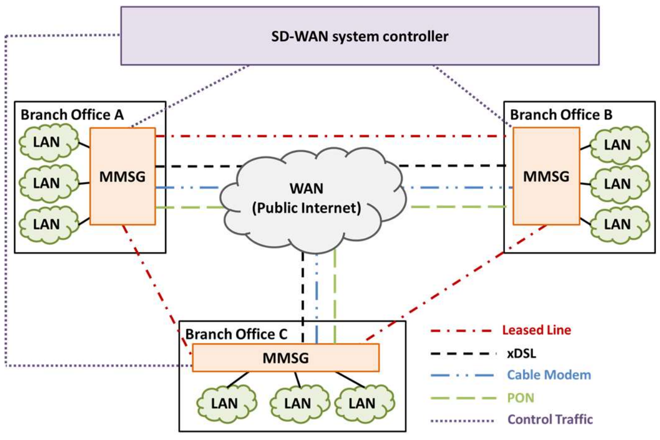


Fig. 1. System architecture

Our system can be viewed as an overlay network that is running on top of the public internet. In the overlay network, we realize multi-WAN-hop routing. A packet leaving its source branch can either be directly sent to the destination branch or take another intermediate branch as a transit node. To facilitate routing over the public internet, we set up IP tunnels between branches. In our system, each port that connects to the public internet is assigned a public IP address. The MMSG delivers a packet from one branch to another by attaching each outgoing IP packet to another outer IP header in which the destination address is the public IP address of the remote branch. By performing IP address swapping on the outer header at a transit MMSG, multi-WAN-hop routing can be achieved.

In addition to using multi-WAN-hop routing, we also employ multiple paths in parallel to enhance throughput. We apply the multipath TCP (MPTCP) [14] to carry a flow between a pair of end hosts that are in different branches over multiple paths in the overlay network. The MPTCP provides a sublayer beneath the TCP to divide a TCP connection into subflows. In the MPTCP, each subflow can take a different path in the network. A subflow has its own congestion and error controls. Our system implements IP over MPTCP (IPoMP) tunneling. As a result, an end-to-end connection can use the UDP, TCP, or any other protocol as their transport protocol. The MMSG at the source branch encapsulates an outgoing IP packet as an MPTCP payload, and the MMSG at the destination branch decapsulates the IP packet. The MPTCP is able to reorder the receiving packets at the destination MMSG to prevent out-of-order delivery when multipath routing is applied.

The MPTCP has been applied in data centers to improve the TCP throughput. The studies in [15][16] show that the MPTCP always outperforms the single-path TCP. Since the MPTCP is a layer-four protocol, in the internet, the routing for the subflows is provided by the underlying IP network that employs equal-cost multipath (ECMP)-based shortest path routing. Provisioning multipaths to support the MPTCP has been carried out in SDN networks [17][18][19].

In [17], an OpenFlow testbed is implemented to provide multiple paths to support the MPTCP. The controller of the system monitors the throughput of the network to determine the configuration of the routing paths. When a network failure is detected, the controller can reconfigure the network to maximize its throughput. In [18], the MPTCP is used to enhance network throughput in a hybrid SDN and Ethernet-based data center network. The experimental results show that the MPTCP outperforms the existing ECMP-based and VLB-based routing in an SDN network. In [19], the authors evaluate the performance of the MPTCP on top of the GÉANT and PlanetLab Europe testbed networks. The authors conclude that the version of the MPTCP implementation was considerably good in 2014, when the paper was published. In this paper, we use the most up-to-date MPTCP implementation in our system.

In these works [17][18][19], SDN is used to provide multipaths to enhance the throughput of connections. End hosts must perform the MPTCP to enjoy the benefit of using multiple paths. However, in our work, the end hosts can perform any layer-four protocols. The operations for the encapsulation and decapsulation of IP packets at MMSGs are transparent to the end hosts. We take advantage of SDN and the MPTCP to enhance the network throughput and reliability of the SD-WAN VPN.

In summary, our major contributions are as follows:

- We propose an MMS system that applies multipaths to enhance network throughput and reliability for interbranch communications inside an organization. The proposed MMSGs are able to handle congestion control and packet out-of-order issues for every interbranch end-to-end flow.
- Through IP address swapping, the proposed system can employ multi-WAN-hop routing in the overlay network on top of the public internet.
- The proposed system handles traffic prioritization to enhance the QoS of the network.
- We present the detailed design of the proposed MMSGs and the system controller.
- A network testbed is implemented to demonstrate the feasibility and performance of the proposed system.

The remainder of this paper is organized as follows. Section II presents the detailed design of the proposed MMSG. In Section III, we introduce the control and management functions used in the proposed MMS system. In Section IV, we report the experimental results and make performance comparisons between networks with and without the proposed system. Finally, concluding remarks are presented in Section V.

II. Multipath Multi-WAN-Hop SDN Gateway

The modules inside an MMSG are shown in Fig. 2. The gateway is an edge device that is used to connect a branch of an organization to the public internet. The interfaces between the MMSG and the public internet depend on the local ISPs to which the branch subscribes. The possible access networks include xDSL, cable modems, PON, PLC, and even LTE/5G. In addition to the ports that connect the MMSG to the public internet, ports to leased lines are available if the branch has them. An MMSG uses the standard Ethernet to connect the internal local area networks.

An MMSG consists of three modules: a traffic classifier (TC), a multipath agent (MPA), and an OpenFlow switch (OFS). The details of these modules are presented below.

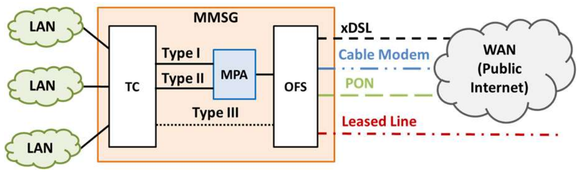


Fig. 2. Modules inside an MMSG

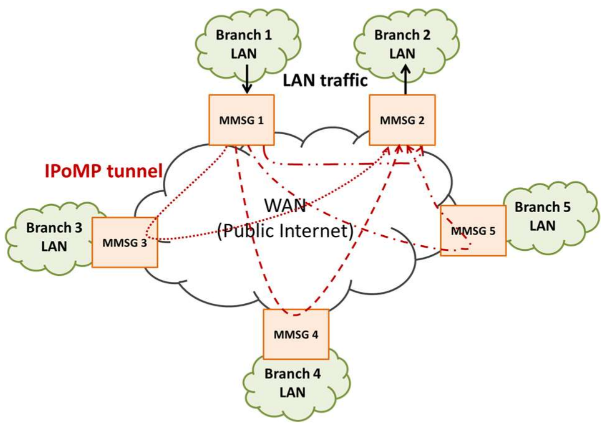


Fig. 3. An example of multipath multi-WAN-hop routing

A. *Traffic classifier (TC)*

The TC is responsible for classifying the outgoing traffic into three priorities in descending order. Type I and Type II are internal interbranch traffic. Their source and destination hosts are in the same organization but not in the same branch. Type III is used for external traffic. The MMSG assigns Type I traffic the highest priority for packet scheduling to minimize the transmission delay and packet loss probability. If the organization has subscribed a leased line service, Type I traffic will be carried by the leased lines; otherwise, Type I traffic will use the public internet for packet delivery. Type II traffic is fundamentally transmitted through the public internet. It will be carried by a leased line only if there is unused bandwidth remaining after serving the Type I traffic.

To enhance throughput, we use multipath routing for Type I and Type II traffic to exploit more than one path for packet delivery. The multipath function is realized by the MPA module. The MPA performs packet scheduling on the multipaths at the sender side and resolves the packet out-of-order problem at the receiver side.

Type III traffic has the lowest priority. The destinations of Type III traffic are outside of the organization. They are sent to the public internet directly without any additional processing by the MMSG.

The TC can be implemented by a server. It can also be implemented by using an OFS. If an OFS is used, the controller needs only to configure the matching rule in the flow table to determine how to classify the outgoing traffic into the three types of classes.

B. *Multipath agent (MPA)*

In the system, if a flow takes multipath routing, it is mapped into an MPTCP connection. In the example shown in Fig. 3, a TCP connection from a source host in branch 1 toward a destination server in branch 2 exists. The packets of this TCP connection are carried by four MPTCP subflows that take paths 1-2, 1-3-2, 1-4-2, and 1-5-2 to the destination.

Our MMS realizes IPoMP on top of the public internet. The encapsulation and decapsulation processes are performed at the MPAs of the MMSGs in the source and destination branches. Our MPAs work as middle boxes, and the end hosts are not involved in the multipath routing. The two end hosts can use any transport protocol (e.g., UDP, TCP, or SCTP) to communicate with each other. They are unaware of the presence of the MPTCP. The SSC determines the routing on the overlay network so that an MPTCP subflow can either take a one-WAN-hop path or a multiple-WAN-hops path to reach the destination.

An MPTCP connection is maintained only by the two end MPAs at the source and destination MMSGs. For transit traffic, packets are not delivered to the MPA at a transit node. They are directly handled by the OFS at the transit MMSG.

Assuming the MPA at a source branch has n ports and the MPA at the destination branch has m ports, there are $n \times m$ MPTCP subflows that support the MPTCP connection. The number of physical ports connecting the MPA to the OFS can be different from the number of physical ports connecting the MMSG to the public internet. For simplicity, in our current implementation, the number of MPA ports is the same as the number of ports connecting the MMSG to the internet.

Because the MPTCP is connection-oriented, mapping an incoming flow into an MPTCP connection will introduce additional setup delay. To eliminate the delay, a set of MPTCP connections between any pair of MMSGs is set up in advance. Thus, as a new traffic flow arrives at a gateway, it is immediately mapped to an existing MPTCP connection for packet delivery. By using this technique, the MPTCP introduces no extra delay in setting up a connection for an outgoing flow.

C. *OpenFlow switch (OFS)*

The OFS is responsible for realizing multi-WAN-hop routing for MPTCP subflows. Its functionalities include IP address swapping, routing path ID insertion, priority queueing, and statistical data collection.

The whole network uses IP tunnels to realize an overlay network on top of the public internet. A packet can traverse through a transit branch before reaching its final destination.

For this case, the destination IP address in the outer header is assigned to be the public IP address of the MMSG at the transit branch. The OFS at the transit branch performs IP address swapping to replace the source and the destination IP addresses in the outer header with a pair of new source and destination IP addresses. More specifically, the source IP address is replaced with the IP address of the port through which the packet leaves the MMSG, and the destination IP address is assigned to be the IP address of the port through which the packet is received at the next hop node.

To facilitate routing at a transit node, we include the source branch ID, destination branch ID, path group ID, and member path ID in the TCP destination port of the outer header. We use the term "path group" to indicate a set of member paths to support an MPTCP connection. The routing of an MPTCP connection between a pair of branches follows the path group assigned at the source MPA. Each MPTCP subflow takes one member path for routing. Through a combination of the path group ID and member path ID for a pair of branches, the specific routing paths for MPTCP subflows in the overlay network can be identified.

By examining the TCP destination port in the outer header, the OFS at a transit node resolves how to perform IP address swapping and how to determine the outgoing link through which the packet can be forwarded. By examining the TCP destination port in the outer header, the OFS at the destination branch determines how to recover the IP addresses back to their original values as they are generated by the MPA at the source branch. As a result, the MPA at the destination branch can successfully perform MPTCP flow identification and packet reordering.

The OFS also determines packet scheduling based on its priority. Priority queueing is applied such that packets with high priority will not be blocked by low-priority packets for network resource usage. The detailed routing and traffic prioritization are presented in the next section.

III. Routing Configuration and traffic prioritization

In this section, we introduce the detailed encoding of the outer header and the setting of flow entries in OFSs for IP address swapping. We show the detailed operations performed in the source, transit, and destination MMSGs. Finally, we illustrate the operations with an example.

A. Operations at the source MMSG

For each outgoing flow, the MPA at the source MMSG determines the routing and maps the outgoing packets into multiple MPTCP subflows. According to the MPTCP, the number of subflows that can be set up between two end nodes is the product of the number of interfaces (ports) of these two nodes. Assuming the MPA in the source branch has n ports and the MPA in the destination branch has m ports, there are $n \times m$ MPTCP subflows that support each MPTCP connection between these two branches. We denote x_i and y_j, where $1 \leq i \leq n$ and $1 \leq j \leq m$, as the IP addresses of the i-th and j-th ports of the MPAs in the source branch and destination branch, respectively. At the destination branch, the MPA can identify an MPTCP subflow by examining the TCP source and destination ports and the pair of source IP and destination IP addresses (x_i, y_j) in the outer header.

To perform routing in the overlay network, in the MMS, each source branch and destination branch pair is preassigned a set of path groups. Figure 3 shows an example in which a path group exists between branch 1 and branch 2. This path group includes four member paths, one for each subflow. When a new MPTCP connection setup is used, the MPA at the source MMSG selects one path group for routing the MPTCP connection. The selection can be based on a round robin or depend on the average throughput of a path group.

To facilitate a transit node to perform multi-WAN-hop routing, we encode the source branch ID, destination branch ID, path group ID, and member path ID in the outer header. This quaternary information is carried by the TCP destination port. Each of these four fields occupies 4 bits.

The destination branch ID and path group ID are assigned by the MPA at the source branch. The destination branch ID and path group ID occupy the first nibble and the last nibble of the TCP destination port. Because the path group ID is 4 bits long, each pair of branches has 16 path groups at maximum. This number is large enough for the application of the SD-WAN VPN. For an MPA, the second nibble and the third nibble of the TCP destination port are fixed at 0x00. Therefore, for a branch with ID b, its MPA has to listen to TCP destination ports ranging from 0x*b000* to 0x*b00k*, where $k \leq 15$ is the maximum path group ID configured in this MMSG. As a result, when the MPA in branch a wants to use path group g to send an MPTCP packet to branch b, the destination TCP port is assigned as 0x*b00g*.

In addition to the destination branch ID and path group ID, to facilitate MMSGs at the transit branch to perform IP address swapping, the source branch also includes the source branch ID and member path ID in the TCP destination port of the outer header. Both the source branch ID and member path ID are included in the second nibble and the third nibble of the TCP destination port by the OFS at the source MMSG. As a result, if a subflow between source branch a and destination branch b follows the routing identified by member path s in path group g, the destination port of the packet's outer header becomes *0xbasg* when it leaves the source branch.

B. Operations at the transit MMSG

At the transit branch, the transit traffic is handled only by the OFS. The system controller sets up the routing paths by downloading flow entries to the OFS. By matching the quaternary information carried by the TCP destination port of the outer header, the OFS retrieves the actions to perform IP address swapping and determines the outgoing link for packet forwarding.

C. Operations at the destination MMSG

By examining the branch ID in the destination port of the outer header, the OFS at the destination MMSG knows it is the final stop for this incoming packet. The OFS is responsible for restoring the source IP and destination IP addresses back to the values assigned by the MPA at the source MMSG. In addition, the destination TCP port is recovered to its original values by resetting the second nibble and the third nibble to zeros.

If an incoming packet does not match any flow entry in the flow table of the OFS, then the packet does not belong to an MPTCP connection inside the organization. In that case, the OFS bypasses the MPA and directly forwards the packet to the device behind the MMSG.

D. Example

Here, we give an example following the case shown in Fig. 3. The ports used in the example are depicted in Fig. 4(a). Each branch has two physical ports. In this example, the source host is inside branch 1, and the destination host is inside branch 2. The MPTCP connection that supports the communication has four subflows.

We assume that this pair of end hosts uses a TCP connection to communicate. As a result, the SD-WAN will encapsulate a TCP/IP packet in an MPTCP payload. Here, we assume that the MPA at the branch selects path group 4 for routing. The detailed routings for these four subflows are as follows:

Subflow 1:	Hop 1: MMSG 1 (port 1)→MMSG 3 (port 1)
	Hop 2: MMSG 3 (port 1)→MMSG 2 (port 1)
Subflow 2:	Hop 1: MMSG 1 (port 2)→MMSG 4 (port 1)
	Hop 2: MMSG 4 (port 2)→MMSG 2 (port 1)
Subflow 3:	Hop 1: MMSG 1 (port 1)→MMSG 5 (port 1)
	Hop 2: MMSG 5 (port 1)→MMSG 2 (port 2)
Subflow 4:	Hop 1: MMSG 1 (port 2)→MMSG 2 (port 2)

Figure 4(b) shows the detailed operations and the setting of the flow entries at each OFS involved to support the example MPTCP between branch 1 and branch 2. We denote as p_i^n the *i*-th WAN port of the MMSG at branch *n*. As shown in the figure, the MPA at branch 1 spreads the packets of the connection into four subflows, in which the source and destination IP pairs in the outer headers are (p_1^1, p_1^2), (p_1^1, p_2^2), (p_2^1, p_1^2), and (p_2^1, p_2^2), respectively. As a result, the MPA assigns 0x*2004* to the TCP destination port to indicate that branch 2 is the final destination and that the routing paths follow path group 4.

After leaving the MPA, these subflow packets are then sent to the OFS at MMSG 1. By matching the destination port, the source IP address, and the destination IP address, the OFS at MMSG 1 determines the next hop node for the packet and accordingly updates the source IP and the destination addresses and the destination port. To accomplish the routing and IP swapping for subflow 1, the match and actions in the flow table of the OFS at MMSG 1 are as listed below. Similarly, the match and actions for the other 2-hop paths, i.e., subflows 2 and 3, can be found in the figure.

Match: "Src IP =p_1^1, Dest IP =p_1^2, and TCP port="0x2004"
Action: Write "Src IP =p_1^1, Dest IP =p_1^3, and Dest port="0x2114", output the packet to WAN port 1

Subflow 4 follows a direct hop in the public internet, and the next hop node for this subflow is branch 2. The match and actions for subflow 4 at the OFS of MMSG 1 are as follows.

Match: "Src IP =p_2^1, Dest IP =p_2^2, and Dest port="0x2004"
Action: Write "Src IP =p_2^1, Dest IP =p_2^2, and Dest port="0x2144", output the packet to WAN port 2

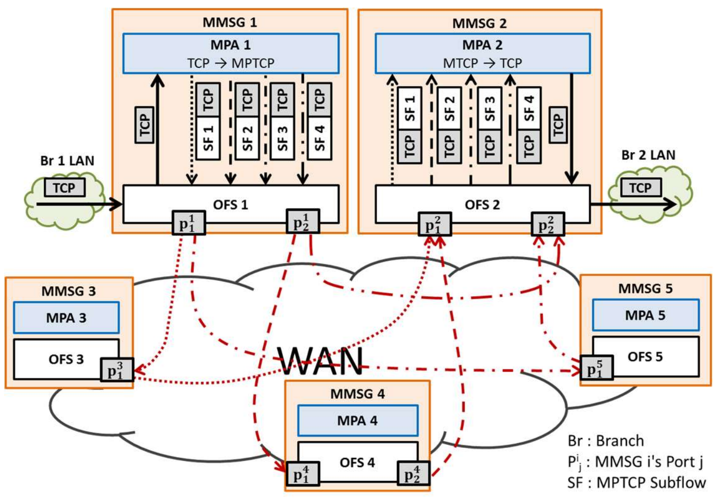
Let us examine the operations at a transit MMSG. For subflow 1, branch 3 is a transit branch. The OFS inside MMSG 3 is configured to accept the MPTCP packet from branch 1 and then forwards the packet to branch 2 after performing IP swapping. The flow entry in branch 3 for subflow 1 is as follows.

Match: "Src IP =p_1^1, Dest IP =p_1^3, and Dest port="0x2114"
Action: Write "Src IP =p_1^3, Dest IP =p_1^2, and Dest port="0x2114", output the packet to WAN port 1

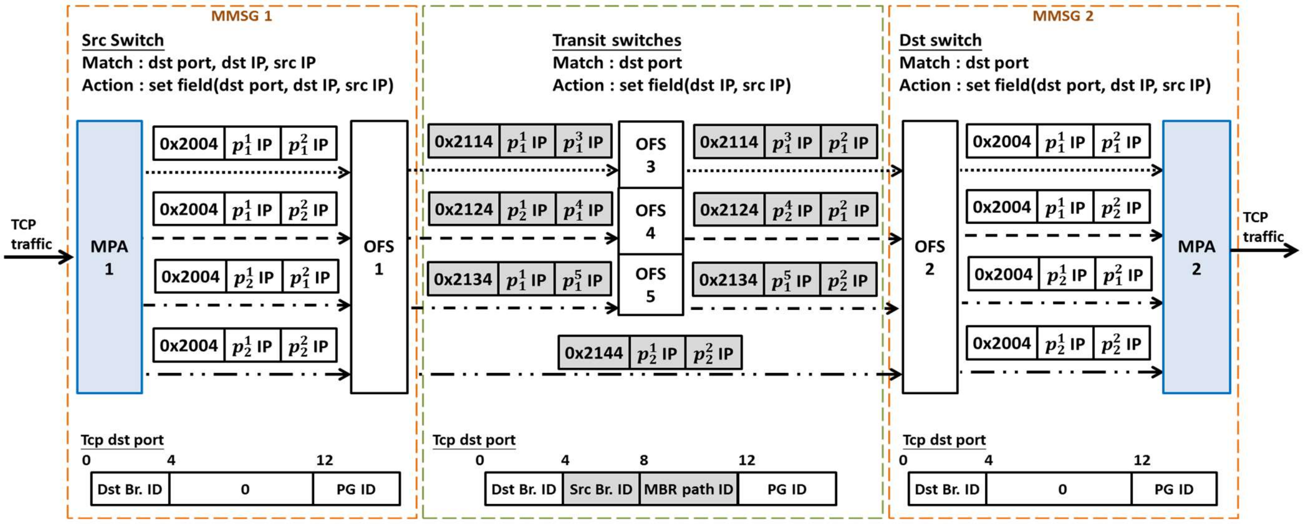
Finally, at the destination node, i.e., branch 2, the OFS converts the source IP, destination IP, and TCP port of all incoming MPTCP packets back to the same values as those when they left from the source MPA at branch 1. For example, the original MPTCP packet of subflow 1 had a source IP=p_1^1, destination IP=p_1^2, and TCP port number 0x2004. The flow entry at the OFS inside MMSG 2 is configured as follows for TCP port number recovery.

Match: "Src IP =p_1^3, Dest IP =p_1^2, and Dest port="0x2114"
Action: Write "Src IP =p_1^1, Dest IP =p_1^2, and Dest port="0x2004", output the packet to the port connected to the MPA

Finally, after receiving an in-sequence packet, the MPA at the destination branch, i.e., branch 2, removes the MPTCP outer header. The native packet is then delivered to the local area network of branch 2.



(a) Path group with four paths to support the example MPTCP connection


MMSG 1
Src Switch
Match : dst port, dst IP, src IP
Action : set field(dst port, dst IP, src IP)
Transit switches
Match : dst port
Action : set field(dst IP, src IP)
MMSG 2
Dst switch
Match : dst port
Action : set field(dst port, dst IP, src IP)
TCP traffic
MPA 1
OFS 1
OFS 3
OFS 4
OFS 5
OFS 2
MPA 2
0x2004 | p_1^1 IP | p_1^2 IP
0x2004 | p_1^1 IP | p_2^2 IP
0x2004 | p_2^1 IP | p_1^2 IP
0x2004 | p_2^1 IP | p_2^2 IP
0x2114 | p_1^1 IP | p_1^3 IP
0x2124 | p_2^1 IP | p_1^4 IP
0x2134 | p_1^1 IP | p_1^5 IP
0x2114 | p_1^3 IP | p_1^2 IP
0x2124 | p_2^4 IP | p_1^2 IP
0x2134 | p_1^5 IP | p_2^2 IP
0x2144 | p_2^1 IP | p_2^2 IP
Tcp dst port
0 4 12
Dst Br. ID | 0 | PG ID
0 4 8 12
Dst Br. ID | Src Br. ID | MBR path ID | PG ID

(b) Outer header and flow entry assignment in the source, transit, and destination MMSGs

Fig. 4. Example of detailed routing configuration in the MMS system

IV. SD-WAN System Controller

In this section, we present the design of the SSC. As shown in Fig. 5, the modules inside the SSC include the user interface (UI), path configuration (PC), statistical data collection (SDC), OpenFlow controller (OFC), and priority management (PM). The functions provided by each module are described below in detail.

- **User Interface (UI)**: Through the UI, a user can add, remove, and modify branch information, including branch IDs, number of access ports, and public IP addresses. A user also specifies the type of ports for leased line services and regular public internet services. Statistical data for each branch and the traffic amount between a pair of branches, which are collected by the SDC module, are provided to the user through the UI.
- **Path Configuration (PC)**: This module is responsible for determining the routing on top of the SD-WAN overlay network. PC module can accept a manual configuration by the operator and/or an automatic configuration. To realize autoconfiguration, this module periodically measures the paths on the overlay network. Based on the measured results, the path groups and their member paths are determined. The path group IDs are sent to the corresponding MPAs to indicate the routing paths between a pair of branches. The PC module also provides the path group IDs and member path IDs to the OpenFlow controller. Accordingly, the OpenFlow controller configures the flow tables in the OFSs. For each pair of branches, we configure 16 path groups. These 16 path groups and the routing of the member paths for each path group are semipermanent. The PC changes the routing of the paths only when it finds a better routing between the pair of branches.
- **Statistical Data Collection (SDC)**: This module collects statistical data through periodically polling MMSGs. The SDC module includes a timer. When the timer expires, it triggers the OpenFlow controller module to perform a multipart request to obtain the byte count on the OFSs inside the MMSGs. The statistical data are provided to the PC module for path configuration. The data are also provided to the user through the UI on demand.
- **OpenFlow Controller (OFC)**: The UI, SDC, and PC are the applications on top of the OFC. The OFC is used to control OpenFlow switches inside the MMSGs through the standard OpenFlow protocol. It receives the path configurations from the PC module, and the routing paths are downloaded to the OFSs. In addition, the OFC collects the statistical data for the SDC. Our OFC is currently implemented based on the Ryu controller [20].
- **Priority Management (PM):** In an SD-WAN, multiple MPTCP connections work on the overlay network. Multi-WAN-hop routing sometimes consumes more bandwidth than single-hop routing. Fig. 6 presents an example, in which each pair of branches has two routing paths: one is a direct WAN-hop path, and the other goes through a two-WAN-hop path by taking the other branch as a transit node. For instance, the two paths from source branch A to destination branch C are A→C and A→B→C. Assuming the physical port speed is c Mbps and the internet can support a bandwidth larger than c Mbps between each pair of branches, if only one-hop routing is applied, the throughput for each connection is close to c Mbps. However, if multipath routing is applied and traffic is equally split between the two paths, the throughput for each subflow becomes $c/3$ Mbps, and the total throughput for each pair of branches becomes $2c/3$. To resolve this problem, we fully take advantage of priority queues provided by the OFS at each MMSG to realize the hop-count-based priority. By assigning higher priority to traffic with shorter WAN hops, this system simultaneously maintains the benefit of multipath routing and prevents bandwidth waste caused by using longer WAN-hop paths. The MPTCP flows can efficiently

utilize the network capacity without causing unnecessary traffic blocking.

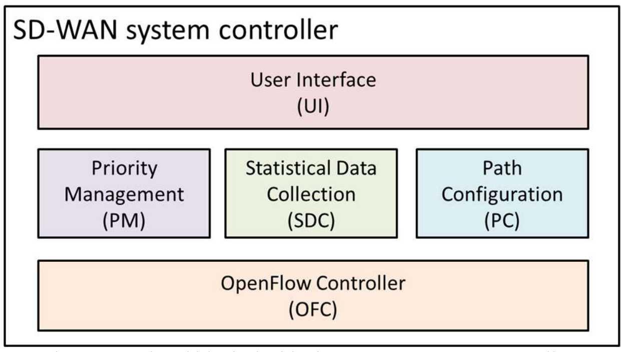
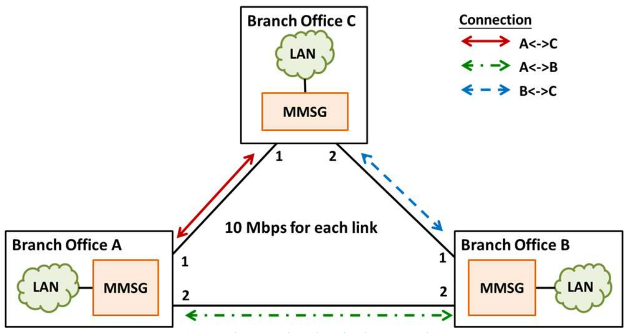
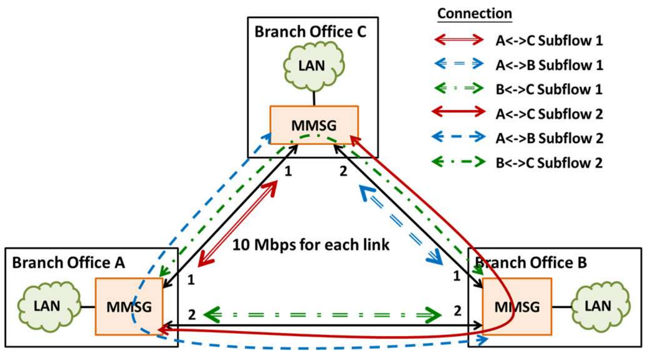


Fig. 5. Functional blocks inside the SD-WAN system controller



(a) Using only single-hop paths



(b) Using both single-hop and two-hop paths

Fig. 6. An example to demonstrate the requirement for using priority management

V. Experimental Results

We conducted four sets of experiments to evaluate the performance of the proposed MMS system. In the first set of experiments, we evaluate the goodput of a pair of end hosts in two different branches when they are communicating with the UDP and TCP. In our MMSG, each end-to-end connection is mapped to an MPTCP connection to employ multipath routing. To evaluate the overhead introduced by our IPoMP approach, we make performance comparisons with the end-to-end MPTCP, in which the two end hosts directly use the MPTCP to communicate with each other without going through our MMSGs in the middle.

We also evaluate the goodput of end-to-end connections under various scenarios on the bottleneck links. In the second set of experiments, we consider the cases where the available bandwidth on the bottleneck link is constant. In the third set of experiments, we evaluate the goodput of end-to-end connections when the SD-WAN traffic shares the bottleneck link with TCP flows. In the final set of experiments, we examine the goodput by assigning various ratios of available bandwidths to the subflows.

A. Overhead generated by IPoMP

To connect two hosts, the MPTCP usually outperforms the TCP. However, in our application, an outgoing IP packet is encapsulated as an MPTCP payload at our source MMSG. It introduces an outer header for each packet. In addition to the overhead caused by the outer header, IPoMP introduces some redundant controls that might reduce the throughput of an end-to-end connection. Because the TCPs at the end hosts have their own congestion control, flow control, and error control, if a pair of end-to-end hosts uses the TCP as their transport protocol, the MPAs in our MMSGs duplicate the control functions. To ensure no interference between the two control mechanisms between the end-to-end TCP and the MPTCP in our MMSGs, in this set of experiments, we make performance comparisons between the cases when the MPTCP is applied and those in which it is not applied in the middle between a pair of end hosts.

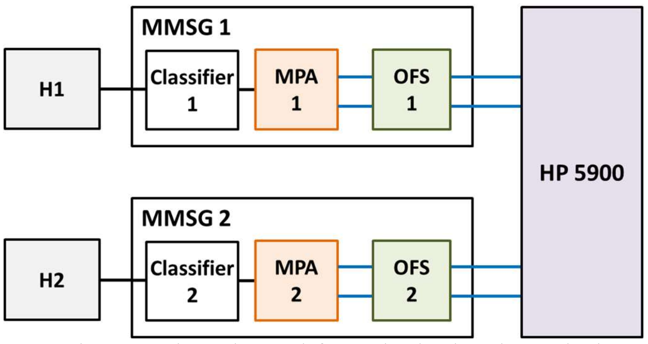


Fig. 7. Experimental network for overhead and goodput evaluation

We used an HP 5900 OpenFlow switch to set up the experiment shown in Fig. 7. Host 1 uses the standard TCP and UDP to transmit data to host 2. Because two physical ports exist at each MMSG, four MPTCP subflows are generated. The four subflows follow separate routes inside the HP 5900 switch. We use the meter function provided by the HP 5900 switch to generate different available bandwidths for the subflows. Table I includes the detailed rate limitation for each subflow and the experimental results. In each test case, the total capacity was set to 40 Mbps. The first column of Table I is the available bandwidths for the four subflows. The second column provides the ratios of the available bandwidth among the subflows. The third and fourth columns are the goodputs when the TCP and UCP are used for data transmission between the pair of hosts. To make performance comparisons, the results shown in the last column come from directly applying the MPTCP at the two end hosts without using MMSGs in the middle.

Table I. Goodput (Mbps) when employing the UDP and TCP at the end hosts

Subflow Rates	Subflow Ratio	TCP over MPTCP	UDP over MPTCP	End-to-end MPTCP
10:10:10:10	1:1:1:1	36.5	37.3	38.4
4:8:12:16	1:2:3:4	34.0	34.7	35.7
2:4:8:26	1:2:4:13	25.9	26.3	26.9
1:2:4:33	1:2:4:33	16.4	16.9	17.3

We observed that the goodputs between the TCP over the MPTCP and UDP over the MPTCP are similar. Because the UDP does not have congestion, flow, and error controls, the results indicate that the control overhead generated by the MMSGs is insignificant when two end hosts use the TCP as their transport protocol. Although both end hosts and MMSGs have their own traffic controls, their control overhead can be ignored.

The gaps between the TCP over the MPTCP and the end-to-end MPTCP are also acceptable. The differences in goodput mainly come from the outer header introduced by the MMSG when IPoMP is applied. Part of the overhead comes from acknowledging a TCP acknowledgment packet. For example, when host 1 sends a TCP packet to host 2, host 2 has to return a pure acknowledgment packet to host 1 if no opportunity for piggybacking exists. However, the acknowledgment packet is encapsulated in an MPTCP packet at MMSG 2, and this packet is sent to MMSG 1 by an MPTCP subflow. Because MMSG 1 does not know that the packet is a TCP acknowledgment, it has to respond with an additional acknowledgment to MMSG 2 to indicate successful receipt of this packet. This process introduces additional overhead and consumes some bandwidth in the system.

Although the total available bandwidth of these four subflows is fixed at 40 Mbps, different distributions of the available bandwidths among the subflows have different goodputs. The results show that the larger the difference in available bandwidth among the subflows is, the smaller the total goodput. This phenomenon comes from the scheduling of the MPTCP. Similar results were also discovered in [21].

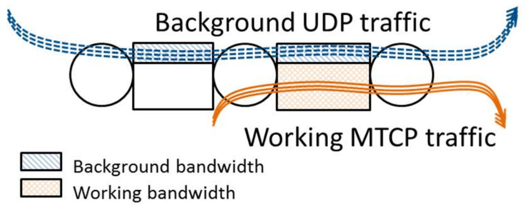
B. Bottleneck link with fixed available bandwidth

The bandwidth obtained for an MPTCP subflow is determined by the bottleneck link on the routing path. Three cases occur in the bottleneck link. As shown in Fig. 8(a), the bottleneck link of the MPTCP subflows contains background UDP traffic. Assume that k MPTCP subflows pass through the bottleneck link and the total throughput of these k subflows is r. Because UDP traffic does not have congestion control, when another new MPTCP subflow arrives at this link, the total available bandwidth for these k+1 subflows is still r.

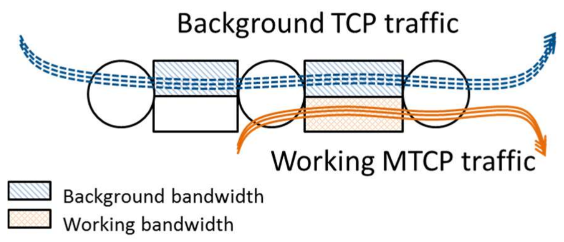
Figure 8(b) gives another case of the bottleneck link when the background traffic comes from TCP flows. Assume again that the total available bandwidth for the k MPTCP subflows in the link is r. Due to the congestion control of the background TCP flows, when another new MPTCP subflow is included in this link, the background TCP flows reduce their congestion windows. As a result, the total throughput for these k+1 MPTCP subflows becomes larger than r.

In the third case shown in Fig. 8(c), the MPTCP subflows share the bottleneck link with some background TCP traffic. However, unlike Case B, this link is not the bottleneck link for these background TCP traffic. As a result, the background traffic will not compete with the link bandwidth and leave a constant amount of bandwidth for MPTCP traffic. Consequently, when k+1 MPTCP subflows go through this link, the available bandwidth remains r, similar to Case A. However, if we introduce an increasing number of MPTCP subflows in this link, this link will eventually become the bottleneck for the background TCP traffic. The situation then becomes the same as in Case B shown in Fig. 8(b).

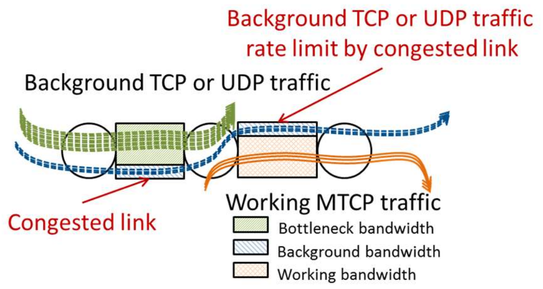
Because Case C can be decomposed into either Case A or Case B, in the following, we consider only Case A and Case B in our experiments.



(a) Case A scenario: The MPTCP shares the same bottleneck link as that of the background UDP



(b) Case B scenario: The MPTCP and background TCP have the same bottleneck link



(c) Case C Scenario: The MPTCP and background TCP have different bottleneck links

Fig. 8. Three cases of bandwidth sharing in the bottleneck link

The experimental network is the same as in Fig. 4(a). The enterprise consists of five branches. When the proposed system is applied, each branch has one MMSG that has two interfaces with which to connect to the public internet. The TC inside the MMSG is an EdgeCore AS4600-54T switch. The MPA is implemented in a Linux-based PC, in which we integrate our MPA program and the up-to-date open-source MPTCP program [22]. The OFS inside the MMSG is an OpenvSwitch 0. Because there are two physical ports in each MMSG, each end-to-end connection between a pair of hosts is supported by four MPTCP subflows.

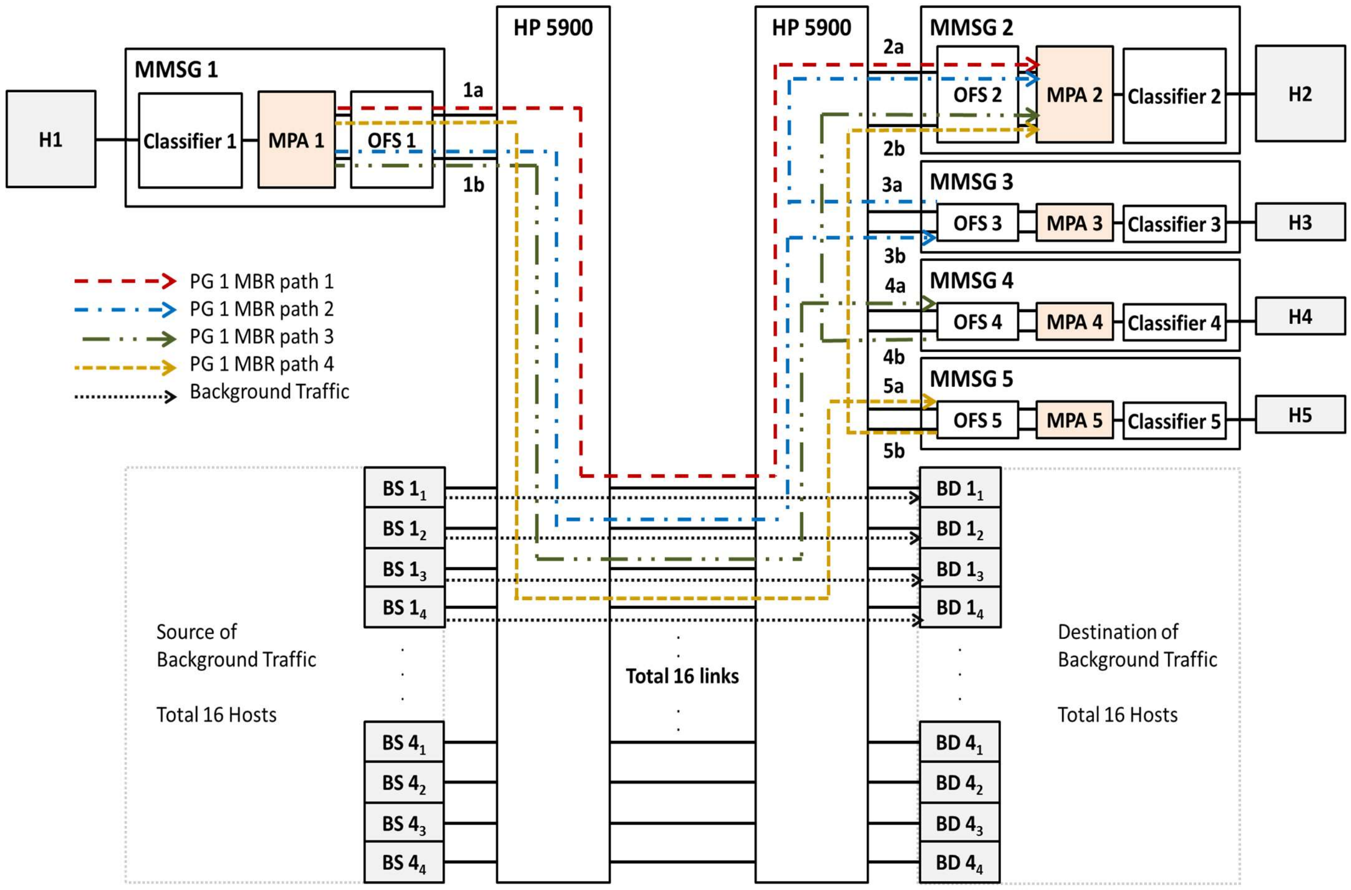


Fig. 9. Experimental network: only the routing paths of path group 1 are shown

The link rate between the OFS and the MPA is 1 Gbps. The port rate for accessing the public internet is limited to 40 Mbps. The SSC is another Linux PC running the Ryu 0 program to control all of the OFSs inside the network.

By using the transit branches, the overlay network provides multiple routing paths. Table II shows the routing paths on the overlay network between branch 1 and branch 2. A path group consists of four member paths, one for each subflow. For example, if an end-to-end flow is carried by path group 1, the routing paths for these four subflows are 1a>2a, 1a>4a>4b>2a, 1a>5b>5a->2b, 1b>2a, and 1b>3b>3a>2b. The first number in this notation denotes the branch ID, and the second character represents the port ID. For instance, 1a>4a>4b>2a means that the path consists of two WAN hops. In the first WAN hop, the subflow leaves from port *a* of branch 1 and arrives at port *a* of branch 4. In the second WAN hop, the flow leaves from port *b* of branch 4 and ends at port *a* of branch 2.

Table II. Configured routing paths

Mbr path	PG 1	PG 2	PG 3	PG 4
1	1a>2a	1a>2b	1b>2a	1b>2b
2	1b>3b>3a>2a	1b>3a>3b>2b	1a>3a>3b>2a	1a>3b>3a>2b
3	1b>4a>4b>2b	1b>4b>4a>2a	1a>4b>4a>2b	1a>4a>4b>2a
4	1a>5a>5b>2b	1a>5b>5a>2a	1b>5b>5a>2b	1b>5a>5b>2a

To emulate the bandwidth of the public internet, we use an OpenFlow network to interconnect the five branches. Figure 9 demonstrates the detailed configuration for the routing paths used by path group 1. We use an additional 16 hosts to introduce background traffic to the network. We emulate the Case A scenario of Fig. 8 by injecting UDP flows to remove the desired amount of bandwidth from the bottleneck link. For the Case B scenario of Fig. 8, we introduce various numbers of TCP flows to share the bottleneck link with the MPTCP subflows. Because Case C can be decomposed into either Case A or Case B, we do not take this case into consideration.

In this subsection, we examine the results of Case A and leave Case B for the next subsection. The physical link capacity on the bottleneck link is 100 Mbps. We consider four path groups, two path groups, and one path group between branch 1 and branch 2 in this set of experiments. Each path group includes four member paths, one per MPTCP subflow. The detailed routing for each path group is shown in Table II. To make performance comparisons, we also perform IP-in-IP tunneling between MMSGs in this set of experiments. When IP-in-IP tunneling is applied, only direct WAN-hop routing is used. For example, when path group 2 is selected for an end-to-end connection, the routings of the four subflows of the MPTCP are 1a>2b, 1b>3a>3b>2b, 1b>4b>4a>2a, and 1a>5b>5a>2a, while only the direct hop routing 1a>2b is used for IP-in-IP tunneling.

In this set of experiments, we consider the Case A scenario shown in Fig. 8(a). Each member path in a path group has the same available bandwidth. Table III shows the available bandwidth on the bottleneck links. Figure 10(a) presents the experimental results when four end-to-end TCP connections are generated from a host in branch 1 toward a host in branch 2. The MMSG at branch 1 uses a round robin to map each end-to-end connection to a path group. As a result, each end-to-end connection takes a different path group.

Table III. Available bandwidth (Mbps) on the bottleneck link
(a) Test cases for Fig. 10(a) and (b)

Test Case	PG 1	PG 2	PG 3	PG 4	Test Case	PG 1	PG 2	PG 3	PG 4
1	8	8	8	8	17	8	6	4	2
2	8	8	8	6	18	6	6	6	2
3	8	8	8	4	19	6	6	4	4
4	8	8	6	6	20	6	6	4	2
5	8	8	8	2	21	6	4	4	4
6	8	8	6	4	22	6	6	2	2
7	8	6	6	6	23	6	4	4	2
8	8	8	6	2	24	4	4	4	4
9	8	8	4	4	25	8	2	2	2
10	8	6	6	4	26	6	4	2	2
11	6	6	6	6	27	4	4	4	2
12	8	8	4	2	28	6	2	2	2
13	8	6	6	2	29	4	4	2	2
14	8	6	4	4	30	4	2	2	2
15	6	6	6	4	31	2	2	2	2
16	8	8	2	2					

(b) Test cases for Fig. 10(c) and (d)

Test Case	PG 1	PG 2
1	8	8
2	8	6
3	8	4
4	6	6
5	6	4
6	8	2
7	4	4
8	6	2
9	4	2
10	2	2

(c) Test cases for Fig. 10(e)

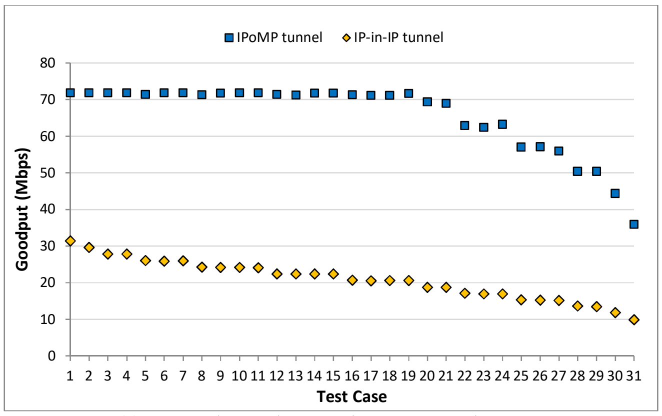
Test Case	PG 1	No. end-to-end TCP connections
1	8	4
2	6	4
3	4	4
4	2	4
5	8	8
6	6	8
7	4	8
8	2	8

We first observe that the goodput provided by the IPoMP tunnels is larger than that provided by the IP-in-IP tunnels. Please note that multiple cases exist in which the goodput is larger than 70 Mbps. Because the rate of each physical port is set to 40 Mbps, the maximum throughput between branch 1 and branch 2 is limited to 80 Mbps. A goodput larger than 70 Mbps representing the transmission rate reaches the port rate limitation. If we focus on the cases in which the goodputs are limited by the bottleneck links but not the physical capacity of the MMSG ports, the goodputs provided by the IPoMP tunnels are approximately four times that provided by the IP-in-IP tunnels.

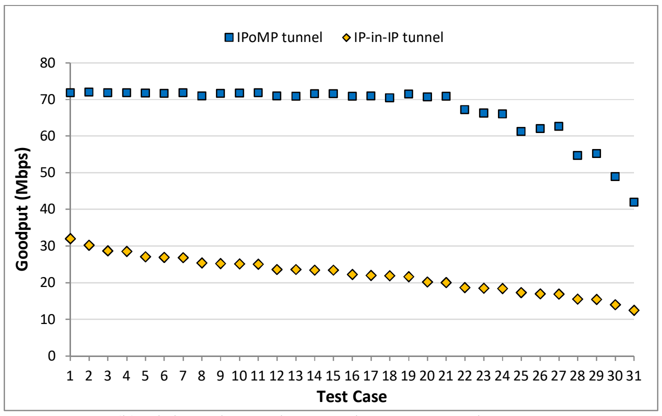
Figure 10(b) displays the results when 8 end-to-end TCP connections are applied. The round robin assignment results in each path group contain two end-to-end TCP connections. Comparing Fig. 10(a) with Fig. 10(b), we find that increasing the number of end-to-end TCP connections does not generate much difference in the total goodput.

We further examine the goodputs by using two path groups. The detailed settings are included in Table III(b). Reducing the number of path groups from four to two makes the physical port rate no longer the constraint for these test cases. Figure 10(c) and Fig. 10(d) display the results when two path groups are used. Comparing these two figures, we confirm again that increasing the number of end-to-end connections for the Case A scenario does not change the total goodputs.

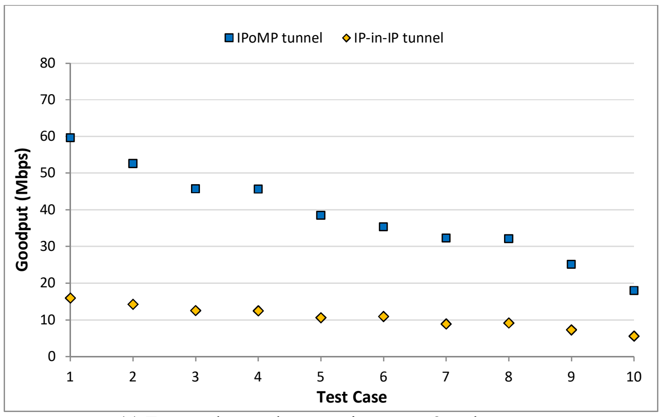
Figure 10(e) displays the results when only one path group is used. The setting of the experiments follows that of Table III(c). The first four test cases and the last four cases in Fig. 10(e) come from the results when four and eight end-to-end TCP connections are employed, respectively. Comparing Fig. 10(c) with the first four cases in Fig. 10(e), we find that the increase in goodput is approximately proportional to the increase in the number of routing paths. Similar results are also confirmed by comparing Fig. 10(d) with the last four test cases in Fig. 10(e).



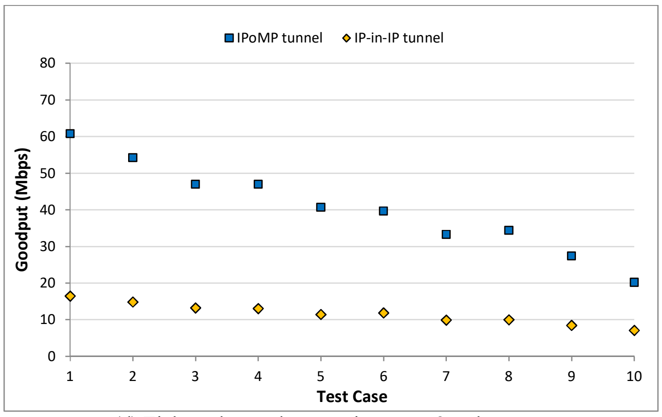
(a) Four end-to-end connections over 4 path groups



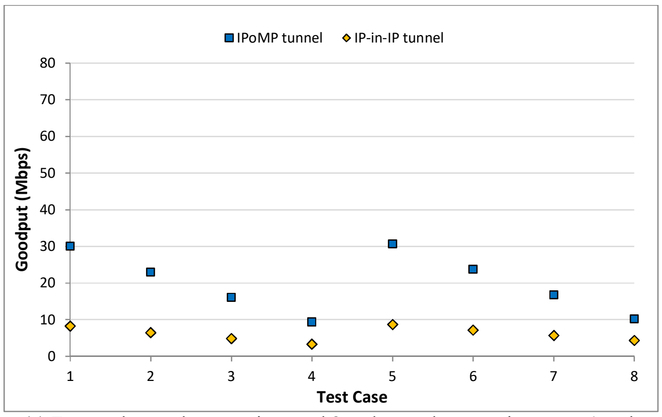
(b) Eight end-to-end connections over 4 path groups



(c) Four end-to-end connections over 2 path groups



(d) Eight end-to-end connections over 2 path groups



(e) Four end-to-end connections and 8 end-to-end connections over 1 path group

Fig. 10. Experimental results under Case A scenarios: SD-WAN traffic and background UDP traffic share bandwidth in the bottleneck link

C. *SD-WAN traffic and background TCP flows sharing the same bottleneck link*

In this set of experiments, we consider the Case B scenario shown in Fig. 8(b). We evaluate the goodput when the bottleneck link is shared between the SD-WAN traffic and background TCP traffic. Due to the congestion control of the TCP and MPTCP, the goodput of an end-to-end connection is different from those cases in the previous subsection.

We used the same network shown in Fig. 9 for this set of experiments. The bandwidth of the bottleneck link is 100 Mbps. We use the path groups shown in Table II for this set of experiments. They are the same as those used in the previous subsection. Table IV shows the number of background TCP connections on the bottleneck links. The number of background TCP connections is the same for each member path in a path group. These background TCP flows are generated from Iperf [24].

We first consider four path groups. Figure 11(a) displays the goodputs when four end-to-end TCP connections are set up between branch 1 and branch 2. The results reveal that the IPoMP tunnel outperforms the IP-in-IP tunnel in all 31 test cases.

Table IV. Number of background TCP connections on the bottleneck link

(a) Test cases for Fig. 11(a) and (b)

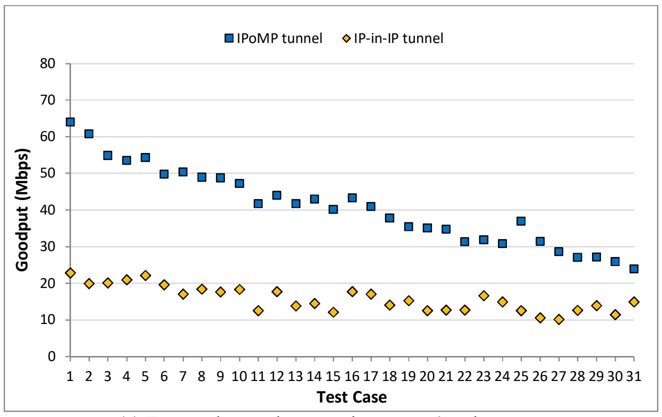
Test Case	PG 1	PG 2	PG 3	PG 4	Test Case	PG 1	PG 2	PG 3	PG 4
1	10	10	10	10	17	10	20	30	40
2	10	10	10	20	18	20	20	20	40
3	10	10	10	30	19	20	20	30	30
4	10	10	20	20	20	20	20	30	40
5	10	10	10	40	21	20	30	30	30
6	10	10	20	30	22	20	20	40	40
7	10	20	20	20	23	20	30	30	40
8	10	10	20	40	24	30	30	30	30
9	10	10	30	30	25	10	40	40	40
10	10	20	20	30	26	20	30	40	40
11	20	20	20	20	27	30	30	30	40
12	10	10	30	40	28	20	40	40	40
13	10	20	20	40	29	30	30	40	40
14	10	20	30	30	30	30	40	40	40
15	20	20	20	30	31	40	40	40	40
16	10	10	40	40					

(b) Test cases for Fig. 11(c) and (d)

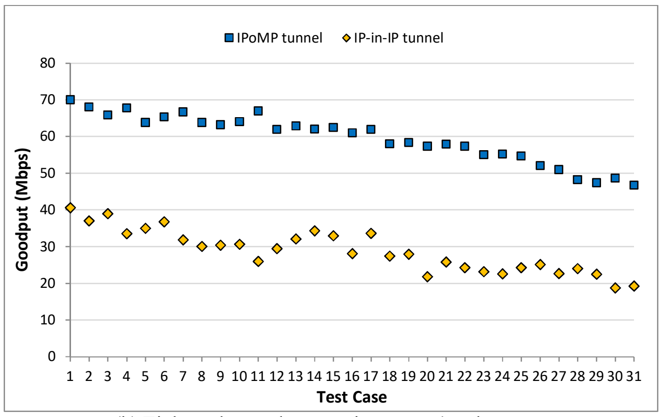
Test Case	PG 1	PG 2
1	10	10
2	10	20
3	10	30
4	20	20
5	20	30
6	10	40
7	30	30
8	20	40
9	30	40
10	40	40

(c) Test cases for Fig. 11(e)

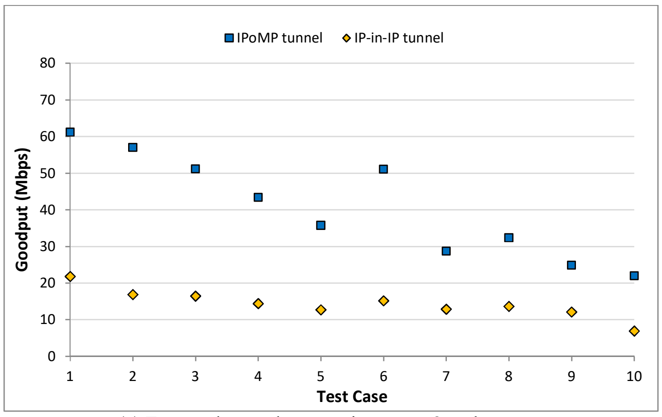
Test Case	PG 1	No. end-to-end TCP connections
1	10	4
2	20	4
3	30	4
4	40	4
5	10	8
6	20	8
7	30	8
8	40	8



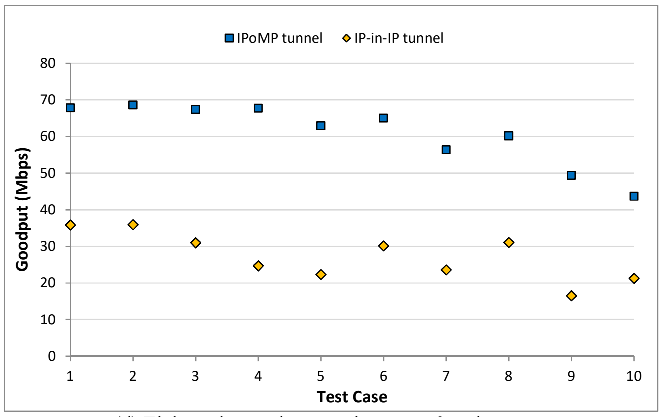
(a) Four end-to-end connections over 4 path groups



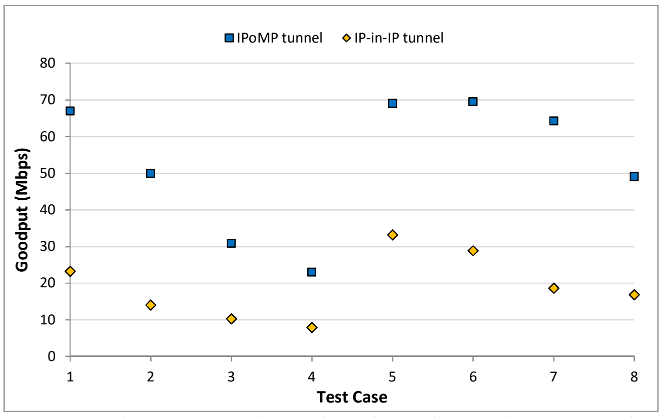
(b) Eight end-to-end connections over 4 path groups



(c) Four end-to-end connections over 2 path groups



(d) Eight end-to-end connections over 2 path groups



(e) Four end-to-end connections and 8 end-to-end connections over 1 path group

Fig. 11. Experimental results under Case B scenarios: SD-WAN traffic and background TCP traffic share bandwidth in the bottleneck link

We further increase the end-to-end connections from four to eight. The results are shown in Fig. 11(b). Comparing Fig. 11(a) with Fig. 11(b), we find that the total goodput in Fig. 11(b) is larger than the goodput shown in Fig. 11(a). The results meet our expectation that the congestion control of each individual background TCP flow reduces its bandwidth usage when the number of SD-WAN flows increases. Unlike the Case A scenarios shown in the previous subsection, where the total goodput remains constant regardless of how many MPTCP flows pass through the bottleneck link, in the Case B scenario, increasing the number of active SD-WAN flows on the bottleneck link can yield a larger total goodput.

Fig. 11(c) and Fig. 11(d) present the results when two path groups are applied, and Fig. 11(e) presents the results when only one path group is applied. We observed that the IPoMP tunnel outperforms the IP-in-IP tunnel in all of these test cases. When more path groups are used, the SD-WAN traffic can obtain more bandwidth under the same setting on the bottleneck links.

D. Different member paths in a path group have different available bandwidths

Table V. Available bandwidth (Mbps) on the bottleneck link

Test Case	Path 1	Path 2	Path 3	Path 4	Number of PGs
1	8	6	4	2	4
2	6	4	2	8	4
3	4	2	8	6	4
4	2	8	6	4	4
5	8	6	4	2	2
6	6	4	2	8	2
7	4	2	8	6	2
8	2	8	6	4	2
9	8	6	4	2	1
10	6	4	2	8	1
11	4	2	8	6	1
12	2	8	6	4	1

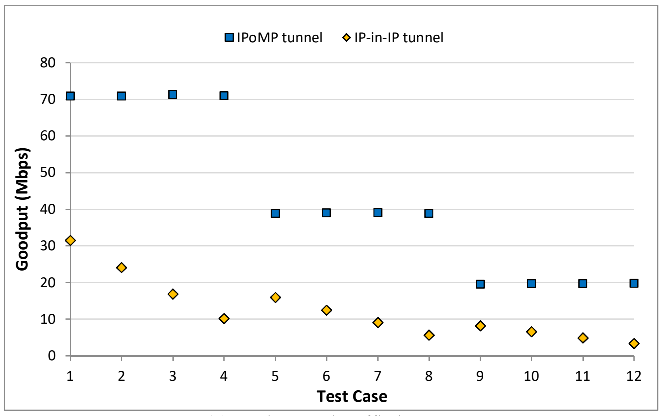
Table VI. Number of background TCP flows on the bottleneck link

Test Case	Path 1	Path 2	Path 3	Path 4	Number of PGs
1	10	20	30	40	4
2	20	30	40	10	4
3	30	40	10	20	4
4	40	10	20	30	4
5	10	20	30	40	2
6	20	30	40	10	2
7	30	40	10	20	2
8	40	10	20	30	2
9	10	20	30	40	1
10	20	30	40	10	1
11	30	40	10	20	1
12	40	10	20	30	1

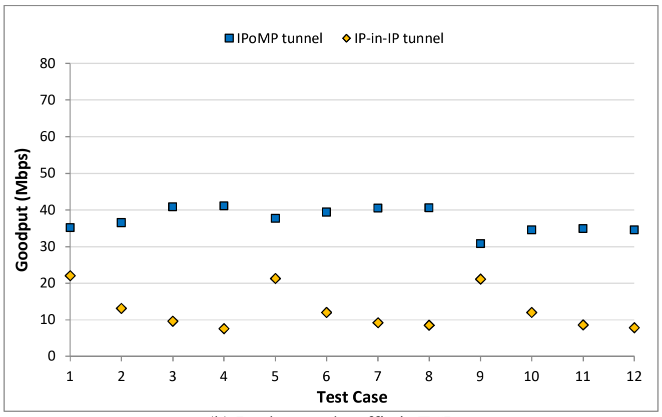
In the previous two subsections, the bottleneck link bandwidth is the same for every member path in a path group. In the final set of experiments, we evaluate the goodputs when the available bandwidth of each member path is different. The settings for the experiments are depicted in Table V and Table VI. The values in Table V are the available bandwidth in the bottleneck link when the background traffic is the UDP, and those in Table VI are the number of background TCP flows in the bottleneck link. For example, four path groups are used in test case 1 in Table V. The available bandwidths for the 1st, 2nd, 3rd, and 4th member paths in each of these four path groups are 8, 6, 4, and 2 Mbps, respectively. The detailed routing of each member path for each path group follows the configuration shown in Table II.

Four TCP end-to-end connections exist between the two hosts: one in branch 1 and the other in branch 2. MMSG 1 uses a round robin to map one end-to-end connection to one of the four path groups. For IPoMP, MMSG 1 uses four member paths in the selected path group to deliver its traffic. When the IP-in-IP tunnel is applied, only the member path that uses the direct

WAN hop in the path group is used.



(a) Background traffic is UDP



(b) Background traffic is TCP

Fig. 12. Experimental results when member paths in a path group have different available bandwidths

The settings that follow the Case A scenario are shown in Table V. Figure 12(a) shows the results. Because an MPTCP connection simultaneously distributes their packets on all four member paths of its assigned path group, it is not sensitive to the unequal available bandwidths among their member paths. On the other hand, IP-in-IP tunneling is very sensitive to the available bandwidth of the routing path. For example, in test cases 4, 8, and 12, the one-WAN-hop path has the least bandwidth in their path groups. This results in a very low goodput for IP-in-IP tunnels.

We further consider the Case B scenario. The bottleneck links are shared with other background TCP flows. The experimental results are shown in Fig. 12(b). Similar to the results shown in Fig. 12(a), the goodput of the IP-in-IP tunnels strongly depends on the routing path. The MPTCP counterparts have more stable goodputs. Jointly examining Fig. 12(a) and Fig. 12(b), we find that the goodput of IPoMP tunneling outperforms that of IP-in-IP tunneling in all test cases even when the available bandwidths on the bottleneck links of the routing paths are different.

VI. Conclusions

In this paper, we presented an architectural design and system implementation to realize the MMS system. To increase the throughput of the end-to-end connections, we developed multiple techniques to enable the MMS to perform multipath and multi-WAN-hop routing on top of the public internet. To eliminate the issue of out-of-order packet delivery in employing multipath routing, we design an MPTCP agent in the MMSG to realize IPoMP between end-to-end branches. In addition, by taking advantage of SDN technology, we design an IP address swapping technique in our MMSG that enables our MMS system to realize multi-WAN-hop routing. We also introduce prioritization in the MMSG to prevent the issue of traffic blocking.

We presented the detailed design and implementation of the MMSG and system controller. We examine the performance of the system in an experimental network. The experimental results show that the proposed IPoMP-based MMS VPN outperforms the conventional IP-in-IP tunneling-based VPN in all of our experimental cases. In many cases, the MMS can provide up to four times the end-to-end goodput of that provided by IP-in-IP tunneling.

In this work, we demonstrated the benefits of applying multipath and multi-WAN-hop routing for SD-WAN VPNs. The experimental results shown in this paper are based on static routing paths, and the scheduling for path group selection is based on a round robin. In fact, the throughput can be further improved if the paths are dynamically configured. In addition to polling the statistical data from the working paths that are already available in our SSC, to further enhance the network throughput of the SD-WAN, we need a tool to measure the available bandwidth of the public internet to explore a new path that has no working SD-WAN traffic on it yet.

We are currently developing a lightweight bandwidth monitoring and estimation module in the system controller. The bandwidth of a path can be measured by employing a tool such as PathLoad [25]. The output of this module will be used to reconfigure the routing paths. One more future work is to include IP security in the MMSG to facilitate enterprise transmission of sensitive commercial data through the public internet.

References

[1] "Google Compute Engine – IaaS," https://cloud.google.com/compute/.
[2] "Microsoft Azure Cloud Computing Platform & Services, " http://azure.microsoft.com/.
[3] "Facebook's new long-haul network," https://engineering.fb.com/data-center-engineering/building-express-backbone-facebook-s-new-long-haul-network/
[4] C.-Y. Hong et al., ``Achieving High Utilization with Software-drivenWAN," in Proc. ACM SIGCOMM, 2013, pp. 15-26.
[5] S. Jain et al., ``B4: Experience with a Globally-deployed Software Defined WAN," in Proc. ACM SIG*COMM*, 2013, pp. 3-14.
[6] A. Kumar et al., ``BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in Proc. ACM SIGCOMM, 2015, pp. 1-14.
[7] H. Yan, Y. Li, W. Dong, and D. Jin, "Software-Defined WAN via Open APIs," IEEE Access, vol. 6, 2018, pp. 33752-33765.
[8] Behzad Mirkhanzadeh, Naeim Taheri, and Siavash Khorsandi, "SDxVPN: A Software-Defined Solution for VPN Service Providers," in *Proc. IEEE/IFIP NOMS*, 2016.

[9] Mohammad Mousa, Ayman M. Bahaa-Eldin, and Mohamed Ali Sobh, “Autonomic management of MPLS backbone networks using SDNs,” in *Proc. IEEE ICCES,* 2017.

[10] C. H. Benet et al., “Policy-based Routing and Load Balancing for EVPN-based Data Center Interconnections,” in Proc. IEEE NFV/SDN, 2017.

[11] “List of SD-WAN Vendors,” https://packetpushers.net/virtual-toolbox/list-sd-wan-vendors/.

[12] “Cisco – SD-WAN,” https://www.cisco.com/c/en_sg/solutions/enterprise-networks/sd-wan/index.html.

[13] "SD-WAN: What it is and why you'll use it one day". networkworld.com, https://www.networkworld.com/article/3031279/sd-wan-what-it-is-and-why-you-ll-use-it-one-day.html.

[14] A. Ford et al., “TCP Extensions for Multipath Operation with Multiple Addresses,” *IETF RFC 6824*.

[15] L. Chaufournier, A. Ali-Eldin, P. Sharma, P. Shenoy, and D. Towsley, “Performance Evaluation of Multi-Path TCP for Data Center and Cloud Workloads,” in *Proc. ICPE '19.*

[16] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley,” Improving Datacenter Performance and Robustness with Multipath TCP,” in *Proc. ACM SIGCOMM*, 2011.

[17] R. v. d. Pol et al., “Multipathing with MPTCP and OpenFlow,” in *Proc. High Perform. Comput., Netw., Storage Anal.* (*SCC*), Nov. 2012, pp. 1617-1624.

[18] S. S. W. Lee, K.Y.Li, K.Y.Chan, J. H. YwiChi, T.-W. Lee, W.-K. Liu, and Y.-J. Lin, “Design of SDN based Large Multi-tenant Data Center Networks,” in *Proc. IEEE CloudNet*, Oct. 2015.

[19] B. Sonkoly, F. Németh, L. Csikor, L. Gulyás, and A. Gulyás, “SDN based Testbeds for Evaluating and Promoting Multipath TCP,” in *Proc. IEEE ICC*, 2014, pp. 3044–3050.

[20] Ryu SDN Framwork. [Online]. Available: https://osrg.github.io/ryu/.

[21] Soonghwan Ro and Dien Nguyen Van, “Performance Evaluation of MPTCP over a Shared Bottleneck Link,” International Journal of Computer and Communication Engineering, vol. 5, no. 3, 2016.

[22] MultiPath TCP - Linux Kernel implementation. [Online]. Available: https://www.multipath-tcp.org/

[23] Open vSwitch. [Online]. Available: http://openvswitch.org/

[24] Iperf. [Online]. Available: https://iperf.fr

[25] M. Jain and C. Dovrolis, “End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput,” IEEE/ACM Transacations on Networking, vol. 11, no. 4, 2003.