QoS-Aware Flexible Traffic Engineering with OpenFlow-Assisted Agile IP-Forwarding Interchanging

Shoujiang Ma, Daoyun Hu, Shengru Li, Nana Xue, Suoheng Li, Yan Shao, Zuqing Zhu†
School of Information Science and Technology, Key Laboratory of Electromagnetic Space Information, CAS,
University of Science and Technology of China, Hefei, China
†Email: {zqzhu}@ieee.org

***Abstract*—In this paper, we propose to use IP-forwarding interchanging (*i.e.*, exchanging packets between IPv4 and IPv6 according to the network status) enabled by OpenFlow to realize quality-of-service (QoS) aware flexible traffic engineering (F-TE) in a hybrid network where IPv4 and IPv6 coexist. Specifically, we have different IP domains interconnected by OpenFlow switches managed by a centralized controller, and design the network system to facilitate online, adaptive and per-flow-based IP-forwarding interchanging for link utilization optimization with the considerations on applications' QoS requirements. We implement the design in a semi-practical network testbed, and demonstrate the advantages of F-TE with experiments that include simultaneous video streaming and file transfer.**

***Index Terms*—OpenFlow, IPv6, Flexible traffic engineering.**

I. Introduction

Over the last few decades, Internet protocol (IP) networking has made great success for providing people a convenient, robust and economical way to access the Internet. However, with the fast development of Internet, the numbers of network devices and emerging applications are growing very fast, which brings a lot of challenges to the current IP infrastructure. For instance, the Internet Assigned Numbers Authority (IANA) assigned its last available IPv4 address-pool in February 2011 [1], which means that in the near future, new network devices may have difficulties to acquire IPv4 addresses for accessing the Internet. In order to resolve this addressing issue, IPv6 has been proposed, developed and deployed [2].

Nevertheless, the transition from IPv4 to IPv6 cannot be carried out with the green-field approach, given the fact that it would be technically and economically impossible for the Internet (a globally-distributed network) to upgrade its IP addressing scheme in one shot. Hence, the transition has to be conducted progressively, and this will make IPv4 and IPv6 coexist for a relatively long period of time [3]. For a smooth transition, one needs the technologies that can support efficient inter-operation of IPv4- and IPv6-devices when the underlying network architectures are not all IPv4- or IPv6-capable. In line of this, the Internet engineering task force (IETF) [4] has recorded and standardized numbers of mechanisms to facilitate IPv4-IPv6 inter-operability [5]. Note that during the IPv4-to-IPv6 transition, numbers of IPv4- or IPv6-capable "islands" can emerge in the Internet and bring new challenges to traffic engineering. For instance, IPv4 traffic will be forwarded through IPv4-islands as long as they are interconnected seamlessly, even though alternative IPv6-islands could be less congested and would provide better quality-of-service (QoS) guarantee and traffic engineering mechanism.

Recently, software-defined networking (SDN) has been proposed to make a network programmable, dynamic and application-aware by decoupling its data and control planes [6]. As a possible implementation of SDN, OpenFlow has been developed as an open standard protocol [7], which leverages flow-based switching and enables software-defined routing, forwarding and managing by using a centralized controller. Considering the enhanced programmability and flexibility, we expect SDN/OpenFlow to provide Internet Service Providers (ISPs) a low-cost solution for the IPv4-to-IPv6 transition. For instance, Xia *et al.* have recently demonstrated a software-defined approach to unify the deployment of IPv6 in a cost-effective way [8]. More importantly, SDN/OpenFlow can be utilized to improve the performance of traffic engineering during the transition, as its centralized control plane and flexible data plane allow us to optimize link utilization in a QoS-aware and network-wide manner.

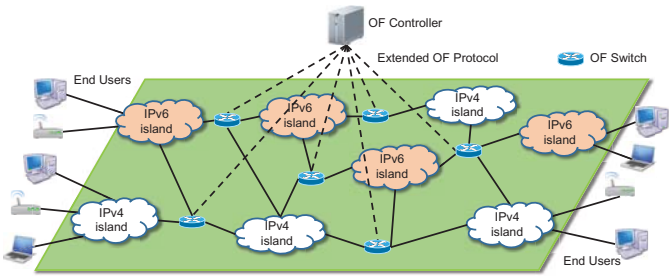


Fig. 1. Overall network architecture for QoS-aware F-TE.

In this paper, we utilize IP-forwarding interchanging enabled by OpenFlow to realize QoS-aware flexible traffic engineering (F-TE) in a network that consists of multiple IPv4- and IPv6-islands. Specifically, we have the "islands" interconnected by OpenFlow switches managed by a centralized controller, and design the OpenFlow system to facilitate online, adaptive and per-flow-based IP-forwarding interchanging (*i.e.*, exchang-

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

ing packets between IPv4 and IPv6 dynamically according to the network status) for link utilization optimization. We design the overall system architecture for realizing F-TE, implement the design in a semi-practical network testbed, and conduct experiments with simultaneous video streaming and file transfer to demonstrate the advantages of F-TE.

The rest of the paper is organized as follows. Section II describes the operation principle and overall architecture of the network system for F-TE. We show the experimental demonstrations in Section III. Finally, Section IV summarizes the paper.

II. Operation Principle and System Architecture

A. Overall Network Architecture

Fig. 1 shows the overall network architecture for realizing QoS-aware F-TE with agile IP-forwarding interchanging. The network consists of multiple IPv4- and IPv6-islands, and in each island, we have IPv4- or IPv6-capable routers for routing and forwarding. Instead of inserting traditional transition gateways (*e.g.*, those use the simple internet transition (SIT) protocol [9]) among the islands, we propose to connect them with OpenFlow (OF) switches that are managed by a centralized controller. The OF controller obtains the overall network status, *i.e.*, by setting up simple network management protocol (SNMP) connections with routers in the islands or by counting the packet loss rate and/or latency through each island, and regulates packet transfer accordingly.

The OF switches are designed to facilitate IP-forwarding interchanging for QoS-aware F-TE. Specifically, based on the instructions (*i.e.*, flow-entries) from the OF controller, an OF switch determines whether a packet flow's next-hop is an IPv4- or IPv6-island, and then performs the IP-forwarding interchanging accordingly. The IP-forwarding interchanging is realized with IPv4/IPv6 tunneling, where the OF switch inserts an IPv6 header and encapsulates the IPv4 packet in an IPv6 one, or *vice versa*. Then, F-TE is realized with the cooperation between the OF controller and switches, which is based on an extended OF protocol. Basically, based on the overall network status, the controller instructs the switches to establish/switch the forwarding path of each packet flow to satisfy different QoS requirements and improve link utilization.

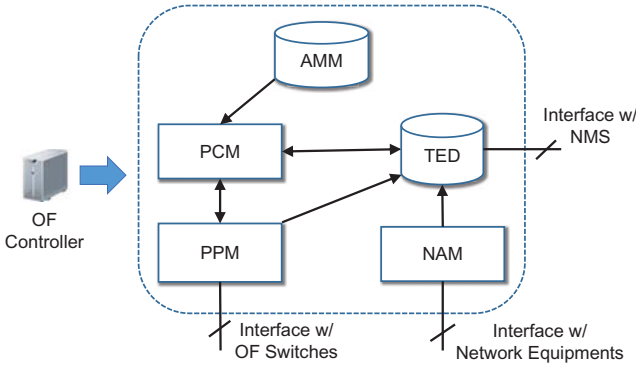


Fig. 2. Proposed structure of OpenFlow controller.

B. OpenFlow Controller

Fig. 2 illustrates the proposed structure of OF controller that can support IP-forwarding interchanging for F-TE. The details of the functional modules are as follows.

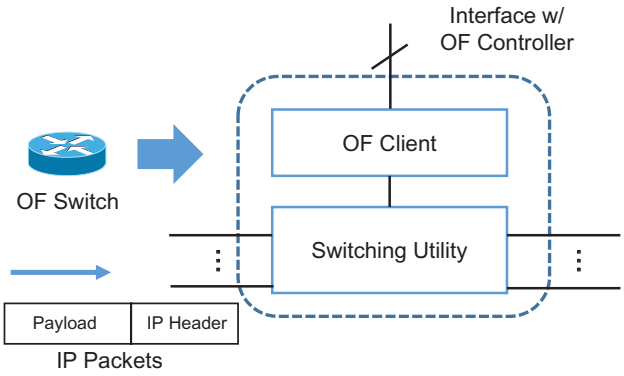


Fig. 3. Structure of the OpenFlow switch.

- **PPM (Path Provision Module)**: It interacts with OF switches and the path computation module (PCM) to establish/switch the forwarding path of a packet flow. For example, during the path setup, PPM receives the *Packet-In* message from the ingress switch and forwards the information to PCM for path computation. After obtaining feedback from PCM, it encodes the flow-entries in *Flow-Mod* messages and sends them to related switches for installing the path. Besides these tasks, PPM also communicates with the traffic engineering database (TED) to manage the records of packet flows.
- **PCM (Path Computation Module)**: It receives path-computation tasks from PPM and optimizes each packet flow' forwarding path to achieve QoS-aware F-TE. Upon receiving the tasks, PCM obtains the current network status from TED, and when the path-computations are done, it instructs PPM to build the corresponding flow-entries. Meanwhile, PCM also monitors the network proactively by checking TED periodically and when a path switching is needed, it calculates the new forwarding path and instructs PPM to implement the path switching.
- **TED (Traffic Engineering Database)**: It stores the current network status, including active OF switches, edge nodes of each IPv4- or IPv6-island, connectivity among the network devices, bandwidth usage on each inter-island link, and status of each island. The network abstraction module (NAM) and PPM update TED in real-time to ensure that it contains the most-updated information. The operator can also retrieve information from TED through an external network management system (NMS).
- **NAM (Network Abstraction Module)**: It collects the network's topology information, abstracts the data plane equipments, and updates the network status in TED proactively. For instance, NAM monitors the edge nodes of an island and when the island becomes congested, it will update the information accordingly in TED.

TABLE I
OPENFLOW EXTENSIONS ON FLOW-MATCHING ACTIONS

Action Name	Detailed Operation
PUSH_IPv6	Add an IPv6 header to an IPv4 packet and implement IPv6 tunneling.
POP_IPv6	Remove the IPv6 header for tunneling and recover an IPv4 packet.
MOD_IPv6	Modify the source and destination addresses of an IPv6 packet.
PUSH_IPv4	Add an IPv4 header to an IPv6 packet and implement IPv4 tunneling.
POP_IPv4	Remove the IPv4 header for tunneling and recover an IPv6 packet.

- **AMM (Address Mapping Module)**: It manages the IPv4/IPv6 address mapping and assists PCM to implement IPv4/IPv6 tunneling. Note that for a more scalable design, one can implement AMM as an independent module outside the OF controller.

C. OpenFlow Switch

We implement the OF switch with OpenvSwitch [10], and propose necessary extensions to support IP-forwarding interchanging for F-TE, *i.e.*, designing five new flow-matching actions as shown in Table I. The structure of the OF switch is illustrated in Fig. 3. The OF client communicates with the OF controller with the extended OF protocol, parses flow-entries, and configures the switching utility accordingly. Packet forwarding is done in the switching utility, which handles packets based on the installed flow-entries and invokes IP-forwarding interchanging when necessary.

D. OpenFlow-Assisted Agile IP-Forwarding Interchanging

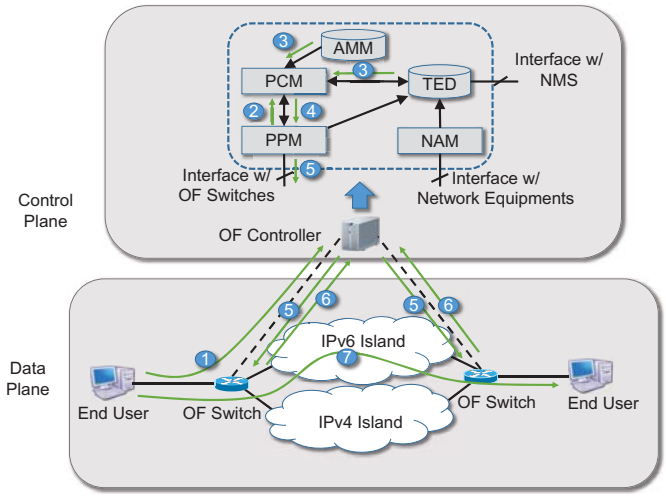


Fig. 4. OpenFlow-assisted agile IP-forwarding interchanging.

Fig. 4 uses an intuitive example to explain the procedure of OF-assisted IP-forwarding interchanging, where we need to switch a packet flow from IPv4 to IPv6 for QoS-aware F-TE.

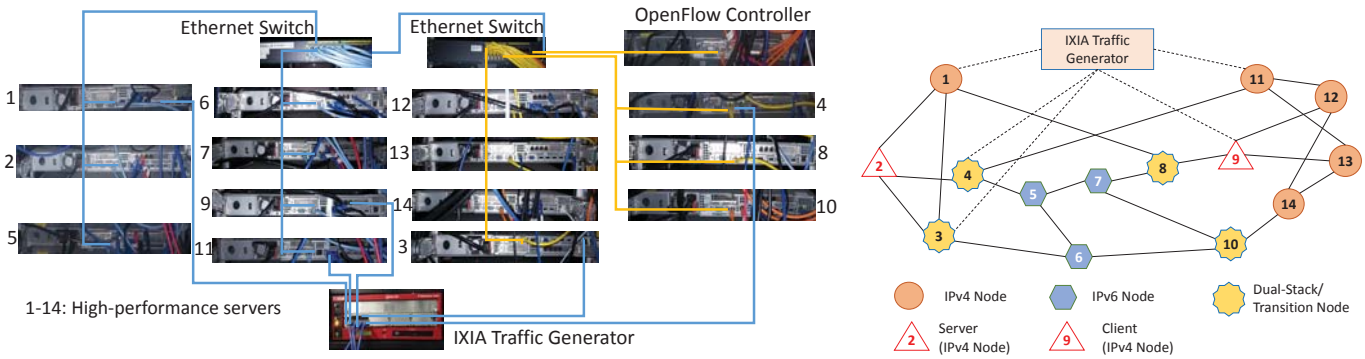
- **Step 1**: When the flow's first IPv4 packet arrives at the ingress OF switch, the switch checks and finds that there is no flow-entry installed for this flow. Then, the OF client in it sends a *Packet-In* message to the OF controller.
- **Step 2**: PPM in the controller receives the *Packet-In* message and instructs PCM to perform path-computation.
- **Step 3**: PCM obtains the current network status from TED, and requests the source and destination IPv6 addresses for IPv6 tunneling from AMM. Since the IPv6 island is less congested, PCM decides to forward the flow to it and invokes an IP-forwarding interchanging.
- **Step 4**: PCM instructs PPM to build the corresponding flow-entries for the two related switches (*i.e.*, the ingress and egress ones).
- **Step 5**: PPM encodes the flow-entries in *Flow-Mod* messages and distributes them to the related switches. The OF client in each switch parses the flow-entry and configures the flow table for switching utility accordingly.
- **Step 6**: The two related switches return their configuration results to the controller using *Barrier-Reply* messages. If the flow's forwarding path is set up correctly, PPM updates TED to include the provisioned flow. Otherwise, PPM invokes an error-recovery mechanism.
- **Step 7**: The ingress switch encapsulates the flow's IPv4 packets in IPv6 payloads (*i.e.*, IPv6 tunneling), and forwards them to its output port connecting to the IPv6 island. In the IPv6 island, the packets get forwarded towards the egress switch, where they are converted back to IPv4 for subsequent processing.

On the other hand, if PCM decides that the flow's next-hop is the IPv4 island, the ingress switch will just forward the packets to the corresponding output port without IP-forwarding interchanging. Note that the fundamental difference between this OF-assisted F-TE system and the traditional IPv4/IPv6 transition gateway is that the proposed one can facilitate online and adaptive IP-forwarding interchanging on a per-flow basis. For instance, when PCM observes a network status change that makes the IPv4 island a better next-hop for the flow, the controller will instruct the switches to perform a path switching for it, while the forwarding actions for the rest flows that go through the same ingress switch are unaffected.

III. EXPERIMENTAL DEMONSTRATIONS

A. Experimental Setup

We implement the proposed OF system with high-performance Linux servers (Lenovo ThinkServer RD540), and set up the network testbed as illustrated in Fig. 5(a). Specifically, we have 14 stand-alone servers connecting with each other as the data plane. Each server equips with multiple network interface cards (NICs) and can function as an IP router or an OpenFlow switch, depending on the networking programs running on it. For IP routing, we program the Linux system to make the server operate as an IPv4 router, an IPv6 router or a dual-stack router. If we need an OpenFlow switch, the server runs OpenvSwitch. The OpenFlow controller is realized with the POX platform and it also runs on an independent Linux server. Fig. 6 shows the web-based graphical user interface



(a) Network testbed. (b) Network topology.

Fig. 5. Experimental Setup.

(GUI) provided by the OpenFlow controller, with which we can obtain the information regarding network topology, network nodes, OpenFlow messages and active packet flows. We incorporate an IXIA traffic generator in the testbed for injecting background traffic in the network and generating congestions. The detailed topology of the testbed is shown in Fig. 5(b), where each node represents a stand-alone server that operates as either an IP router or an OF switch. More specifically, *Nodes* 1, 2, 9 and 11-14 are IPv4 routers, *Nodes* 5-7 are IPv6 routers, and the rest nodes can be either dual-stack IP routers or OpenFlow switches, depending on the experimental scenario. In the experiments, we consider the communication between *Nodes* 2 and 9 such that we run the application servers on *Node* 2 and put the clients on *Node* 9.

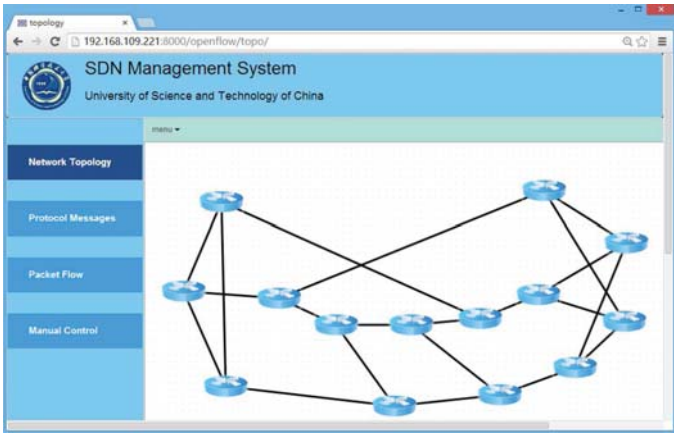


Fig. 6. SDN Management System Web GUI.

In order to demonstrate the advantages of QoS-aware F-TE and evaluate its performance, we consider the coexistence of video streaming and file transfer services in a network where there is dynamic and unpredicted background traffics, and design three experimental scenarios as follows.

- *Scenario* 1 (Without TE): There is no traffic engineering (TE), and both the video streaming and file transfer use the shortest path between *Nodes* 2 and 9 in Fig. 5(b).
- *Scenario* 2 (Conventional TE): Conventional dual-stack IP routers are used on *Nodes* 3, 4, 8 and 10. Basically, we can switch the flows' paths for TE while the end-to-end IP-forwarding scheme (IPv4 or IPv6) cannot be changed, *i.e.*, IP-forwarding interchanging is not supported.
- *Scenario* 3 (F-TE): OF switches that support IP-forwarding interchanging are implemented on *Nodes* 3, 4, 8 and 10, and QoS-aware F-TE is fully supported.

For the video streaming, we use a 60-second H.264 video sequence that is encoded as 1080P. Specifically, on *Node* 2, the video is encapsulated with IP/UDP/RTP headers and sent out to *Node* 9, while on *Node* 9, the video client receives and plays it back for performance evaluation. Meanwhile, the file transfer is realized by using Iperf [11] to generate TCP packets and send them from *Node* 2 to *Node* 9. Since the video streaming has stricter QoS requirements on transmission latency and jitter, we provision it with high priority in the experiments, *i.e.*, the file transfer should give way to it when there is congestion.

B. Experimental Results and Discussion

Fig. 7 shows the experimental results on end-to-end delay of the video streaming from the three experimental scenarios. Apparently, *Scenario* 3 (F-TE) achieves the best results on delay, because it can avoid congested links with agile IP-forwarding interchanging (*i.e.*, packets can be changed from IPv4 to IPv6, or *vice versa* adaptively) unless the IPv4- and IPv6-islands are all congested, *e.g.*, during $[45, 47]$ seconds. It can be seen that the delay results from *Scenario* 1 (Without TE) are the worst, because once the shortest path ($2\rightarrow1\rightarrow8\rightarrow9$) is congested, the video packets will suffer from severe queuing delay increase due to the lack of TE. For *Scenario* 2 (Conventional TE), since the priority of video streaming is higher than that of file transfer, the video packets can be switched to use path $2\rightarrow4\rightarrow11\rightarrow13\rightarrow9$ with TE, when the shortest path is congested. However, the prolonged delay cannot be avoided anymore when the whole IPv4 domain becomes crowded, *e.g.*, when both *Links* $1\rightarrow8$ and $4\rightarrow11$ are congested during $[33, 47]$ seconds. Note that all of the results on delay also possess some irregular spurs that follows the similar trend, which we believe are caused by the operating

systems on *Nodes* 2 and 9. The results on video streaming bandwidth are plotted in Fig. 8, which show the similar trend as those in Fig. 7 and further verify the advantage of F-TE.

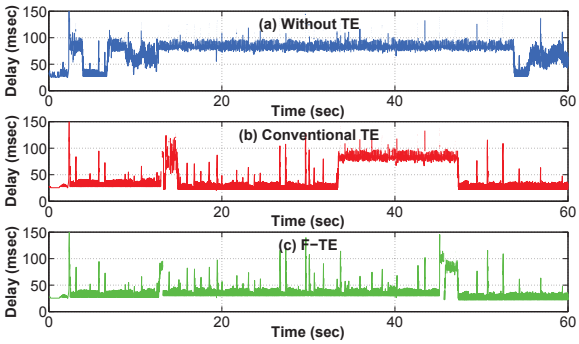


Fig. 7. Results on packet-level end-to-end delay of video streaming.

Fig. 9 illustrates whether the received video frames can be instantly decoded in the three scenarios. A vertical red bar means that a frame is not decodable due to severe packet loss. The results on the luminance component's peak signal-to-noise ratio (Y-PSNR) for the received video are shown in Fig. 10. Note that there is one dip on the Y-PSNR curves from both F-TE and Conventional TE during $[10, 20]$ seconds. We believe that is caused by the path switchings in the period. Also, we notice that the durations of the Y-PSNR dips from Conventional TE are longer than those from F-TE. This is because when the forwarding path of video streaming becomes congested, Conventional TE can only switch it to another one that is still within the IPv4 domain and this causes that the video and file packets share the same path and compete for transmission opportunities, which makes the severe-frame-loss period last longer. Fortunately, the issue discussed above will not happen for F-TE, since it can easily switch the video packets to use a less crowded path that goes through the IPv6 domain. Fig. 11 includes two screen-shots captured for the received videos, which illustrates the actual video play-back qualities. Moreover, the statistics of the experimental results are given in Table II. In all, with all these experimental results, we can clearly see that F-TE provides the best video streaming performance among the three scenarios.

Fig. 12 is captured by wireshark for the video packets that go through IP-forwarding interchanging, which shows that the operation is performed on a per-flow basis as expected. The results on bandwidth used for the file transfer are shown in Fig. 13. We observe that among the three scenarios, F-TE still provides the largest file transfer bandwidth, which means that it utilizes the link resources for different applications in the best way. However, we also notice that when the network becomes relatively crowded after 34 seconds, the file transfer bandwidth from F-TE also shrinks. This is because the file transfer's priority is lower and has to give way to video streaming in such situation, to ensure that the strict QoS requirement from it can be satisfied. Hence, the results in Fig. 13 confirm that the proposed F-TE is QoS-aware.

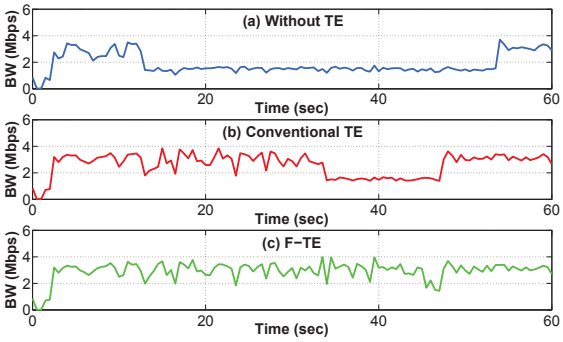


Fig. 8. Results on video streaming bandwidth at the client side.

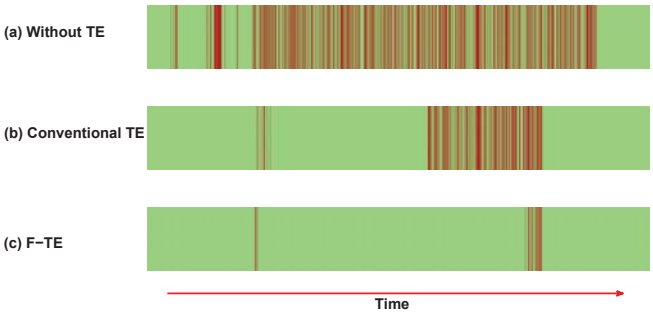


Fig. 9. Results on video frames' decodability.

TABLE II
STATISTICS OF EXPERIMENTAL RESULTS.

		Without TE	Conventional TE	F-TE
Delay (msec)	Average	75.4	40.9	37.1
	Variance	21.2	22.7	13.6
Bandwidth (Mbps)	Average	1.85	2.60	2.93
	Variance	0.77	0.85	0.69
Link Utilization (%)	Average	37.1	51.9	58.5
Frame Loss (%)	Average	31.0	11.0	1.2
Y-PSNR (dB)	Average	–	38.75	46.48

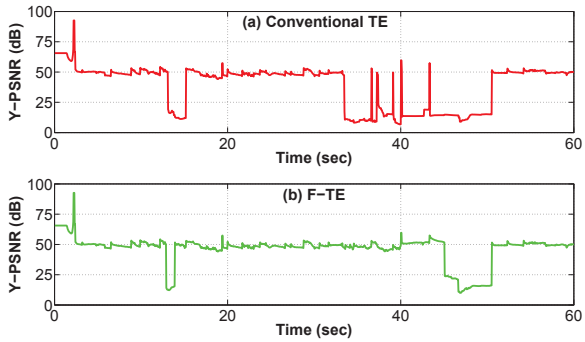


Fig. 10. Results on received videos' Y-PSNRs.

(a) Conventional TE (b) F-TE

Fig. 11. Screen-shots of video play-backs.

IV. Conclusion

This paper proposed to use IP-forwarding interchanging enabled by OpenFlow to realize QoS-aware F-TE in a network that consists of multiple IPv4- and IPv6-islands. We designed the overall system architecture, implemented the design in a semi-practical network testbed, and demonstrated the advantages of F-TE with experiments that included simultaneous video streaming and file transfer.

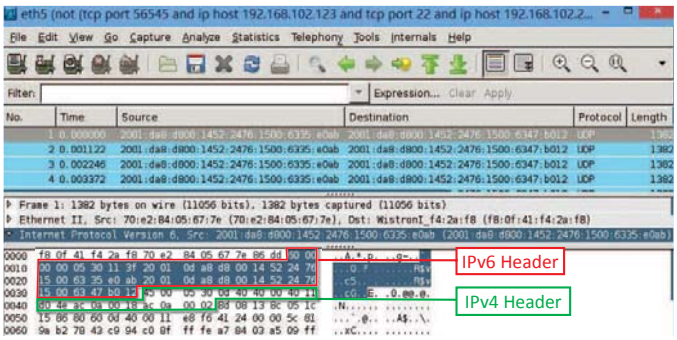


Fig. 12. Video packets that go through IP-forwarding interchanging.

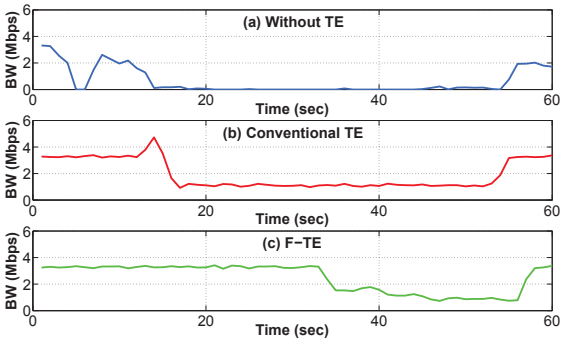


Fig. 13. Results on file transfer bandwidth.

Acknowledgments

This work was supported in part by the Project NCET-11-0884, the NSFC Project 61371117, the Fund WK2100060010, the SPR Program of CAS (XDA06030902), Natural Science Research Project for Universities in Anhui (KJ2014ZD38), and the state key laboratory of advanced optical communication systems networks, China.

References

[1] Internet Assigned Numbers Authority (IANA). [Online]. Available: https://www.iana.org/
[2] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," *RFC 2460*, Dec. 1998.
[3] G. Hu *et al.*, "A general framework of source address validation and traceback for IPv4/IPv6 transition scenarios," *IEEE Network*, vol. 27, pp. 66–73, Nov./Dec. 2013.
[4] Internet Engineering Task Force (IETF). [Online]. Available: http://www.ietf.org/
[5] P. Wu *et al.*, "Transition from IPv4 to IPv6: A state-of-the-art survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 1407–1424, Third Quarter 2013.
[6] G. Goth, "Software-defined networking could shake up more than packets," *IEEE Internet Comput.*, vol. 15, pp. 6–9, Jul./Aug. 2011.
[7] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *Comput. Commun. Rev.*, vol. 38, pp. 69–74, Feb. 2008.
[8] W. Xia *et al.*, "A software defined approach to unified IPv6 transition," in *Proc. ACM SIGCOMM 2013*, pp. 547–548, Oct. 2013.
[9] E. Nordmark and R. Gilligan, "Basic transition mechanisms for IPv6 hosts and routers," *RFC 4213*, Oct. 2005.
[10] OpenvSwitch. [Online]. Available: http://openvswitch.org/
[11] Iperf - the TCP/UDP bandwidth measurement tool. [Online]. Available: https://iperf.fr/