Wireless Pers Commun (2016) 86:1829–1843
DOI 10.1007/s11277-015-3145-0

BTSDN: BGP-Based Transition for the Existing Networks to SDN

Pingping Lin[1,2] · Jun Bi[1,2] · Hongyu Hu[1,2]

Published online: 18 December 2015

Abstract Currently, the architecture of network device is closed. This is detrimental to the network innovation. Software defined networking (SDN) (McKeown 2009) decouples the vertically coupled architecture, and reconstructs the Internet as a modular structure. The idea of SDN is widely accepted by both of the academic and industry researchers, and is considered as a promising way to re-architect the Internet. OpenFlow as a typical instance of the SDN has been deployed by many universities and research institutions all over the world. However, the research and technologies on transitioning the existing networks to SDN are not mature. So this paper takes the instance OpenFlow for example and proposes a simple and practical Border Gateway Protocol (BGP) based transition solution named BTSDN. BTSDN fully explores the characteristics of OpenFlow network, proposes to continue using the current BGP protocol, and retains the legacy BGP border routers to connect the OpenFlow network with the rest of Internet. The experiment at the end of this paper preliminary verifies the feasibility of BTSDN.

Keywords Software defined networking · OpenFlow · BGP · Transition

✉ Jun Bi
junbi@tsinghua.edu.cn

Pingping Lin
pingpinglin2012@gmail.com

Hongyu Hu
huhongyu@cernet.edu.cn

[1] Department of Computer Science, Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China

[2] Tsinghua National Laboratory for Information Science and Technology, Room 4-204, FIT Building, Beijing 100084, China

1 Introduction

Currently, the architecture of the network device is closed. This is not favor of network innovation. Protocols related to the core network lay or core network device are hard to apply. Software defined networking (SDN) [1] decouples the vertical and tight coupled architecture, and reconstructs the Internet as a modular structure. At the same time, it opens up the control plane and the protocol implementation in control plane. Then the architecture in network device is no longer closed. In this way, SDN promotes the rapid innovation and the evolution of the network. Currently the idea of SDN is well received by both of the academic researchers and industry researchers. SDN becomes a hot topic in recent years and is considered as a promising way to re-architect the Internet. In addition, some research groups are formed like the Open Networking Summit [2]. Besides, the Open Networking Foundation [3] is defining the standard for SDN. And more than 100 companies have joined this foundation to accelerate the creation of standards, products, and applications, such as NEC, Google, Facebook, Microsoft, IBM, VMware, Juniper, Cisco and so on.

Software defined networking already has some implementations such as OpenFlow [4], NetOpen [5], in which OpenFlow as the most popular instance is deployed by many universities and research institutions in the world. To give readers a more concrete picture, we use OpenFlow to explain the whole ideas of this paper. SDN adopts the centralized control model. Currently a popular architecture is composed by the controller, FlowVisor [6], OpenFlow switch, and OpenFlow protocol as shown in the figure below (Fig. 1).

Software defined networking separates the control plane from the network equipment, and moves it to the SDN controller. Controller takes charge of all the functions in control plane, while the OpenFlow switch retains only the basic data forwarding function. FlowVisor can achieve the network virtualization, and different applications on the network controllers such as multicast [7], network situation assessment [8] and network status statistics could form their own private virtual networks. Furthermore, the SDN controller can just run in normal host or server to control the data packet forwarding in OpenFlow switches through a standardized OpenFlow protocol and a secure channel.

At present, SDN is considered as a promising way to re-architect the Internet. Transitioning the existing network to SDN becomes an important issue to the network researchers in current stage. However, so far the SDN itself is not mature in many aspects and the

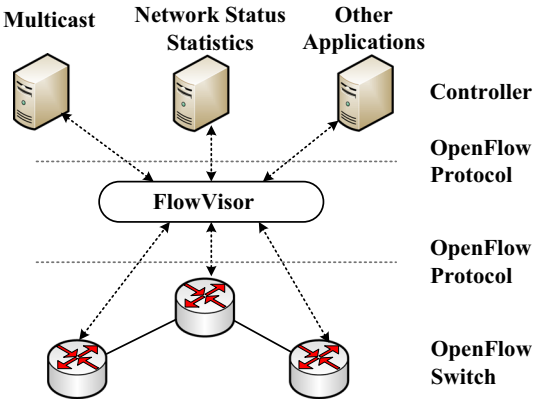

Fig. 1 Architecture of SDN

research on the transition issue just starts. In such situation, this paper takes the most widely deployed instance OpenFlow for example, and proposes a practical solution named BTSDN by integrating SDN networks to the current Internet with the Border Gateway Protocol (BGP) [9] and the BGP border routers. In BTSDN, the SDN networks and the current Internet coexist with each other, and the SDN networks can be incrementally deployed to the Internet, finally to replace the current Internet globally.

The main contributions of BTSDN are as follows: (1) It proposes that the controller runs Internal Border Gateway Protocol (IBGP) to learn the global routing information from border routers. (2) It proposes a feasible transition solution for the existing networks to SDN through the existing BGP protocols and BGP border routers.

The rest of the paper is organized as follows: the next Section presents the related work. Section 3 describes the design of the BTSDN architecture for transitioning the existing networks to SDN. Then this paper gives emulation and evaluation to the proposed solution in Sect. 4. Finally, Sect. 5 concludes this paper.

2 Related Work

At present, the technologies on transitioning the existing networks to SDN are few and not mature because SDN itself just started a few years ago. However, there are several preliminary progresses in connecting different OpenFlow networks together.

Tunneling [10]: different OpenFlow networks in different locations can be transparently connected together by the tunneling software. In the connected networks, OpenFlow can be further divide into virtual LANs (Local Area Networks). OpenFlow web site [11] provides the tunneling software for OpenFlow deployment. Also there some researchers focusing on building federated experiment testbed such as OFELIA [12], a European project providing OpenFlow based experimental facilities. Those works are either to make a LAN larger or to build a federated environment for experiment. They are not designed for the transition of the existing networks to SDN.

RouteFlow [13] is one of the implementations of IP routing on OpenFlow switches. The inter-domain is the standard BGP. RouteFlow instantiates a Virtual Machine (VM) for each OpenFlow switch with as many virtual network interfaces as there are active ports in the corresponding device, and runs a stack of open-source routing protocols on the virtual topology. All control messages are exchanged between VMs as if they are running a distributed control plane. Such a solution incurs the overhead of distribution without the benefits of scale.

WE-bridge [14] does not use BGP. It tries to take the advantage of SDN/OpenFlow network. It tried a multiple IP header fields matching routing solution among different administrative inter-domains. In WE-bridge testbed, there are no legacy routers in SDN networks. Also, the purpose of the routing solution is to achieve policy and fine-grained inter-domain routing for future SDN networks, not for the transition stage to pure SDN networks. Thus, WE-Bridge is not designed for on transitioning the existing networks to SDN.

To enable incremental deployment of SDN, we conducted a research work of SND-IP network peering [15] in 2013. This work focuses on the interaction between BGP based SDN domain and legacy IP network. This solution has successfully applied BGP between SDN and IP domains or between SDN and SDN domains. This solution has several deployments such as Internet2. In this solution, we totally discarded the legacy border BGP

routers in OpenFlow network. In the real world, border routers usually have a high performance and were bought with high prices. During the incremental deployment of OpenFlow, the network operators probably will not to throw the border routers directly. On the other side, we run software BGP in the SDN domain since we noticed that it needs years to enhance the BGP software to make sure it works well in large networks. Thus, in this paper, we propose another solution named BTSDN for transiting to SDN/OpenFlow network by retaining the border BGP routers, which can be deployed at the current stage.

3 Architecture of BTSDN

3.1 Overview of BTSDN

As mentioned above, BTSDN uses the BGP and retains the BGP routers in networks. The geographical distribution of BGP border routers, OpenFlow switches, and the controller is organized as in the Fig. 2.

In BTSDN, the functions of BGP including Internal Border Gateway Protocol (IBGP) and External Border Gateway Protocol (EBGP) [9] are the same as them in the Internet today. The inter-domain is still running the EBGP while the intra-domain is running the IBGP and OpenFlow protocol. BTSDN lets OpenFlow controller running the IBGP (we run a software BGP Quagga [16] as a network application above controller) to get the global routing information from the border EBGP routers which will be explained later. Further, each border router connects to one or several OpenFlow switches downstream.

In the BTSDN architecture, OpenFlow switch such as S(n) directly connected to a border router plays the role of the OpenFlow protocol proxy for the border routers. Since the traditional BGP routers do not support the OpenFlow protocol, the controller could not control the behavior of the border routers. But the controller can install certain instructions into the flowtable (specified in OpenFlow protocol) on OpenFlow proxy, and in this way indirectly control the border router. Therefore, in BTSDN, the controller still can control the behavior of the entire intra-domain network.

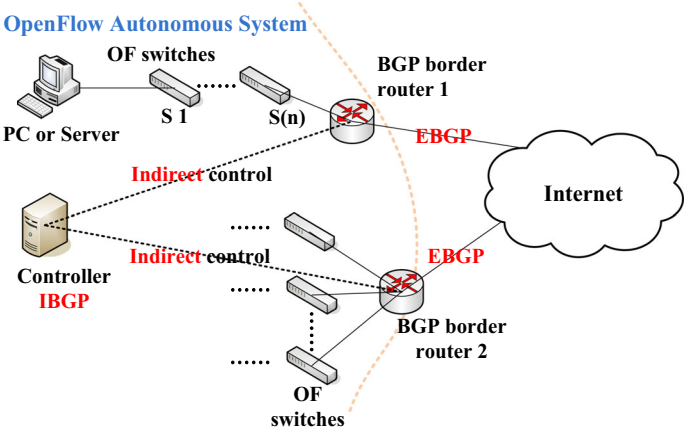

Fig. 2 Architecture of BTSDN

In addition, the OpenFlow switch forwards each packet according to the actions in the matched entry, in which, the matching fields of each flowtable entry are as follows:

Ingress Port	……	Ether src	Ether ds	……	IPv4 src	IPv4 dst	IPv4 proto/ ARP opcode	others

The IP address, MAC address, switch port are all included in the matching fields. Compared with a single OpenFlow flowtable, the current router needs two tables which mean the routing table and the ARP (Address Resolution Protocol) table to achieve the packet routing and forwarding. The routing table is for finding the next hop router, and the ARP table is for finding the MAC address of next hop router. Compared with the current packet routing and forwarding process, in OpenFlow, the controller masters the entire topology and computes the routing path. The only task of the OpenFlow switch is forwarding packet by the routing path installed by controller. In Fig. 2 after a packet entering this domain, even the destination MAC address is wrong, the OpenFlow switch (n) to switch 2 can still forward it to OpenFlow switch 1. But the switch 1 must know and rewrite the destination MAC address to the MAC address of the host; otherwise, the host will not accept the packet. When an intra-domain packet wants to be sent to other autonomous domain, the OpenFlow switch (n) should also know the MAC address of the directly connected border router port.

3.2 Inter-Domain Packet Delivery

OpenFlow moves the control plane out of the network forwarding equipment to the controller. The distributed intra-domain route computing becomes to a centralized route computing model. However, the OpenFlow network still retains the following characteristics: (1) OpenFlow does not change the format of the IP packet. (2) The centralized control model only changed the routing in intra-domain. Besides, the current Internet uses BGP for inter-domain routing. In the past decades, BGP has been widely deployed and now constitutes a critical part of the Internet infrastructure. Based on the characteristics of OpenFlow network, BGP can still be used for inter-domain routing in BTSDN.

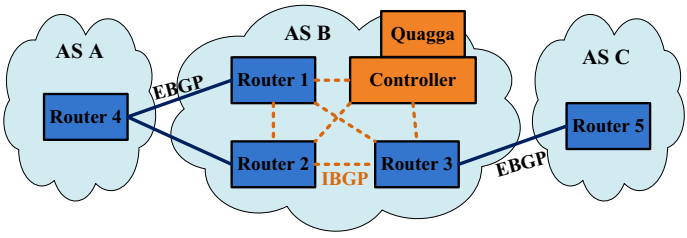

Fig. 3 Controller running IBGP

3.3 Intra-Domain Packet Delivery

In this section, we explain the intra-domain routing in two scenarios. (1) Scenario of a data flow exits the OpenFlow domain. (2) Scenario of a data flow enters the OpenFlow domain.

3.3.1 Controller Running IBGP

In BGP, border routers in inter-domain run External Border Gateway Protocol (EBGP) and in intra-domain run IBGP to set up a full mesh network for the routing information synchronization. By the IBGP synchronization mechanism, an IBGP router can learn the global routing information in real-time. Therefore, if we let controllers to run IBGP, they also can learn the global inter-domain routing information (Fig. 3).

As shown in the figure above, in BTSDN, the BGP still works as the same as in current Internet. The only difference is that the controller also runs the IBGP as an IBGP router to learn the global routing information.

BGP connections setup process for Controller and Routers: as mentioned before, we run software BGP software Quagga [16] as a network application above controller, and let Quagga to talk to all the border routers. Before Quagga can talk to those routers, it needs to know the IP address and MAC address of each router. Thus, we configure the Quagga as configuring a normal router, and tell Quagga all the border routers' IP addresses. Floodlight [17] can learn all the IP and MAC pairs of all the host/routers by its proxy ARP. So for the MAC address of those routers, we design and add an *ARP proxy module*, which can extract the ARP entries from Floodlight and insert those entries into the system ARP table. Then we pre-install forwarding paths between controller and border routers. In this way, the Quagga on controller can step up all the connections with all the border routers.

In OpenFlow, controllers can master and control the entire real-time intra-domain routing information such as the congestion and bandwidth. After running IBGP, the controller also masters the global routing information of the Internet. So the controller can send packets to correct egress routers and calculate out the best routing paths for all the data flows whether they enter or exit the OpenFlow domain. This advantage of mastering the global routing information is more obvious in the Multi-homing [18] scenarios, since packets with different source IP address should exit the OpenFlow domain through different egress routers.

3.3.2 Data Flow Exits OpenFlow Domain

In the current Internet, once a switch receives a packet from a downstream host, it will forward this packet to its default gateway and the packet will be routed according to a distributed interior gateway protocol such as OSPF [19], RIP [20], IS–IS [21]. But this does

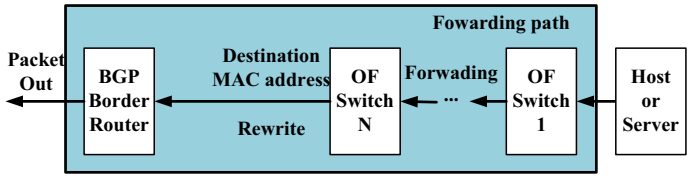

Fig. 4 Data flow exits OpenFlow domain

not work in OpenFlow network, because OpenFlow network works in a centralized manner. All the forwarding paths are calculated by the central controller in real-time. So once a new data flow is sent out from a host and reaches the directly connected OpenFlow switch, the switch will send the first packet to the controller if there is no matched flow entry in flowtable. Then controller calculates out a routing path (several flowtable entries) and installs it to the related OpenFlow switches. Then those OpenFlow switches forward this data flow according to the installed flowtable entries.

As in Fig. 4, until the data flow reaches the switch N which is directly connected to the BGP border router, the whole process in BTSDN is the same with that in current OpenFlow environment. But in BTSDN, after packets reaching switch N, the packet forwarding is different. As analyzed in the previous section, only when router receives a packet whose destination MAC is the same with the MAC address of corresponding router port, the router will accept the packet (except for special packets such as broadcast packets). Otherwise, router will discard the packet. With no change to the existing BGP protocol and BGP border routers, the forwarding from switch N to BGP border router can use the following method:

Destination MAC addresses rewriting: The ARP proxy module can learn the IP and MAC address pair of each router. Based on such information, we added a *destination MAC rewriting module* to controller. This module inserts the destination MAC rewriting flow entry to switch N. Before forwarding packet to the router, OpenFlow switch will first resets the destination MAC address in the packet to the MAC address of router according to the MAC rewriting flow entry, and then forwards packet to the BGP border router.

Once the border router receives a data package, the router firstly resolves the destination IP address in the header. Then it will find out the next hop router and the forwarding the packet to the corresponding router port according to the FIB (Forward Information Base) table. Then in link layer, the router will find out the MAC address of next hop router in the ARP binding table and rewrites the destination MAC address to the MAC address of next hop router, and at last sends the packet to the Internet.

3.3.3 Data Flow Enters OpenFlow Domain

In the current Internet, the border router itself computes the intra-domain routing path by certain distributed interior gateway protocol such as OSPF, and finds out the next hop router. Then by the MAC address learning process, it substitutes the destination MAC address in packet to the MAC address of next hop router, and sends the packet out to the appropriate router port.

However, OpenFlow is a centralized control model and all the routing paths are computed by the controller. In such situation, the border router should send the first packet of a new data flow to the controller. However, the legacy BGP router does not

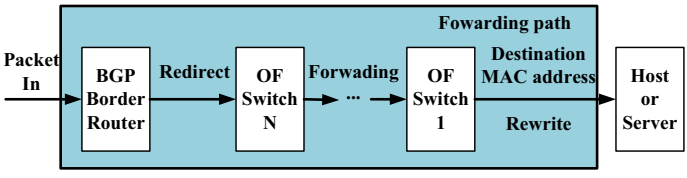

Fig. 5 Data flow enters OpenFlow domain

support the OpenFlow protocol and cannot send the first packet to the controller. Thus, in BTSDN, once a data flow enters the OpenFlow domain, the first issue is how does a border router forward the first data packet to the downstream OpenFlow switch? This can be achieved by the redirect function in routers. The border router should redirect all the packets received from the Internet to the directly connected OpenFlow switch. OpenFlow switch is responsible for sending the first packet to the controller. Then the controller carries out the routing path calculation and installation. At last all the routing path related OpenFlow switches forward the data flow to switch 1 as shown in Fig. 5.

Because the destination host only accepts packets whose destination MAC address is the same with the MAC address of itself (except for special packets such as broadcast packets). The method here is the same with the method when data flow exits an OpenFlow domain. The destination MAC rewriting module adds the destination rewrite flow entry to OpenFlow switch 1. OpenFlow switch will resets the destination MAC address in data packets to the MAC address of host/server according to the flow entry, and then sends the packet out to host/server.

In such scenario, the OpenFlow switch N plays the role of the OpenFlow protocol proxy of the border router. Thus the controller indirectly controls the border routers by directly controlling the OpenFlow proxy switches.

Table 1 Configuration for router connectivity manager

```
{
  "routers": [
    {
      "switchDpid": "100",
      "switchPort": "2",
      "routerIpAddress": "100.0.0.1",
      "tcpPort": "179"
    },
    {
      "switchDpid": "200",
      "switchPort": "3",
      "routerIpAddress": "100.0.0.2",
      "tcpPort": "2000"
    }
  ]
}
```

3.4 High Availability Design

In Fig. 3, we only draw one Quagga in it. For real network, single instance of Quagga software could not meet the requirement of high availability characteristic. Thus, we designed running several instances of Quagga software inside the SDN/OpenFlow network. All the Quagga instances and all the border routers are connected in full mesh.

3.4.1 Data Plane Connectivity

For the data plane connectivity, we designed and implemented a new module to controller, named router connectivity manager.

Firstly, we run Quagga software on a normal host or server and attach this host/server to an OpenFlow switch port of data plane. The network administrator knows the exact location of all the software Quagga routers and border hardware routers. The network administrator needs to configure two things: (1) Login all the software and hardware routers, configure each router as in the legacy network, such as tell each router the AS number, the peer IP address, the peer AS number, and so on. (2) Write down all the locations of all the software and hardware routers in a configuration file for the router connectivity manager. Each location should be represented by OpenFlow switch DPID (data path ID) and switch port (Table 1).

For each router entry, it includes the switch DPID, switch port, the IP address of the router, and the BGP communication port. The switch DPID and port is used for calculate connectivity paths. The router IP address and TCP port are used to match for flow entries. The router connectivity manager will automatically calculates all the paths according to the network topology and the configuration above, and pro-actively install all the paths for all the software and hardware routers.

Furthermore, for each router pair, there should be four flow paths and in each direction there should be two paths with the flow entries as follows:

Match fields in IP header			Action
dstIpAddress	srcIpAddress	dstTcpPort	rewriteDstMac
dstIpAddress	srcIpAddress	srcTcpPort	rewriteDstMac

In the table above, the MAC address of each router can be learned by proxy ARP.

3.4.2 Quagga Responsibilities

The Quagga software here has two responsibilities: (1) learn global route table. (2) The network administrator can also configure all the local and public routes into the Quagga. Quagga can announce those routes to its peers. In such way, hosts in other networks can also access the hosts with the IP addresses in this SDN network.

However, we should pay attention that all software Quagga hosts/servers do not route traffic. The purpose of running software Quagga is just to exchange routes with other BGP peers. The legacy border hardware BGP routers route traffic as they do in legacy non-SDN networks.

All the software Quagga routers are peering with each other. Each software Quagga has a global route table including local public routes.

3.4.3 Traffic Classifier

Before we calculate and install all the forwarding paths, we should classify the traffic by judging where traffic comes from and where this traffic wants to go. Then, we can conclude what is the path should be looks like, what we should match for this traffic and what actions should we carry out to this traffic.

Since all the routes are in Quagga, we can classify all the traffic according to the next hop of the destination IP address together with the switch port where the traffic comes from. The figure below shows the workflow about how to prepare to classify traffic (Fig. 6).

Table 2 below shows the traffic classification categories. Four cases will be concluded from the location of next hop IP address and incoming switch port. After traffic classfication, then we can calculate the correct flow path which is composed by flow entries.

4 Experiment and Results

4.1 Experiment Environment and Design

As shown in Fig. 7, ten PC nodes emulate three ASes (Autonomous Systems). AS 1 is an OpenFlow AS. AS 2 and AS 3 are legacy IP (non-OpenFlow) domains.

Hardware configuration: ten PCs and each with three network interface cards (Intel Core 2 Quad processor, 4 GB memory, network interface card 100 Mbps/port).

Software configuration: three PCs (PC1, PC2, and PC3) play the role of web users and were installed with the Microsoft Windows XP Professional operating system. All other

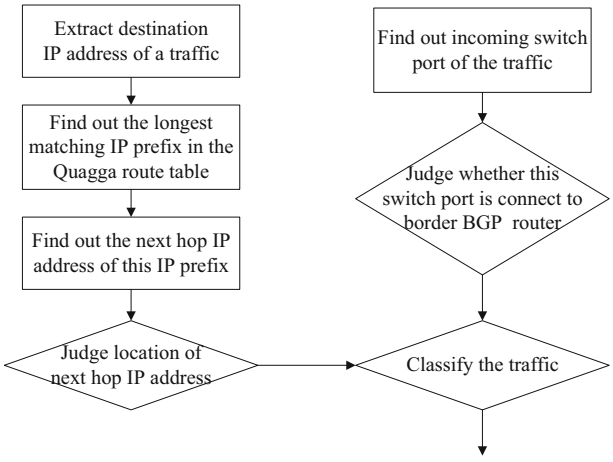

Fig. 6 Work flow of preparing how to classify a traffic

Table 2 Traffic classification categories

Case number	Traffic conditions	Traffic category
Case 1	Next hop IP address belongs to local BGP router, incoming switch port is connected to edge BGP router	This is a traffic from Internet to access local host/server
Case 2	Next hop IP address belongs to local BGP router, incoming switch port is not connected to edge BGP router	This is a traffic from local host/server to access a local host/server
Case 3	Next hop IP address belongs to peer BGP router, incoming switch port is connected to edge BGP router	This is a transit traffic from Internet to transverse local network and to Internet again
Case 4	Next hop IP address belongs to peer BGP router, incoming switch port is not connected to edge BGP router	This is a traffic from local host/server to access Internet

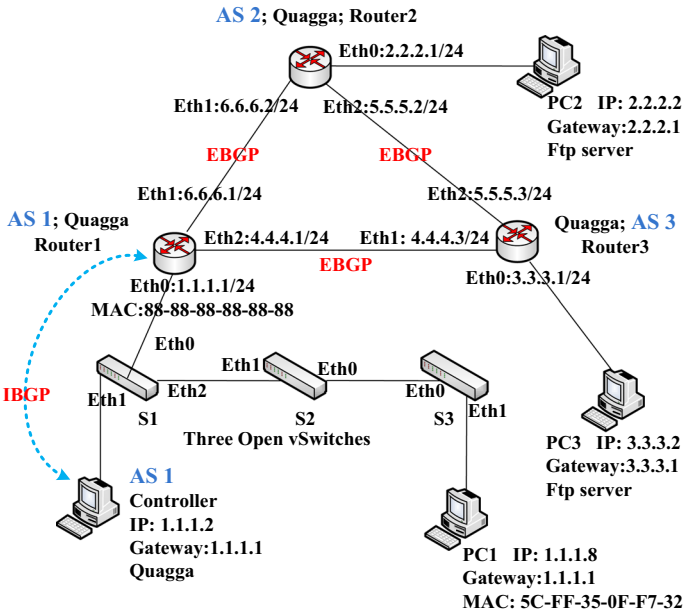

Fig. 7 The experimental topology

PCs including the controller are installed with Ubuntu-10.04-desktop-i386. The BGP software is quagga-0.99.20, and the switch software is Open vSwitch-1.4.0. Controller is Floodlight [17] and the FTP (File Transfer Protocol) server is WinFTP version 2.1.2.

4.2 Experimental Results and Analysis

We carry out the whole experiment in two steps.

Step 1 Verifying the feasibility of the controller running IBGP to get global routing information.

Firstly, we use the static flow pusher API (Application Programming Interface) in Floodlight to install the BGP path between controller/Quagga and router 1. Then we configure all the Ethernet ports with the number shown in the experiment topology. Then we start the BGP demon processes of Quagga software in the three border routers and the controller. Then we configure all the BGP routers with the AS numbers in Fig. 7, and also configure the controller with AS number 1. At last, it also needs to start the IP forwarding function in all the Quagga nodes.

Then router 2 announces IP address prefix 2.2.2.0/24 and 77.77.77.0/24, router 3 announces prefix 3.3.3.0/24, router 1 announces prefix 1.1.1.0/24. After about 6 s, we look up the route table in router 1, and find that router 1 has learned all the routes to the network 1.1.1.0/24, 77.77.77.0/24, 2.2.2.0/24 and 3.3.3.0/24. Then we find that the controller also has learned all the IP prefixes as in the table below (Table 3).

Thus, the approach of letting controller running IBGP to gain the global route items is feasible.

Step 2 Verifying the feasibility of BTSDN.

The target of this step is to verify that whether PC 1 can communicate with PCs in other non-OpenFlow domains in proactive and passive models respectively. We install the WinFTP software in PC 2 and PC 3. PC 1 will upload and download files from those ftp servers.

In this step, we firstly start the redirection function in router 1, and use the static flow pusher API of Floodlight controller to manually install the forwarding flow paths for PC1: (S1, S2, S3), (S3, S2, S1). We also add the destination MAC rewriting instructions into S3 and S1. The main instructions are show in the table (Table 4).

After such configuration, flow path installation, destination MAC address rewriting, finally PC1 can upload and download files from all the FTP servers. The target of step 2 is achieved.

Table 3 The routes learned by IBGP in controller

Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF, I
- ISIS, B - BGP, > - selected route, * - FIB route
K>* 0.0.0.0/0 via 1.1.1.1, eth0
B 1.1.1.0/24 [200/0] via 1.1.1.1 inactive, 00:04:50
C>* 1.1.1.0/24 is directly connected, eth0
B>* 2.2.2.0/24 [200/0] via 6.6.6.2 (recursive via 1.1.1.1), 00:04:50
B>* 3.3.3.0/24 [200/0] via 4.4.4.3 (recursive via 1.1.1.1), 00:04:50
B>* 77.77.77.0/24 [200/0] via 6.6.6.2 (recursive via 1.1.1.1), 00:01:05
C>* 127.0.0.0/8 is directly connected, lo
K>* 169.254.0.0/16 is directly connected, eth0

Table 4 Main instructions installed to Open vSwitch

	Fields	Action 1	Action 2
S1	Source IP: 1.1.1.8 import: eth2	Set the destination MAC address to 88-88-88-88-88-88	Output through eth0
S3	Destination IP: 1.1.1.8 import: eth0	Set the destination MAC address to 5C-FF-35-0F-F7-32	Output through eth1

4.3 Evaluation

This experiment preliminary proves the feasibility of BTSDN. Besides, BTSDN uses BGP as the Inter-domain routing protocol, so it inherits the advantages and disadvantages of BGP. The main advantages are: BGP is a mature protocol, and has been globally deployed. BGP has the ability to handle the inter-domain routing with no performance problem. The disadvantage of running BGP for the inter-domain is that we cannot route packets with multiple fields of IP header although OpenFlow supports this and it is still destination IP address based routing due to the nature of the BGP. Considering the main purpose of this solution is for transition stage of SDN, so not taking the full advantage of SDN/OpenFlow is acceptable.

BTSDN does not adopt any new protocol or new network equipment. It fully explores the characteristics of OpenFlow and uses the traditional BGP protocol and BGP border routers to help the existing networks transitioning to SDN. BTSDN is a simple and practical solution.

5 Conclusion

Currently SDN is well received by both of the academic researchers and industry researchers. SDN has been applied to campus and enterprise. However, there are few transition solutions for the existing networks to SDN. In this paper, we present a simple and practical legacy BGP and BGP border router based solution BTSDN for transiting the existing networks to SDN.

BTSDN fully explores the characteristics of OpenFlow network: OpenFlow does not change the format of IP packet, and the centralized control only takes effect in intra-domain. So BTSDN reuses the current BGP to connect the OpenFlow network and the rest of the Internet at the same time BTSDN still retains the traditional BGP border routers.

In order to achieve a cooperation of OpenFlow network and the legacy BGP border routers, BTSDN makes usage of the flow entries without changing the border routers. Besides, by running the IBGP on OpenFlow controller, the controller knows not only the entire intra-domain network status, but also the global routing information. This is very important for controller to calculate out a correct and reasonable routing path for cross-domain packets. For validation, we built a BTSDN environment and preliminary verified its feasibility.

Acknowledgments Supported by the National High-tech R&D Program ("863" Program) of China (No. 2013AA013505), the National Science Foundation of China (No. 61472213) and National Research Foundation of Korea (NRF-2014K1A1A2064649).

References

1. McKeown, N. (2009) Keynote talk: Software-defined networking. In *Proceedings of IEEE INFOCOM'09*, April 2009.
2. Open Networking Summit. http://opennetsummit.org/
3. Open Networking Foundation. https://www.opennetworking.org/
4. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, *38*(2), 69–74.
5. Kim, N., & Kim, J. (2011). Building netopen networking services over open flow-based programmable networks. In *Proceedings of ICOIN'11*, 2011.
6. Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., & Parulkar, G. (2009). *FlowVisor: A network virtualization layer*. Tech. Rep. OPENFLOW-TR-2009–01. OpenFlow Consortium, October 2009.
7. Karipidis, E., Sidiropoulos, N. D., & Luo, Z.-Q. (2008). Quality of service and max–min–fair transmit beamforming to multiple co-channel multicast groups. *IEEE Transactions on Signal Processing, 56*, 1268.
8. Qin, Y., Feng, D., Chen, K., Lian, Y. (2011). Research on monitor position in network situation assessment. *American Journal of Engineering and Technology Research*, *11*(9), 2197–2203.
9. Rekhter, Y., Li, T. & Hares, S. (2006). A border gateway protocol 4 (bgp-4). In *RFC4271*, January 2006.
10. Tunneling Software for OpenFlow Deployment, http://www.openflow.org/wk/index.php/Tunneling_Software_for_OpenFlow_Deployment
11. Tunneling—Capsulator. http://www.openflow.org/wk/index.php/Tunneling_-_Capsulator
12. OpenFlow in Europe. http://www.fp7-ofelia.eu/
13. RoutFlow. http://cpqd.github.io/RouteFlow/
14. Lin, P., Bi, J., Wolff, S., Wang, Y., Anmin, X., Chen, Z., et al. (2015). A west–east bridge based SDN inter-domain testbed. *IEEE Communications Magazine, 53*(2), 190–197.
15. Lin, P., Hart, J., Kobayashi, M., Krishnaswamy, U., Murakami, T., Wang, K.-C., Al-Shabibi, A., & Bi, J. (2013). Seamless interworking of SDN and IP. *ACM Computer Communication Review*, *43*(4): 475–476. Also in proceedings of ACM SIGCOMM13.
16. Quagga. http://www.nongnu.org/quagga/
17. Floodlight. http://www.projectfloodlight.org/floodlight/
18. Sousa, B., Pentikousis, K., & Curado, M. (2011). Multihoming management for future networks. *ACM/Springer Mobile Networks and Applications, 16*(4), 505–517.
19. Moy, J. (1998). Open shortest path first version 2. In RFC2328, April 1998.
20. Malkin, G. (1998). Routing information protocol version 2. In RFC 2453, November 1998.
21. Oran, D. (1990). Intermediate system to intermediate system. In RFC1142, February 1990.

Pingping Lin received her B.S. degree in computer science from the University of Electronic Science and Technology of China (UESTC) in 2008. She received her Ph.D. degree in computer science from Tsinghua University in 2014. The work in this article was carried out during her Ph.D. research. Her research interests include evolvable Internet architecture, source address validation, and software defined networking.

Jun Bi received B.S., M.S., and Ph.D. degrees in computer science from Tsinghua University. Currently he is a full professor and the director of the Network Architecture and IPv6 Research Division, Institute for Network Sciences and Cyberspace, Tsinghua University. He is a senior member of IEEE and ACM, co-chair of the AsiaFI (Asia Future Internet Forum) steering group, and co-chair of the CANS Future Internet/SDN working group. His research interests include network architecture and Future Internet.

Hongyu Hu received her Ph.D. degree in computer application major from Beijing Institute of Technology, Beijing, China, in 2009. She worked as a post-doctor at Tsinghua University from 2009 to 2012. She is now a teacher at Tsinghua University. Her current research interests include SDN, OpenFlow, and Internet architecture.