Enabling Shared Control and Trust in Hybrid SDN/Legacy Networks

Pinyi Shi, Sergio Rivera, Lowell Pike, Zongming Fei, James Griffioen, Kenneth Calvert
Laboratory for Advanced Networking
University of Kentucky
Lexington, Kentucky 40506–0495, USA
Emails: {pinyishi,sergio,pike,fei,griff,calvert}@netlab.uky.edu

***Abstract*—A key concept of software-defined networking (SDN) is separation of the control and data plane. This idea provides several benefits, including fine-grained network control and monitoring, and the ability to deploy new services in a limited scope. Unfortunately, it is often cost-prohibitive for enterprises (and universities in particular) to upgrade their existing networks to wholly SDN-capable networks all at once. A compromise solution is to deploy SDN capabilities incrementally in the network. The challenge then is to take full advantage of SDN-based services throughout the network, in an integrated fashion rather than in a few "islands" of SDN support. At the University of Kentucky, SDN has been integrated into the campus network for several years. In this paper, we describe two aspects of this challenge, along with our solution approaches. One is the general reluctance of campus network administrations to allow novel or experimental (SDN-based) services in the production network. The other is how to extend such services throughout the legacy part of the network. For the former, we lay out a set of principles designed to ensure that the production service is not harmed. For the latter, we use policy based routing and a graph database to extend our previously-described VIP Lanes service. Our simulation results in a campus-like topology testbed show that we can provide a host with custom path service even if it is connected to a legacy router.**

***Keywords*-software defined networks, legacy routers, hybrid network, campus network**

I. Introduction

Software-defined networking (SDN) separates the logical, software-based control of network switches from the hardware used to implement packet forwarding. This enables rapid evolution of network control software, and speeds up development and deployment of new network services. This separation typically translates into a network design that consolidates the control plane software into a *network controller* that is responsible for making decisions about how packets will be handled according to policy, and then pushing those decisions—in the form of match-action rules—to the switches that make up the network. SDN-enabled switches simply use those rules to forward, drop, modify, or otherwise process packets *in the data plane*. Given an accurate "bird's eye" view of the network, the controller can ensure that network policies are enforced consistently throughout the network. This simplifies network management and monitoring for network administrators [1].

SDN controllers typically uses a protocol like OpenFlow [2] or Open vSwitch Database (OVSDB) [3] to communicate control information to/from SDN-enabled switches. In many cases, SDN controllers also support a Northbound Interface (NBI) that allows external applications to communicate with, and influence the behavior of, the controller. Fig. 1 illustrates the general architecture of an SDN network and the software—both SDN controller and external applications—that drives the network.

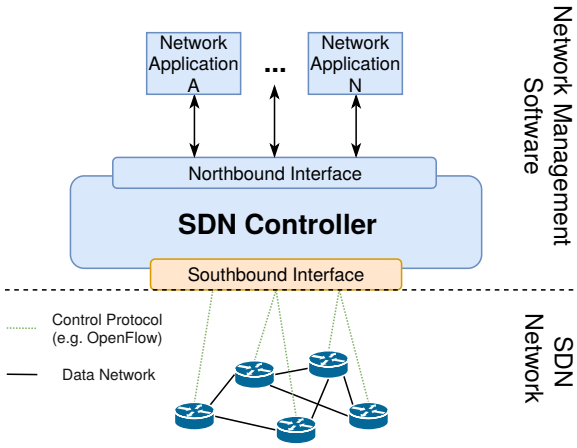


Fig. 1. SDN controller and its Northbound and Southbound Interfaces

The benefits of SDN are most obvious within a single "stub" administrative domain, especially one that has substantial intra-domain ("east-west") traffic, as well as ("north-south") traffic to/from the rest of the Internet. Thus, data centers were the early adopters of SDN, while campus and enterprise networks have been somewhat slower to get on the bandwagon.

Because of the heavy emphasis on distributed algorithms in Internet control protocols (in particular routing), such networks have traditionally relied on various layer 2 mechanisms and vendor-supplied services to aid management. In a pure SDN network (i.e., one in which all network elements are SDN-capable), all control plane functionality would be implemented in the controller(s) software. However, several factors prevent wholesale conversion of enterprise networks to pure SDN. One is simply cost. Another is the challenge of reimplementing all control plane functionality, developed over decades, for a new platform. (Moreover, it is not clear that all aspects of control *should* be centralized.) As a result, SDN approaches typically

either (1) deploy SDN switches (with limited control plane functionality) that interoperate with legacy network switches, or (2) implement integrated SDN software that co-exists with the conventional network control software embedded in SDN-enhanced legacy switches.

In the first approach, the network is partitioned into SDN regions and legacy regions—forming a *hybrid network*. Switches in the SDN regions are controlled by SDN software, while switches in the legacy regions execute standard distributed control protocols (e.g., OSPF, LLDP, etc.) to control the network. The SDN regions interconnect to the legacy regions at clearly defined points and are configured to interoperate in some way. Because the two regions are independent networks, they can be controlled independently (i.e., no need for shared control between SDN developers and IT network operators). Cooperation is only needed at the interconnection points, and can be achieved by defining a small, static, set of ways to exchange packets between SDN and legacy regions. For example, one proposal has strategically-placed SDN switches act like a legacy switch and send legacy control (e.g., ARP) messages to neighboring legacy switches [4]. The obvious downside of the hybrid approach is that advanced SDN services are limited to the SDN regions of the network.

In the second approach, legacy network switches are enhanced with SDN capabilities (or simulations thereof), allowing both forms of control to co-exist on the same switch. This approach, which we call *integrated*, was largely the motivation for the original OpenFlow design [2] which proposed that SDN capabilities be added into legacy switches. In this case, SDN-enhanced switches are able to support the plethora of legacy control protocols while also allowing SDN controllers to modify the forwarding behavior of switches. This approach (i) leverages past investments in control functionality; (ii) enables experimentation with, and development of, new control functionality; and (iii) potentially allows SDN developers to deploy services across the entire enterprise. It would therefore seem to be the preferred approach. However, it seems to be rarely used in practice.

There are several reasons (besides the cost of replacing/upgrading the entire network) why enterprise and campus information technology (IT) administrators might be reluctant to adopt the integrated approach in production networks. First and foremost, their job is to keep the production network running smoothly. They therefore—quite understandably—tend to be wary of new technology until it has proven itself reliable, perhaps in a confined portion of the network (hence the prevalence of the hybrid approach). Also, IT groups find it challenging enough to keep up with legacy control protocols, let alone develop new skills around SDN control software. And while the integrated approach can support experimentation with new network services [2], as noted above, IT groups generally have no interest whatsoever in allowing experiments on their production network. It should be noted that these reasons for not embracing an integrated approach (or even the hybrid approach) are not technical in nature. Instead they are based on lack of funding, lack of knowledge, lack of trust, and fear of disruption of production network services.

At the University of Kentucky (UK), we have been operating an *integrated* SDN network on the campus for several years [5], [6]. The integrated approach has allowed us to deploy new services across large portions of the campus that would not have been possible if we had taken a hybrid approach. Key to our ability to support an integrated model has been the ability of researchers and SDN developers to work cooperatively with our IT department. Achieving this level of cooperative control and trust was the result of establishing procedures and guidelines that shaped the way we built and deployed SDN capabilities on campus. Moreover, although our SDN deployment did require substantial funding to replace legacy network equipment across much of the campus—which would have been prohibitive without opportunities such as the National Science Foundation's Campus Cyberinfrastructure (CC*) program [7]—we have recently developed the ability to support certain key SDN capabilities using non-SDN-enabled routers, thereby allowing us to extend the footprint of our SDN services to the entire campus.

The agreements and procedures between our research group and our campus IT network groups have been developed and refined over several years. Our initial deployment had aspects of a hybrid design, but over time has evolved to where we can now envision providing advanced services at any point where they are needed, even in parts of the network served only by legacy routers. In addition to developing cooperative procedures, we had to develop the software infrastructure needed to support the procedures and enable shared control over the network—which had to be based on trust that the control could not be misused. In particular, we designed and developed a unified controller that is capable of controlling flows that traverse both SDN-enabled and non-SDN-enabled switches.

The basic idea is to use policy-based routing supported by legacy routers to make them behave like SDN-enabled routers, allowing the SDN controller to control paths that flows take across the campus network regardless of the SDN capabilities of the campus switches. Moreover, in the case where some nodes are SDN-enabled and some are not, we leverage policy-based routing to forward packets from non-SDN-enabled switches toward SDN-enabled switches where more advanced SDN features (e.g., header modification) can occur. As a result, our design supports an integrated control framework that can deal with hybrid networks.

The rest of the paper is organized as follows. Section II begins by discussing the challenges and the foundational agreements and procedures between our IT groups and research groups needed to support hybrid and integrated approaches. Section III then describes our early deployments that utilize a combination of integrated and hybrid approaches to support high-speed approved flows for big data transmission. In Section IV, we describe the mechanisms that we developed to support a fully integrated approach, including the use of graph databases as well as the use of Policy Based Routing (PBR). In Section V, we describe experimental results show-

ing the performance we are able to get over our integrated network, including non-SDN-enabled switches. In Section VI, we describe related work and other transitions from legacy networks to SDN networks. We discuss drawbacks of our solution and possible future work and wrap up with our conclusions in Section VII.

II. Shared Control and Trust

While SDN networks have become common place in (commercial) data centers, they have not been widely adopted by enterprise or campus networks. As noted earlier, the reasons for this are often non-technical, and instead arise from a lack of policies and procedures for sharing control and establishing trust between IT network administrators and SDN developers. In fact, offering the "ability for (application) software to control the network" can be a scary thought for many IT network administrators, who are tasked with the job of ensuring the network is rock-solid and are all-too-aware of "horror stories" in which small configuration errors have taken down entire networks. While SDN provides many opportunities for developing and deploying innovative applications, the controllers and the applications they support (via their Northbound Interface) essentially have complete control of the data plane—so that even the smallest error could wreak havoc on the network. The challenge is thus to persuade IT decision-makers to allow SDN developers to create and deploy new software-controlled services in the campus *production* network—not just small segregated experimental networks.

It should be noted that while network vendors often support one or more SDN controllers [8]–[11], these controllers typically have only the most basic, limited, capabilities and thus offer relatively little added value in and of themselves. The real value is created by writing SDN controller modules or applications that talk to controllers. So while IT network administrators may trust a vendor, there is little advantage to an SDN network based on vendor-supported controllers alone. To obtain the full benefits, IT/network administrators have to trust and allow software from SDN developers (their own or third parties) to control "their network".

The costs of upgrading to SDN-equipment have become less of an issue over time. Many of the production-grade switches being sold today support some form of SDN, which means that campus networks are becoming more SDN-capable as part of their normal upgrade path. However, large portions of most campuses networks have not been upgraded to SDN-enabled equipment, and even when they are, IT network administrators do not turn on SDN for the reasons mentioned earlier.

A. Principles of Cooperation

To address these issues, we have been developing a cooperative model at the University of Kentucky that has enabled our SDN developers (researchers) and IT Network Administration to establish mutual trust based on procedures and policies for shared control of the campus network infrastructure. A key part of this has been the development of software that enforces the polices, assuring IT staff that SDN developers can "do no harm" (or will only harm themselves).

A key step toward assuring network administrators that SDN developers will do no harm, is to establish the following foundational principle of shared network control:

> *By default, the network will continue to be controlled by IT using their existing procedures, and packets will continue to be processed according to IT policies as always. Only IT-approved traffic will be eligible to be controlled by SDN developers (and their software).*

In other words, the default setting is that packets and flows will be handled by network switches as they always have been. SDN processing can only be applied to certain types of flows known to (and approved by) IT as safe and non-threatening to their normal operations.

To implement this principle, we developed controller software that ensures a default SDN rule is inserted into all campus network switches that says, "if the packet does not match a flow approved by IT for SDN-processing, it must be processed using the switch's normal data path"—as if SDN capabilities are not present. While this may appear obvious and straightforward, it is not as simple to implement as it may seem. For example, although the OpenFlow protocol supports a so-called "normal rule" action that can be applied as the default action, it is an optional part of the Openflow protocol and is not supported by all switches. Even if a switch supports OpenFlow, it may not support the "normal rule" that we require. In other cases, the switch may support the "normal rule", but may take the packet off the switch's fast-path to carry out the normal rule action, thus incurring a significant performance penalty on "normal" traffic. As a result, it is important to identify which switches in the network are capable of supporting the "normal rule" at line speed, and which are not, and thus require other mechanisms to support the "do no harm" objective.

Because the normal rule is used to catch and process all network traffic by default, we maintain the following invariant:

> *All SDN processing rules are inserted at a higher priority so that SDN traffic is effectively "picked off" before it hits the normal processing path.*

As a result, IT network managers can continue to manage switches as they always have, knowing that any SDN packets have been "picked off" and processed independent of the switch's normal data path. In other words, IT can continue to manage the network without coordination with the SDN developers (other than approving which flows are allowed to be "picked off" for SDN processing).

To provide further assurance that SDN will not affect normal traffic, our control software ensures the following:

> *SDN processing is only applied at the granularity of a flow, specified as a five tuple (IP source and destination, type UDP/TCP, and Port source and destination). The flow must be approved by IT or its delegate.*

In other words, the only traffic that will be processed by SDN software must match a flow spec approved by IT. The details of our control software used to authorize SDN flows is outside the scope of this paper, but is part of our VIP Lanes project [5]. In short, VIP Lanes allows IT to delegate administration of portions of the flow space to other IT staff, faculty, or students. For example, IT may delegate control of all ssh flows (TCP port 22) to or from machines on the Computer Science Department network (defined by an IP address range) to the computer science network administrator to manage. The ability to give out specific portions of the flow space, down to the flow (i.e., port) level, allows campus IT to be assured that SDN processing is limited to approved traffic, and cannot be used to intercept other campus traffic.

Our control software further limits what type of processing can be applied to flows. Although OpenFlow allows packets to be forwarded out of certain ports, modified/rewritten, dropped, etc., we maintain a conservative stance:

> *The set of actions that can be applied based on the flow specification is limited.*

For example, VIP Lanes flows are only allowed to redirect a flow out of a certain port, possibly rewriting the VLAN field and destination MAC address. In another SDN service we run based on fail2ban [12], it only uses the drop operation to dynamically block ssh attacks on port 22.

B. Enabling SDN Control of Non-SDN Switches

To address the hybrid network problem where our controller software can only manage the SDN regions of the network, we extended the capabilities of our SDN controller to view legacy switches as SDN switches with limited capabilities. As a result, our SDN controller is now capable of controlling both SDN-enabled switches and non-SDN legacy switches—making it possible for SDN developers to create new services that reach the entire campus network. We have used this feature to extend our VIP Lanes service [5] to support hosts connected to areas of the campus network that do not have SDN-enabled switches.

Our initial SDN controller had a bird's-eye view of, and control over, the SDN network topology, but lacked visibility/control over the legacy regions of the network. While legacy routers may not support SDN protocols like OpenFlow, they typically support the ability to redirect traffic or drop packets, which are the key capabilities needed by SDN applications. In particular, many legacy switches support policy routing, which allows network administrators to configure routes that take precedence over routes computed by the control software (i.e., the intra-domain routing protocol such as OSPF or Cisco EIGRP). We can leverage these capabilities to allow our SDN control software to redirect traffic and define alternate paths across the network. In addition, most switches support access control lists (ACL) that define which packets are allowed to pass through the switch. Strategic use of these capabilities can be used to block or drop specific flows on behalf of the SDN controller. Moreover, it is possible for a centralized controller to communicate with legacy switches either through SNMP or via remote login (ssh) to the switch's command line interface (CLI). Consequently, we have enhanced our controller to make use of these legacy features to allow SDN applications to program legacy switches much as they do on SDN-enabled switches, thereby opening up new SDN capabilities to hosts located on legacy regions of the network.

III. A Motivating SDN Service

A key motivation for converting our campus network to an SDN-enabled network was a desire to support high-speed flows both to/from the Internet (north/south) and across campus (east/west). We observed that the primary bottleneck for high-speed flows was often the middleboxes that are littered around campus, and, in particular, guard the Internet/campus boundary in the form of firewalls, intrusion detection/prevention systems, and NAT boxes. Our goal was to create an SDN service that leveraged SDN to route high-speed (pre-authorized flows) around performance-limiting campus middleboxes.

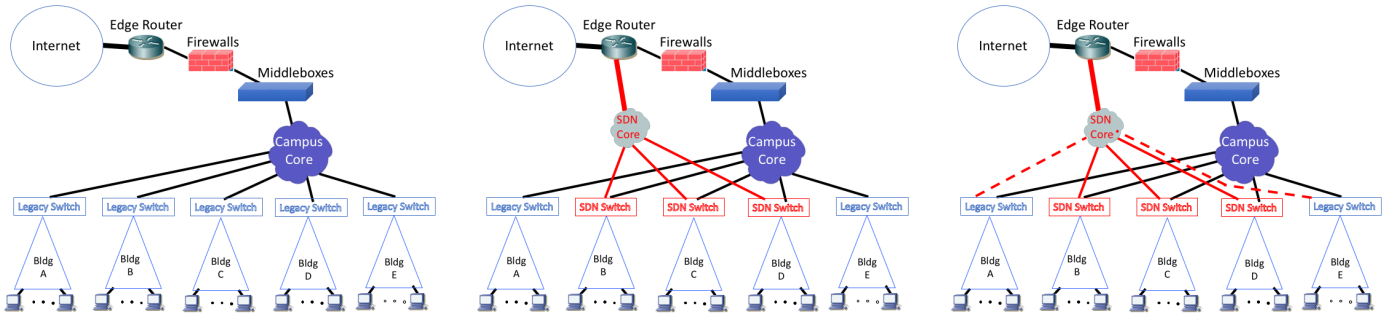
Like most campuses, we began with a conventional campus network architecture consisting of a campus core connected to distribution nodes, which in turn connect to a hierarchy of access switches as shown Fig. 2a. Unlike many other SDN deployments, we decided against a separate SDN network just for machines that needed high speed flows. Instead, we decided to replace the production network in entire buildings with SDN, allowing any machine in the building to create high speed flows if desired. However, we were not able to replace the equipment in all buildings, so we were still left with a hybrid SDN network (see Fig. 2b).

More recently we have enhanced our control software and VIP Lanes services to be able to control legacy switches as well as SDN-enabled switches. As a result, we have the potential to offer high-speed flows from anywhere on campus using the integrated network shown in Fig. 2c.

In the following, we briefly describe our VIP Lanes service and the way it uses SDN to route around middlebox bottlenecks. We then show how we extended it from a hybrid network to an integrated network.

A. Supporting High-speed Flows

In recent years the need to transfer big data at high speed has been increasing. A key challenge to fast big data transfers are the middleboxes (e.g., firewalls, NAT, and IDS systems) that have become essential components of enterprise and campus networks. These appliances provide valuable network services; however, they also present "obstacles" that degrade the performance of big data transfers. To address these middlebox issues, our previous work developed and deployed a VIP Lanes [5] SDN network on the campus of the University of Kentucky. Trusted users can use the system to create pre-approved flows that bypass middleboxes and traverse a path directly to the campus edge router. Users are placed in groups and are delegated the responsibility to manage portions of the flow space, deciding which flows should be allowed to bypass



(a) Traditional Network (b) Hybrid SDN/Legacy (c) Integrated SDN Network

Fig. 2. Evolution of Campus Network

policy-enforcing middleboxes and be allowed to perform high speed big data transfers. Unlike a science DMZ approach, VIP Lanes assigns trust to users/groups as opposed to machines, and can assign trust on a per-flow basis. The structure of the VIP Lanes system is illustrated in Fig. 3.

To use the VIP Lanes system, an authorized user can request the creation of a VIP Lane from the local machine (identified by a local source IP address and port number) to a destination machine (identified by a destination IP address and port number), using a particular protocol (TCP or UDP). The user's credentials are first checked to verify they have been authorized to create VIP Lanes for that portion of the flow space. Once verified, the SDN controller – more specifically, the VIP Lane Path Service [6] – can uses its knowledge of the topology to compute a VIP Lane path that bypasses middleboxes.

Recently we have enhanced our VIP Lanes service to provide high-speed VIP Lane paths to the users in buildings that are not SDN-enabled. As a first step, we physically connected the legacy routers in those buildings (A and E) with the SDN core shown in Fig. 2c. To help the SDN controller obtain a view of the entire network topology – including the legacy switches and connected hosts – we used a static JSON-encoded configuration file in combination with a Neo4j graph database. In our network graph, network devices (e.g., switches and routers) and hosts are vertices while the links among them are the edges. The path service library first loads the topology from the controller and creates an initial topology graph in Neo4j. Then it takes advantage of the Simple Network Management Protocol (SNMP) along with Cisco Discovery Protocol (CDP) to search the information of the legacy Layer 3 (L3) switches based on the IP addresses provided in the static files. It also queries the ARP table of the legacy switches to find the information of potential hosts. After all the required information of the legacy network is collected, a new graph is created based on the collected information and is added to Neo4j as a complement to the initial graph. The path service library on the SDN controller then has a bird's-eye view of the entire hybrid network, including all the potential hosts that may use the VIP Lanes system. With a view of the entire topology, the path service library is able to compute a hybrid SDN path for the hosts that are connected to a legacy switch. To redirect traffic from nodes on the legacy network into the SDN network, we developed a new controller module designed to communicate with legacy switches using Policy Based Routing (PBR) – i.e., the PBR Module – to intercept VIP Lanes traffic and bypass performance limiting middleboxes.

B. System Architecture

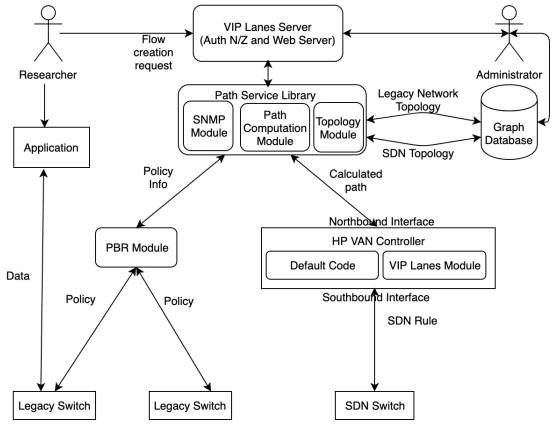


Fig. 3. Architecture of the VIP Lanes system

The system architecture of VIP Lanes is shown in Fig. 3. The major components of the VIP Lanes system include the VIP Lanes server, the path service library, the graph database, the Policy Based Routing (PBR) module, and the VIP Lanes modules on the SDN controller. The basic workflow looks like this: (1) The big data researcher uses the VIP Lanes server for a flow request. After the authentication for credentials, he/she inputs (source IP address, destination IP address, protocol, destination port, timeout) on the web GUI. (2) The VIP Lanes server authenticates whether the source IP address can be used by the user by checking the permissions of the groups the user is in. (3) If the authentication process is successful, the VIP Lanes server sends the request to the path service library. The path service library communicates with the graph database to fetch the information of the entire topology. Then

it puts together the request from the VIP Lanes server and the topology information to calculate a path that bypasses the middleboxes. This path may include those legacy switches that connecting the researcher to the campus network. (4) The path service library calls the module on the controller using the REST API to insert SDN rules on all the SDN switches on the path, and the PBR module to install policy-based routing policies on legacy routers. For each calculated path, two VIP Lanes are created for both the forward traffic and reverse traffic. The user can also specify the timeout of the SDN rules based on how much data he/she is going to transfer. The policies installed on legacy routers have to be removed explicitly by the PBR module when they are no longer needed.

IV. Integrating Topology Discovery and Route Control for Legacy Networks

In this section, we present our efforts to complement the SDN topology information discovered by the controller with network devices and end-systems that are typically found in portions of the campus legacy network, and describe how we configured Policy-Based Routing (PBR) on legacy L3 devices to redirect the traffic to the SDN network.

A. Topology Discovery

Unlike an SDN-only network, where the SDN controller has both a *complete* view of the network and well-defined standardized southbound protocols to reconfigure network devices, a hybrid network is more complicated to manage. Part of this difficulty is the fact that the non-SDN portion of the network is unknown to the SDN controller; legacy devices are designed to behave independently based on their own (local) control plane and information received from neighbors via distributed (possibly proprietary) protocols. Fortunately, over the years, these protocols that are used to discover neighbors or calculate routes, and ultimately modify state data in legacy devices can be polled from monitoring entities using the Simple Network Management Protocol (SNMP). In fact, in our initial prototype of VIP Lanes, we leveraged SNMP to obtain information regarding the VLANs supported in the links connecting any two SDN-enabled devices and whether a switch was layer 2 or layer 3 device. The information was useful to determine the set of actions (i.e. packet header modification) that needed to be included per rule. To deal with the legacy network, we took a step further and extended this functionality by querying additional Management Information Bases (MIBs) to discover not only neighboring L3 legacy devices attached to SDN-enabled switches but also the hosts connected to those legacy devices. Similarly to what we did previously, we added all the information to the topology database we use to calculate paths, adding specific properties that allow our queries and path computation library to be aware of the legacy portion of a potential path. While this approach should work with any vendor (SNMP is widely available in legacy switches), our implementation used Cisco L3 switches. L3 is the right place to redirect traffic using policy routing because flows in VIP Lanes are defined based on IP addresses and other facotrs, and in our setup, each Cisco L3 device was connected to both an SDN-enabled switch and the normal campus core network.

Moreover, due to the campus policy, the Cisco Legacy switches run Cisco Discovery Protocol (CDP) instead of the Link Layer Discovery Protocol (LLDP). Given a set of IP addresses of the potential Cisco legacy switches that need to be integrated to SDN network, we performed SNMPWALKs on each SDN switch using the CDP MIB. After parsing the results, we check whether any IP address returned by SNMP is in the predefined set of legacy switches. For connections between a legacy switch and an SDN switch, we used the existing SNMP-based mechanism to get information such as the VLAN ID, port numbers and their associated MAC addresses. This step is unchanged because the fetched information is essential to determine the packet header rewrite operations that SDN switches must perform.

In terms of end-system discovery, we did not query information beyond what was learned by the L3 device. For instance, information about how the host is connected to the legacy L3 switch (e.g., how access switch(es) is (are) placed between them). To get the information about the hosts, we make use of the Address Resolution Protocol (ARP) table on the legacy switch. The ARP table provides a mapping between the IP address and the MAC address. We use SNMP along with the ARP MIB to get the required information (IP, MAC address, Name, Port, VLAN ID) and store it for each host. For the link between the host and legacy switch, we store information such as the MAC addresses of the interfaces on both ends of the link as well as the VLAN where the host was discovered. After getting all the essential information, we transformed that information into nodes and links in the system's topology graph database in the same way we presented in [6] and assigned a special label to the learned legacy devices.

B. Policy-Based Routing

In legacy network, the default routing mechanism is destination address based. To provide fine-grained control of routing paths, we can use Policy Based Routing (PBR), which can include source address and some other information for making a decision. Since the syntax of PBR is vendor dependent, here we only discuss the PBR for Cisco switches.

According to [13], Policy Based Routing allows us to define a policy for a unicast flow. Access Control List (ACL) and route-map are two important components of PBR. There are two types of ACLs, standard and extended. A standard ACL only matches on the source address while the extended ACL can also match on the destination address as well as the ports. In a PBR, the route-map first classifies the traffic based on the access list and then defines the action such as setting the next hop address. When the PBR is configured, it has to be applied to the ingress interface where the traffic is coming from. Then the matched class of traffic will be routed based on the action in the PBR.

Since the users of the VIP Lanes system specify both the source IP address and the destination IP address when they request to create flows on the server, the SDN rules inserted

on SDN-capable switches also use both source and destination IP addresses in the match statement. We then decide to use the extended ACL in the PBR so that the granularity of the control is the same as what is included in the SDN rules created on SDN-capable switches. Fig. 4 shows a sample configuration of PBR and the application on an interface. The access list permits the TCP traffic from source IP address 172.23.7.194 to destination IP address 172.23.7.178. The route-map defines the action for this class of traffic: sent to the next-hop address of 10.1.5.1. Finally, this route-map is applied to interface VLAN16 as an IP routing policy. So when the traffic arrives on interface VLAN16 and match the access list, it will be routed to the next-hop with address 10.1.5.1.

```
xavier#sh access-list
Extended IP access list 101
    10 permit tcp host 172.23.7.194 host 172.23.7.178
xavier#
xavier#sh route-map
route-map pbr, permit, sequence 10
  Match clauses:
    ip address (access-lists): 101
  Set clauses:
    ip next-hop 10.1.5.1
  Policy routing matches: 0 packets, 0 bytes
xavier#
xavier#sh ip policy
Interface      Route map
Vlan16         pbr
```

Fig. 4. An Example Configuration for Policy Based Routing

V. Experiment Setup and Test Results

In this section, we present the throughput measurements for two types of experiments. In the first one, we analyzed performance between two machines on campus (i.e., east-west flows) using the `iperf` tool. Then, in the second experiment, we were interested in analyzing the behavior of flows from machines deep in our campus network to various Internet2 sites using the `perfSONAR` tool. The results from both experiments show that hosts in the legacy portion of the network can enjoy the service provided by the VIP Lanes system.

A. East-West Flow Experiment

For this experiment we set up a laboratory testbed as shown in Fig. 5. The SDN portion of the testbed comprised Aruba 3800 series switches with OpenFlow enabled and running in hybrid mode. In the legacy network portion, the Cisco L3 switch was a Cisco 3750 running Cisco IOS version 12.2(55)SE7.

In this experiment, we first measured the throughput between two machines, la2-pc1 and la3-pc1, that were connected directly to the SDN network. We used `iperf` to send TCP traffic between these two machines and compared the results between the default route (going through the *Core* switch) and SDN route by turning VIP Lanes on/off. The result was plotted in Fig. 6 and shows that the performance over the default route between these two hosts under default conditions was severely limited by the bottleneck links (dashed lines in

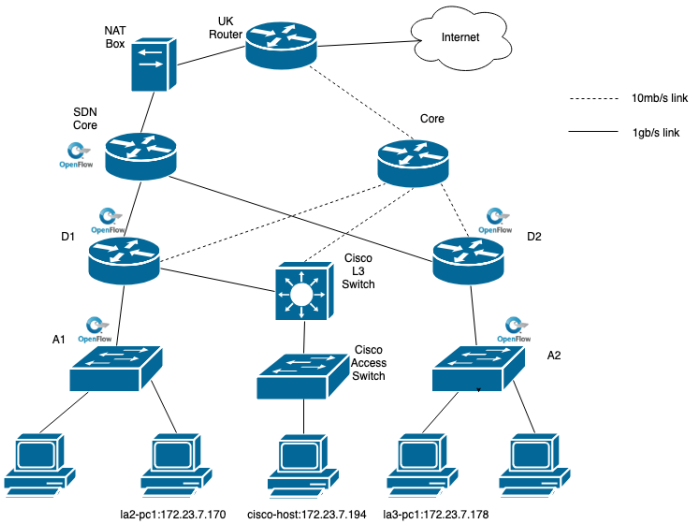


Fig. 5. Campus-like hybrid topology prototype

Fig. 5) connected to the *Core* switch (data fluctuating around 10 Mbps). However, when we enabled VIP Lanes and let the traffic go through the SDN core we got throughput around 510 Mbps.

We performed a similar experiment to measure the throughput between the Cisco-host and la3-pc1. In this case, the key difference was that one of the end points (the *cisco-host*) was attached to the legacy portion of the testbed. As the result in Fig. 7 shows, the default throughput stayed around 10 Mbps whereas the SDN throughput got closer to the result we obtained in our previous experiment, showing that VIP Lanes involving PBR entries to reroute flows traversing the Cisco L3 switch to the SDN network did not impact negatively on the performance.

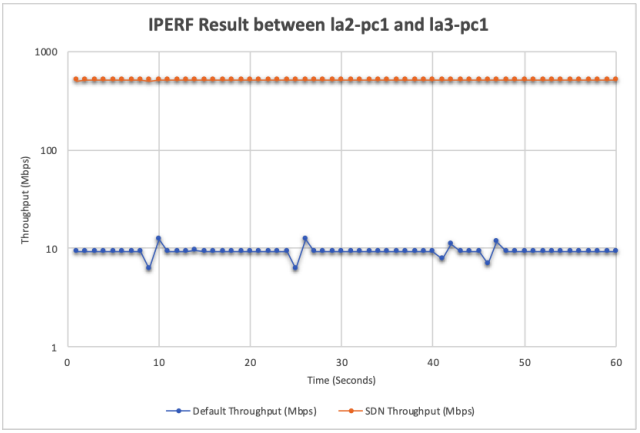


Fig. 6. Iperf result between la2-pc1 and la3-pc1 (log scale)

B. North-south flow experiment

The VIP Lanes system has been deployed on the campus production network for the researchers to move the big data to/from the cloud storage. In this experiment, we tested on the real campus production network to see whether trust and control can be realized on the legacy portion of the network.

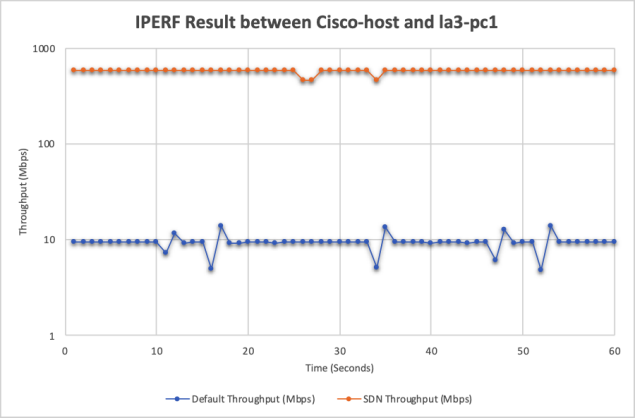


Fig. 7. Iperf result between Cisco-host and la3-pc1 (log scale)

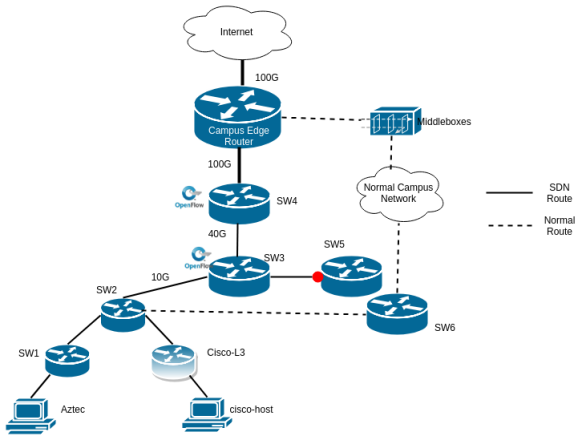


Fig. 8. Sample part of campus network topology

Fig. 8 shows the difference between how SDN traffic and normal traffic go out of campus. If we do not use VIP Lanes, when the traffic arrives on SW2, it is directed to SW6 and then forwarded to the normal campus network, going through performance-limiting middleboxes, the campus edge router, and finally delivered out to the Internet. We notice that the middleboxes on this normal path will reduce the big data transfer performance. If the VIP Lanes system is used, the traffic will go directly from SW3, SW4 to the campus edge router, which bypasses the middleboxes. Similar to our previous experiment, we applied PBR on the Cisco-L3 switch to redirect flows towards the SDN core (SW4).

The reason for applying Policy Based Routing on the Cisco-L3 switch was to make sure that the traffic matching on the policy, such as the trusted flows, would be redirected to follow a path that contains at least one SDN-capable switch. To fulfill the goal, we took advantage of SW5, which was a switch connected to the SDN-capable switch SW3. The path from Cisco-L3 to SW5 was configured as a trunk for the same VLAN so that the interface on SW5 (marked red) could be used as the next-hop IP address in Policy Based Routing. This was the configuration we used without modifying the existing topology. By default, the traffic would go through the normal campus network. After applying PBR, the traffic was redirected to SW5 where SDN-capable switch SW3 was on the path. When the traffic arrived on SW3, it followed the SDN rules installed and was redirected to SW4. So the path control has been realized by means of Policy Based Routing and SDN rules.

In the experiment, we measured the throughput from a host machine on the University of Kentucky (UK) campus network to four remote machines that are located at different sites of Internet2. We picked four remote sites from the ESnet throughput test hosts list based on their geographical distances from Kentucky. The selected sites were `chic-pt1.es.net` (Chicago), `atla-pt1.es.net` (Atlanta), `hous-pt1.es.net` (Houston) and `ga-pt1.es.net` (San Diego). The tool we used for the tests was `perfSONAR`, which internally runs iperf3 for the measurement. For each site, we sent traffic through both the UK campus core network and the SDN network that bypasses the middleboxes. Both end-systems were equipped with a 10Gbps network interface card (NIC). For both the normal path going through the UK campus network and the middlebox-free SDN path, we ran 10 tests to each of the four remote sites, the duration per test was 30 seconds.

TABLE I
THROUGHPUT COMPARISON TO DIFFERENT SITES USING NORMAL AND SDN PATHS

Sites	*Normal (Gbps)*		*SDN (Gbps)*		
	Mean	SD	Mean	SD	**Speedup**
San Diego, CA	1.28	0.26	7.97	0.01	**6.2x**
Houston, TX	1.90	0.29	8.78	0.46	**4.6x**
Atlanta, GA	1.97	0.28	8.74	0.31	**4.4x**
Chicago, IL	2.59	0.36	9.52	0.30	**3.6x**

As the results in Table I show, the average performance we got for the traffic going through the SDN middlebox-free path was much better than that of the traffic going through the normal campus network. The best average performance we got was for the traffic we sent to the Chicago site, which reached 9.52Gbps. This number is very close to the maximum speed the network interface card (NIC) supports. The San Diego site was the one to which we got the lowest performance for both the SDN network and the normal campus network. Even though it was expected, because San Diego is the most far away site among the four from the UK campus, this site yields the highest speedup factor across the four sites. For the standard deviation, the results for the SDN path and the normal path did not vary significantly, remaining to be consistent with the measurements we presented in our initial prototype. We also observed from the raw data (not shown here) that the number of retransmissions for the SDN path was always very low, while for the transfers going through the normal campus network, a low throughput was always accompanied with a large number of retransmissions. Our test results clearly show that the control has been realized on the legacy portion of the network for the hosts to get good performance when they transfer big data.

VI. Related Work

There are many topics in hybrid networks. Some focus more on the architecture and design problem, while others focus on the virtualization problem such as building controllers for hybrid networks.

Levin et al. proposed the Panopticon [14] network architecture where at least one SDN-capable switch is deployed on any source-destination path. In that way, every packet that traverses the network is forced to go through an SDN-capable switch and further forwarded to the SDN controller for appropriate processing as if the packet were in a pure SDN environment. Unlike their work, our hybrid SDN network does not intercept packets at the control plane but rather proactively deploys PBR and OpenFlow entries proactively to maximize performance. HybNET [15] is a framework that automates the management in a hybrid network environment where a configuration mechanism translates legacy network configuration into OpenFlow configuration. By means of virtualization using VLANs, the SDN portion of the network is in charge of network management whereas legacy switches are only used as forwarding devices.

Telekinesis [16] provides fine-grained path control over legacy paths. They use OpenFlow to instruct SDN switches to send non-standard *LegacyFlowMod* packets to legacy switches, forcing them to update their forwarding tables. The authors addressed limitations regarding overhead and instability of the calculated network paths in [4] but their approach still relied on unconventional indirect management. Instead, we leveraged existing mechanisms included in a limited set of legacy switches to pick off flows and route them towards the SDN network without indirect mechanisms.

Lastly, ClosedFlow [17] proposed an approach that leverages the fact that each node in an OSPF network has information about the entire network. Similar to our work, they used remote access tools (SSH) and built-in PBR to deploy fine-grained rules. However, we addressed topology discovery in a less intrusive fashion by using SNMP to query information gathered by discovery protocols such as CDP or LLDP (and their MIBs) as opposed to their per-device configurations.

VII. Conclusions

Deploying SDN networks and applications in a production network faces many non-technical and technical challenges. We described our experience working with campus IT to build a trust relationship and develop a set of principles so that the campus IT department allows us to develop and deploy applications that have a shared control over certain flows in the production campus network. We used the VIP Lanes system as an example to show that it can take advantage of the shared control offered through SDN to provide high-speed data transfer for big data applications on our campus network. We demonstrated our approach to integrating the legacy region of the campus network into our system so that it can provide the same service to those users connected with legacy routers.

Acknowledgment

This work was supported in part by the National Science Foundation under Grants ACI-1541380, ACI-1541426, ACI-1642134, and CNS-1551453. The work of Calvert was supported while he was on assignment at the National Science Foundation.

References

[1] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *arXiv preprint arXiv:1406.0440*, 2014.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3] B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol," Internet Requests for Comments, RFC Editor, RFC 7047, December 2013. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7047.txt

[4] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, "Magneto: Unified fine-grained path control in legacy and openflow hybrid networks," in *Proceedings of the Symposium on SDN Research*. ACM, 2017, pp. 75–87.

[5] J. Griffioen, K. Calvert, Z. Fei, S. Rivera, J. Chappell, M. Hayashida, C. Carpenter, Y. Song, and H. Nasir, "VIP Lanes: High-speed custom communication paths for authorized flows," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–9.

[6] S. Rivera, M. Hayashida, J. Griffioen, and Z. Fei, "Dynamically creating custom SDN high-speed network paths for big data science flows," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. ACM, 2017.

[7] National Science Foundation, "NSF 15-534: Campus cyberinfrastructure – data, networking and innovation program (CC*DNI)," https://www.nsf.gov/pubs/2015/nsf15534/nsf15534.htm, 2015.

[8] Hewlett Packard Enterprise, "HP Virtual Application Networks SDN Controller," https://h17007.www1.hpe.com/docs/networking/solutions/sdn/4AA4-8807ENW.PDF, 2013.

[9] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a Model-Driven SDN Controller Architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–6.

[10] Project Floodlight, "Open source software for building software-defined networks," https://www.projectfloodlight.org/floodlight/, 2019.

[11] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6.

[12] C. Jaquier, "Fail2ban," https://github.com/fail2ban/fail2ban, 2019.

[13] "Cisco IOS Software Configuration Guide," https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/15-0SY/configuration/guide/15_0_sy_swcg/policy_based_routing_pbr.pdf.

[14] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, 2014, pp. 333–345.

[15] H. Lu, N. Arora, H. Zhang, C. Lumezanu, J. Rhee, and G. Jiang, "Hybnet: Network manager for a hybrid network infrastructure," in *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference*, ser. Middleware Industry '13. New York, NY, USA: ACM, 2013, pp. 6:1–6:6. [Online]. Available: http://doi.acm.org/10.1145/2541596.2541602

[16] C. Jin, C. Lumezanu, Q. Xu, Z.-L. Zhang, and G. Jiang, "Telekinesis: Controlling legacy switch routing with openflow in hybrid networks," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15. New York, NY, USA: ACM, 2015, pp. 20:1–20:7.

[17] R. Hand and E. Keller, "Closedflow: Openflow-like control over proprietary devices," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 7–12.