Carving Research Slices Out of Your Production Networks with OpenFlow *

Rob Sherwood
† DT Inc. R&D Lab
5050 El Camino Real
Los Altos, CA USA
robert.sherwood@telekom.de

Glen Gibb
⋆ Stanford University
351 Gates Building
Palo Alto, CA USA
grg@stanford.edu

Masayoshi Kobayashi
◇ NEC Labs
Tokyo, Japan
mkobaya1@stanford.edu

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Management, Design, Experimentation

Keywords

Flowvisor, OpenFlow, Architecture, Virtual networks, Slicing

1. SLICED PROGRAMMABLE NETWORKS

OpenFlow [6] has been demonstrated as a way for researchers to run networking experiments in their production network. Last year, we demonstrated how an OpenFlow controller running on NOX [4] could move VMs seamlessly around an OpenFlow network [1]. While OpenFlow has potential [3] to open control of the network, only one researcher can innovate on the network at a time. What is required is a way to divide, or *slice*, network resources so that researchers and network administrators can use them in parallel. Network slicing implies that actions in one slice do not negatively affect other slices, even if they share the same underlying physical hardware. A common network slicing technique is VLANs. With VLANs, the administrator partitions the network by switch port and all traffic is mapped to a VLAN by input port or explicit tag. This coarse-grained type of network slicing complicates more interesting experiments such as IP mobility or wireless handover.

Here, we demonstrate FlowVisor, a special purpose OpenFlow controller that allows multiple researchers to run experiments safely and independently on the same production OpenFlow network. To motivate FlowVisor's flexibility, we demonstrate five network slices running in parallel: one slice for the production network and four slices running experimental code. Our demonstration runs on real network hardware deployed on our production network[1] at Stanford and a wide-area test-bed with a mix of wired and wireless technologies.

2. FLOWVISOR ARCHITECTURE

Architecturally, FlowVisor acts as a transparent virtualization layer between OpenFlow switches and controllers. Network devices generate OpenFlow protocol messages, which go to the FlowVisor and are then routed by network slice to the appropriate researcher(s) (Figure 1). OpenFlow messages from researcher controllers are vetted by the FlowVisor to ensure that the isolation between slices is maintained before being forwarded to switches. Thus, the FlowVisor appears as a virtual controller to the switches and as a network of virtual switches to the researcher controllers.

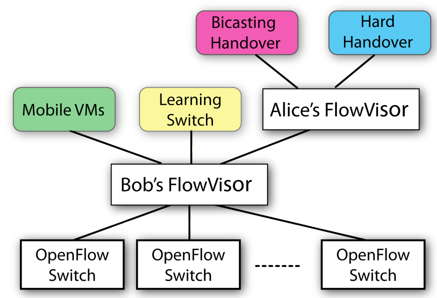

Figure 1: FlowVisor allows multiple researchers to operate in parallel on slices of an OpenFlow network. FlowVisor acts as a transparent proxy between network devices, OpenFlow controllers, and other FlowVisor's.

FlowVisor is intentionally architecturally neutral: it makes no assumption about the function or operation of the switches or controllers, save that they speak OpenFlow. We architect FlowVisor as a transparent layer for three reasons:

Centralized policy enforcement. All control traffic, from switch to controller and from controller to switch, traverses the FlowVisor. This provides FlowVisor a complete view of the network's state and allows it to enforce policy by dropping or rewriting OpenFlow control messages. Additionally, centralizing policy decisions makes it easier to reason about the set of allowable actions and debug errors should they occur.

Recursive delegation is the ability to re-delegate control of a subset of a network slice. Because FlowVisor acts as a transparent proxy, it is possible to cascade FlowVisor instances, making recursive delegation trivial. We expect recursive delegation will be an important property for virtual networks as it eases network administration overhead and improves resource allocation.

Decouple control and virtualization technologies. Rather than building virtualization support directly into the OpenFlow protocol itself, we intentionally keep the control and virtualization aspects orthogonal. This allows each technology to evolve independently, avoiding new forms of ossification.

FlowVisor defines slices along any combination of ten packet header fields, including physical layer (switch ports), link layer

*This work is supported in part by NSF, Cisco, Deutsche Telekom, DoCoMo, Ericsson, NEC and Xilinx.

[1] That is, the network where the authors read their daily mail, surf the web, etc.

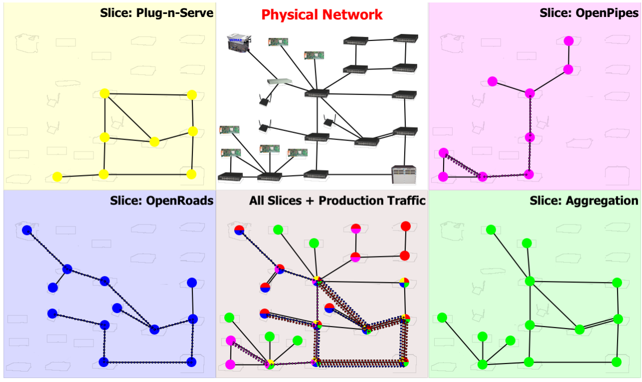

Figure 2: We present a slice monitoring program that graphically displays flows in real time. Each corner represents an experimental slice, the middle top frame the physical topology, and the middle bottom frame shows an aggregated "administrative" view.

(src/dst mac addresses, ether type), network layer (src/dst IP address, IP protocol), and transport layer (src/dst UDP/TCP ports). Additionally, FlowVisor slices can be defined with negation ("all packets *but* TCP packets with dst port 80"), unions ("ethertype is ARP *or* IP dst address is 255.255.255.255"), or intersections ("netblock 192.168/16 *and* IP protocol is TCP"). We believe that such fine-grained slicing will be a useful tool for network researchers and administrators alike.

3. TRIAL DEPLOYMENT

To motivate its utility, we deploy the FlowVisor on our production network and run four experiments in tandem with our everyday traffic. We create a custom real-time slice monitoring tool(Figure 2) to help visualize our network. The tool dynamically shows the testbed topology with flows color-coded by experiment . The monitoring tool displays a simultaneous view of the entire physical network topology (Figure 2,top middle) and the virtual topology corresponding to each slice.

The four experiments are chosen to show the diversity of experiments that FlowVisor supports, and will each be running an isolated slice specifically "carved" for its needs.

Plug-N-Serve tested [5] various algorithms for load-balancing web requests in unstructured networks. In this experiment, web queries arrive at a configurable rate and are load-balanced both by path and by server. The Plug-N-Serve experiment's query generator tested FlowVisor's new flow switch CPU isolation features.

OpenRoads experiments [8, 7] with loss-less handover between the WiMAX and wifi wireless nodes. By dynamically re-directing how traffic flows through the network, OpenRoads is able to provide finer-grained control of mobility policies, e.g., make-before-break or break-before-make connections. OpenRoads made heavy use of "read-only" FlowSpace, testing the FlowVisor's traffic isolation capabilities.

Aggregation demonstrates OpenFlow's ability to define flows as groups of TCP sessions. In the experiment, hundreds of TCP flows are "bundled" into a single flow table entry. The aggregation experiment's slice definition had 512 rules, testing the FlowVisor's processing and rewriting capabilities.

OpenPipes demonstrates [2] how hardware designs can be partitions over a physical network. In this experiment, the traffic consisted of video frames encapsulated in raw ethernet and was piped through various video filters running on nodes distributed through the network. OpenPipe's traffic stressed the FlowVisor flexible FlowSpace slicing in terms of its ability to slice by ethernet type.

4. ADDITIONAL AUTHORS

A small army of authors contributed to this demo (refer to title for affliations):

Michael Chan ⋆	Adam Covington⋆	Mario Flajslik ⋆
Nikhil Handigol⋆	Te-Yuan Huang ⋆	Peyman Kazemian⋆
Jad Naous ⋆	Srinivasan Seetharaman†	David Underhill⋆
Tatsuya Yabe ◇	Kok-Kiong Yap ⋆	Yiannis Yiakoumis⋆
Hongyi Zeng⋆	Guido Appenzeller ⋆	Ramesh Johari ⋆
Nick McKeown⋆	Guru Parulkar⋆	

5. REFERENCES

[1] D. Erickson et al. A demonstration of virtual machine mobility in an OpenFlow network. In *Proceedings of ACM SIGCOMM (Demo)*, page 513, Seattle, WA, Aug. 2008.

[2] G. Gibb, D. Underhill, A. Covington, T. Yabe, and N. McKeown. OpenPipes: Prototyping high-speed networking systems. In *"Proc. ACM SIGCOMM Conference (Demo)"*, Barcelona, Spain, August 2009.

[3] K. Greene. Special reports 10 emerging technologies 2009. *MIT Technology Review*, 2009. `http://www.technologyreview.com/biotech/22120/`.

[4] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards and operating system for networks. In *ACM SIGCOMM Computer Communication Review*, July 2008.

[5] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari. Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow. In *ACM SIGCOMM Demo*, August 2009.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[7] K.-K. Yap and et al. Lossless Handover with n-casting between WiFi-WiMAX on OpenRoads. In *ACM Mobicom (Demo)*, 2009.

[8] K.-K. Yap, M. Kobayashi, R. Sherwood, N. Handigol, T.-Y. Huang, M. Chan, and N. McKeown. OpenRoads: Empowering research in mobile networks. In *Proceedings of ACM SIGCOMM (Poster)*, Barcelona, Spain, August 2009.