Carving Research Slices Out of Your Production Networks with OpenFlow

Rob Sherwood, Michael Chan,Glen Gibb, Nikhil Handigol,Te-Yuan Huang,Peyman Kazemian, Masayoshi Kobayashi, David Underhill, Kok-Kiong Yap, Guido Appenzeller, and Nick McKeown
Deutsche Telekom Inc. R&D Lab, Stanford University, NEC System Platforms Research Labs

1. SLICED PROGRAMMABLE NETWORKS

OpenFlow [4] 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 [3] could move VMs seamlessly around an OpenFlow network [1]. While OpenFlow has potential [2] 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 four network slices running in parallel: one slice for the production network and three slices running experimental code (Figure 1). 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.

Categories and Subject Descriptors: C.2.2 – Computer Systems Organization **[Computer-Communication Networks]**: Network Architecture and Design; C.4 – Computer Systems Organization **[Performance of Systems]**

General Terms:
Management, Design, Experimentation

Keywords:
Flowvisor, OpenFlow, Architecture, Virtual networks, Network slicing

2. FLOWVISOR ARCHITECTURE

Architecturally, FlowVisor acts as a transparent proxy. 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.

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

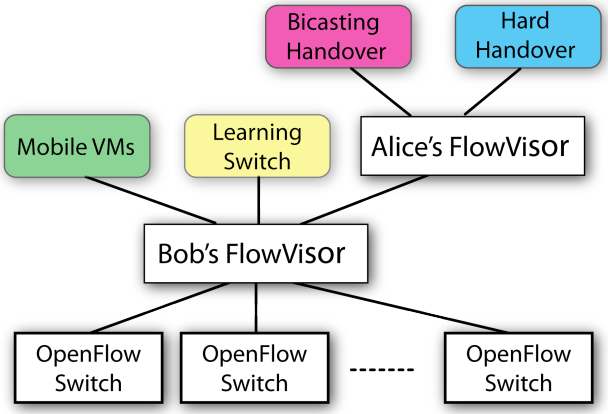

Figure 1: FlowVisor allows multiple researchers to operate in parallel on slices of an OpenFlow network. FlowVisor acts as a transparent proxy between network devices, OpenFlow controllers, and other FlowVisor's.

Thus, the FlowVisor appears as a virtual controller to the switches and as a network of virtual switches to the researcher controllers.

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 proxy 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 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

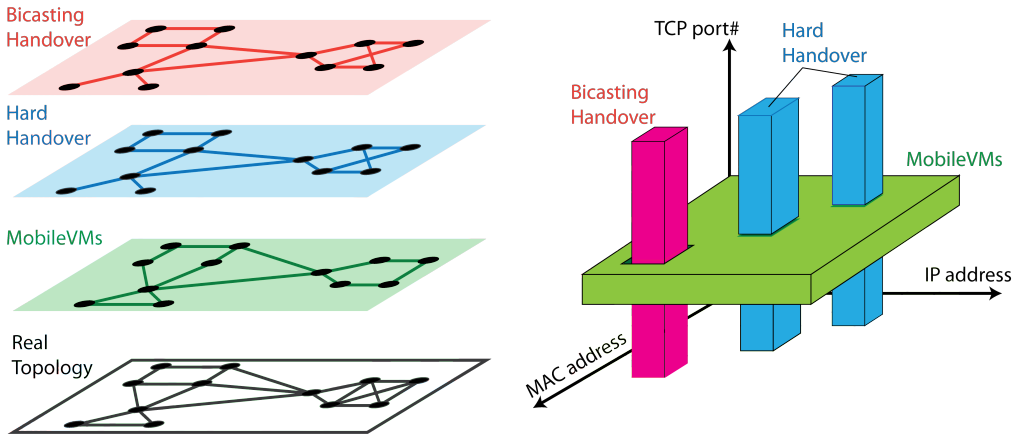

Figure 2: We present a monitoring program that graphically displays flows in real time in their respective network slice. Slices are defined by union, intersection, and difference of 10 packet fields—three of which are shown here.

aspects orthogonal. This allows each technology to evolve independently, avoiding new forms of ossification.

We believe that our demonstration highlights these architectural features.

3. DEMONSTRATION

To visualize our four experiments, we display a custom slice monitoring tool running on a large screen (Figure 2). The tool dynamically shows in real time the test-bed topology and color-coded flows from each experiment . The monitoring tool displays a simultaneous view of the entire physical network topology (Figure 2,bottom layer) and the virtual topology corresponding to each slice.

We further demonstrate that FlowVisor has flexible and fine-grained network slices control. These slices can be recursively delegated (Figure 1). In our demonstration, Bob delegates a network slice to Alice, who in turn re-delegates it to the Bicast experiment. Further, FlowVisor allows slices to be defined along any combination of ten packet header fields (Figure 2), including physical layer (switch ports), link layer (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.

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

Learning Switch A controller that performs the standard MAC-address learning switch algorithm. The learning switch's network slice is defined as the default, that is, any flow that does not belong to any other experiment is part of this slice. We include the learning switch to demonstrate how research and non-research traffic can safely co-exist.

Mobile VMs In this experiment, a virtual machine (VM) running a latency-sensitive video game is migrated live between servers while maintaining the same IP address. The server and video game software remain unchanged; an OpenFlow controller performs dynamic re-writing of the traffic so that connectivity is maintained. This experiment's slice is defined as all game traffic on a specific UDP port. This experiment won the SIGCOMM 2008 Best Demo award [1].

Hard Handover mobility agent This OpenFlow controller manages a pair of mobile laptops running at Stanford. We demonstrate that with OpenFlow, it is possible to re-route flows dynamically and seamlessly hand-off between an 802.11 access point and a WiMAX base station. Hard handover's network slice is defined as the packets destined to the MAC addresses of the mobile laptops. To visualize the effect of the handover, we display a video streamed to the laptops.

Bicasting mobility agent Setup similarly to hard handover, this controller manages the traffic for a second pair of mobile laptops. In this experiment, we demonstrate the effect of bi-casting, i.e., duplicating packets along two independent links. To visualize the effects of bicasting, we display a video streaming to the laptops.

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

[3] 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.

[4] 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.