OpenRoads: Empowering Research in Mobile Networks

Kok-Kiong Yap*, Masayoshi Kobayashi◇, Rob Sherwood†,
Te-Yuan Huang*, Michael Chan*, Nikhil Handigol*, and Nick McKeown*
*Stanford University, ◇NEC System Platforms Research Labs, and †Deutsche Telekom Inc., R&D Lab
yapkke@stanford.edu, m-kobayashi@eo.jp.nec.com, robert.sherwood@telekom.com,
{nikhilh,huangty,mcfchan,nickm}@stanford.edu

ABSTRACT

We present OpenRoads, an open-source platform for innovation in mobile networks. OpenRoads enable researchers to innovate using their own production networks, through providing an wireless extension OpenFlow. Therefore, you can think of OpenRoads as "OpenFlow Wireless".

The OpenRoads' architecture consists of three layers: flow, slicing and controller. These layers provide flexible control, virtualization and high-level abstraction. This allows researchers to implement wildly different algorithms and run them concurrently in one network. OpenRoads also incorporates multiple wireless technologies, specifically WiFi and WiMAX. We have deployed OpenRoads, and used it as our production network. Our goal here is for those to deploy OpenRoads and build their own experiments on it.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Design, Experimentation, Verification

Keywords

Wireless Testbed, OpenFlow, FlowVisor

1. BACKGROUND

At Sigcomm 2009, we demonstrated *Carving Research Slices Out of Your Production Networks with OpenFlow* [6], which is described in this issue of *ACM CCR*. The demonstration showed how we deployed OpenFlow into part of our Stanford campus network, then "sliced" the network five ways; four slices contained experiments, and the fifth slice carries legacy production traffic. Four of the slices used wireline switches, while one slice - the OpenRoads slice - had WiFi and WiMAX nodes too.

You can think of OpenRoads as "OpenFlow Wireless" – a wireless extension to our OpenFlow testbed. The WiFi access points and WiMAX basestation contain a flow-table, which is controlled remotely via the OpenFlow protocol. Our goal is to create a high quality open-source wireless network platform that, as a community, we can all deploy in our college campuses. OpenRoads is designed to replace the existing WiFi (and WiMAX) networks; or run in parallel. At Stanford, we have deployed 85 WiFi access points and 2 WiMAX basestation running OpenFlow as a parallel network used for all traffic in a small number of research groups [8]; in time we plan to replace the production wireless network in three buildings.

So far, we have created a handful of mobility managers (some of which were created as part of a class) and run them concurrently in our network. The mobility managers include hard handover, informed handover, n-casting and Hoolock. In informed handover, the client is provided with information to aid handover decisions. In n-casting, flows are multicasted to the client over multiple independent paths to reduce loss. Hoolock is a lossless handover scheme which exploits multiple interfaces on a client. The point here is not to argue that any particular handover method is better than the others; rather, our goal is to show how different researchers could create and then run their own mobility manager at the same time. All of our mobility managers are able to perform handover between WiFi and WiMAX without any code change. In these mobility managers, the OpenRoads' platform allows us to redirect flows in just 12 lines of C/C++.

2. ARCHITECTURE

The OpenRoads architecture is shown in Fig. 1, and consists of three layers: The flow layer, the slicing layer, and the controller layer.

Flow Layer: In the flow layer, the datapath flow-table is managed using the OpenFlow protocol in the usual way. Medium-specific parameters (e.g. channel, power levels, SSID) are controlled and monitored using SNMP. To open up the production network to researchers, OpenRoads provides researchers control of the datapath using OpenFlow and control of the device configuration using SNMP.

Slicing Layer: The datapath of our OpenRoads network is sliced using the FlowVisor [6]. FlowVisor is an OpenFlow virtualization layer that appears to the switches as a transparent controller proxy. The switches believe they are controlled by the FlowVisor; which in turn hosts multiple guest controllers, each of which believes it controls all the switches. The FlowVisor intercepts OpenFlow protocol messages to maintain the slicing according to a policy written by the network administrator. FlowVisor is an open-source tool [1], and could easily be replaced by a single controller, or by an alternative slicing layer.

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

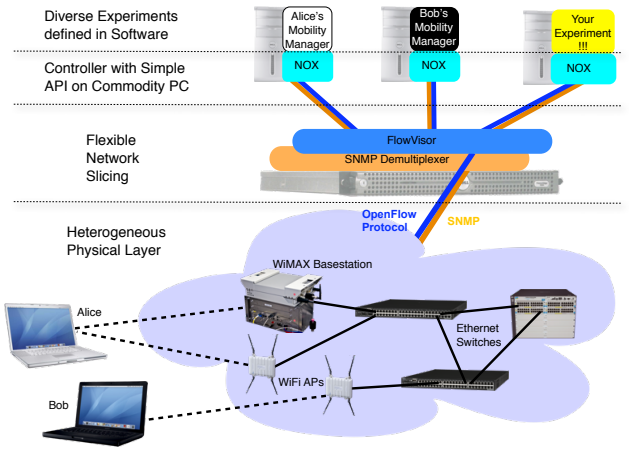


Figure 1: OpenRoads' Architecture

We also need to slice the SNMP configuration of each access point and basestation. For this, we created an SNMP demultiplexer.

Each experiment is given a "flow-space" (a range of packet header values). By ensuring that experiments cannot exert control beyond their designated "flow-space", FlowVisor provides isolation between experiments. Also, by appearing as a controller to network devices and as a device to the controllers, FlowVisor (and the SNMP demultiplexer) allows researchers to develop experiments as if they have complete control over the entire network. This allows the scale of the experiment to be expanded without any change in code.

Controller Layer: All of our experiments are built on top of the NOX [2] network-wide operating system. NOX provides a network-wide view of topology and state in the network, and allows an application to make decisions and cache them in the flow-tables within the datapath. Like FlowVisor, NOX is an open-source tool [4] and could easily be replaced by an alternative controller.

The OpenRoads architecture promotes a simple flow layer, pushing many complexities to the controller layer. Not only does this allows researchers to implement wildly different routing algorithms and mobility managers, it provides a way in which many of these algorithms can be operated simultaneously on a physical network. Our production network has simultaneously run several of our mobility managers in the course of our work, a feature we vividly demonstrated in the demonstration [6].

3. EXAMPLE

Our poster (and the demo described [6]) show one simple example of how OpenRoads can be used. In our example, a video is streamed from a video server to a wireless client. To emulate the behavior under loss, we instructed the WiFi APs to drop 7% of the packets, leading to a poor quality video stream at the client. Our n-casting experiment runs as an applications on the NOX controller (written in 227 lines of C/C++ code) – when it detects high loss in the network, it selects a switch in the topology to bicast the traffic over two different paths. Both copies of the packets arrive at the client over different wireless channels (the client has two WiFi and one WiMAX interface), and its bonding driver stitches the stream back together and discards duplicates. If the loss gets worse, the n-casting application tricasts the video over WiMAX as well. The goal here is not to suggest that all videos should be n-cast; but to demonstrate that routing and mobility can easily be made application-specific - perhaps even under the control of the application itself. A video of this demonstration can be found at [7].

4. DISSEMINATION

OpenRoads, or "OpenFlow Wireless" is developed to support our vision for the future mobile Internet [3], where one can move seamlessly between different radio technologies and radio technologies. We present OpenRoads as an open-source platform [5] to the community at large. We hope it will inspire others to deploy OpenRoads in their network, and build their own experiments.

5. REFERENCES

[1] Flowvisor. `http://openflowswitch.org/wk/index.php/FlowVisor`.

[2] 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.

[3] Kok-Kiong Yap, Rob Sherwood, Masayoshi Kobayashi, Nikhil Handigol, Te-Yuan Huang, Michael Chan, Nick McKeown, and Guru Parulkar . Blueprint for Introducing Innovation into the Wireless Networks we use every day. Technical report, Stanford University, Deutsche Telekom R&D Lab, and NEC. Available from `http://OpenFlowSwitch.org/downloads/technicalreports/openflow-tr-2009-3-openflow-wireless.pdf`.

[4] NOX: An OpenFlow Controller. `http://noxrepo.org/wp/`.

[5] OpenFlow Wireless. `http://www.openflowswitch.org/wk/index.php/OpenRoads`.

[6] Rob Sherwood, Michael Chan, Adam Covington, Glen Gibb, Mario Flajslik, Nikhil Handigol, Te-Yuan Huang, Peyman Kazemian, Masayoshi Kobayashi, Jad Naous, Srinivasan Seetharaman, David Underhill, Tatsuya Yabe, Kok-Kiong Yap, Yiannis Yiakoumis, Hongyi Zeng, Guido Appenzeller, Ramesh Johari, Nick McKeown, and Guru Parulkar. Carving research slices out of your production networks with OpenFlow. In *Proceedings of ACM SIGCOMM (Demo)*, Barcelona, Spain, August 2009.

[7] n-casting using OpenFlow. `http://www.openflowswitch.org/wp/n-casting-mobility-using-openflow/`.

[8] K.-K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. McKeown. The Stanford OpenRoads Deployment. In *WiNTECH '09: Proceedings of the fourth ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. ACM, 2009.

Stanford University

OpenRoads:Empowering Research in Mobile Networks

Kok-Kiong Yap, Masayoshi Kobayashi, Nikhil Handigol, Te-Yuan Huang, Michael Chan, Rob Sherwood, and Nick McKeown

Stanford, NEC, and Deutsche Telekom

Goals

- Enable innovation in campus networks
- Create open platform that we can **all** use and innovate on

Approach

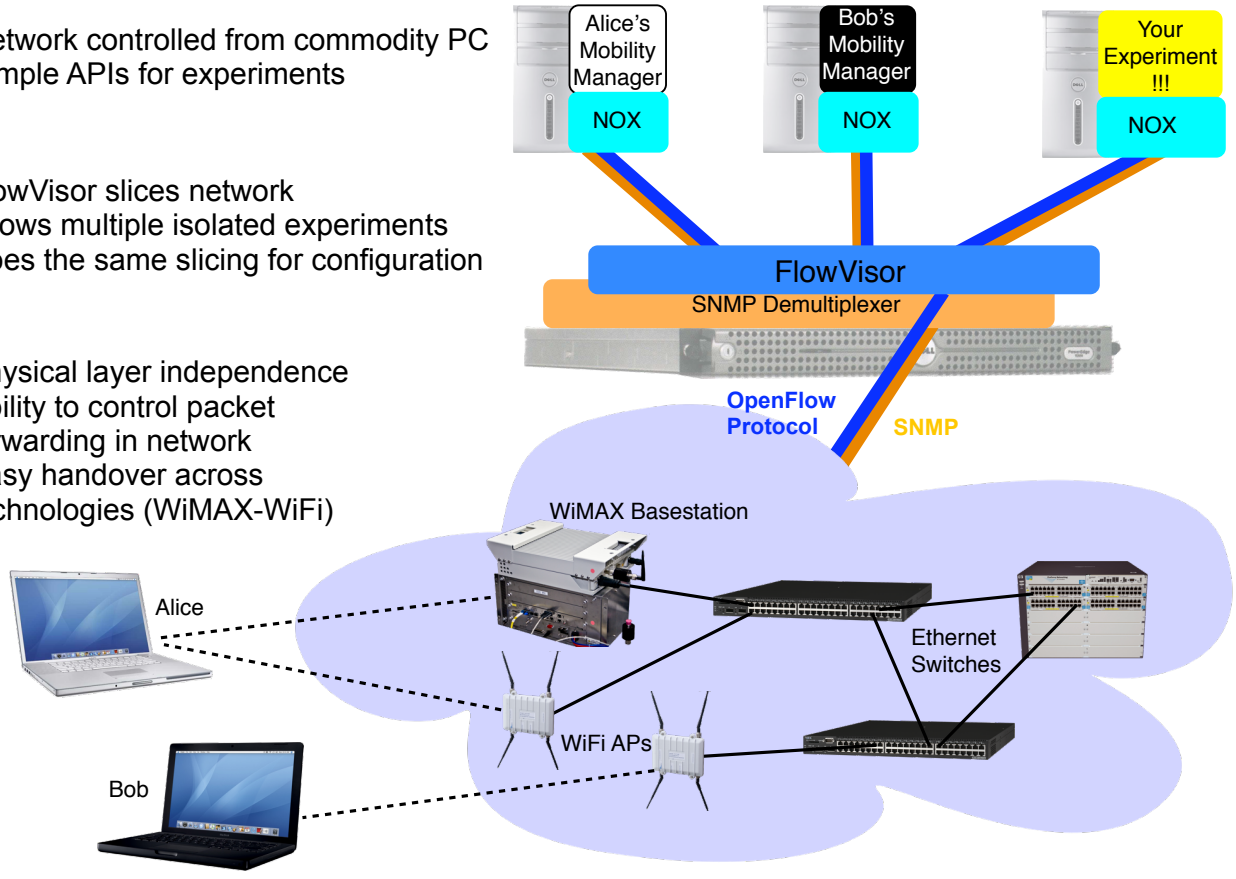
- Work with many equipment vendors
- Add OpenFlow to switches, APs, and basestations
- Deploy across Stanford
- Replicate at other campuses
- Share our work (http://OpenFlowSwitch.org)

OpenRoads Architecture

- Network controlled from commodity PC
- Simple APIs for experiments

- FlowVisor slices network
- Allows multiple isolated experiments
- Does the same slicing for configuration

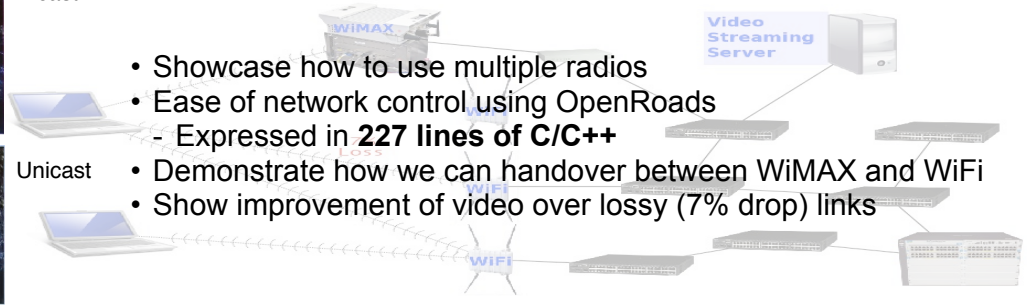
- Physical layer independence
- Ability to control packet forwarding in network
- Easy handover across technologies (WiMAX-WiFi)



n-casting on OpenRoads


Tricast


Unicast



- Showcase how to use multiple radios
- Ease of network control using OpenRoads
  - Expressed in **227 lines of C/C++**
- Demonstrate how we can handover between WiMAX and WiFi
- Show improvement of video over lossy (7% drop) links

n-casting on OpenRoads runs on top of the FlowVisor, as part of the "Carving Research Slices Out of Your Production Networks with OpenFlow" demonstration at SIGCOMM 2009, sharing the same physical network with other experiments and production traffic.

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