Lossless Handover with n-casting between WiFi-WiMAX on OpenRoads

Kok-Kiong Yap, Te-Yuan Huang, Masayoshi Kobayashi, Michael Chan,
Rob Sherwood, Guru Parulkar and Nick McKeown
Stanford University, NEC, Deutsche Telekom R&D Lab
{yapkke,huangty}@stanford.edu,m-kobayashi@eo.jp.nec.com,
mcfchan@stanford.edu,robert.sherwood@telekom.com,{parulkar,nickm}@stanford.edu

1. INTRODUCTION

We envisioned a future mobile wireless Internet with many radios, and many networks (in POMI [3]). By that, we mean many radios will be found in a future mobile device. It is now common for a handheld to have WiFi, bluetooth, GSM and/or 3G radios, with WiMAX and LTE in the horizon and many more to come. With shrinking geometries and increasing power efficiency of the radios, we can expect multiple radios (some of the same kind) to be installed into a single handheld. Coupled with the availability of many networks around us, we can connect to multiple networks simultaneously using the multiple radios in our handhelds. What this means is that we can exploit multihoming solutions or increase the robustness of the channel via duplication of packets over multiple interfaces. And with make-before-break handover, we have achieved seamless pervasive connectivity at all times.

To achieve our vision, many difficulties has to be overcome. One striking difficulty is our inability to fluidly switch between the many networks around us. The backbone network for WiFi and WiMAX are distinctively different. While typical WiFi uses packet-oriented Ethernet/IP, the WiMAX backbone handles packets using the idea of service flows. Hence, a vertical handover between the two backbone network is complicated and highly specialized between the two technologies. By building mobile wireless networks around a specific radio technology, we hinders the rapid incorporation of new wireless technologies. A sub-optimal system results due to such architectural barrier. Further, it is typically difficult to control forwarding decisions in these datapath elements, making it hard to redirect flows between the networks, a basic need in mobility management.

To overcome this, we propose "flattening" of the mobile wireless network, where multiple wireless technologies are connected via an unified network substrate. One where the forwarding decisions can be effectively and flexibly controlled. By using this unified network substrate, we can build networks consisting of heterogeneous wireless technologies, by incorporating "dumb" wireless termination points, as we will explain in the next section. We will showcase the feasibility and desirability of this approach in this demonstration, by presenting our n-casting mobility manager built on such a network. Our n-casting mobility manager will show how we seamlessly switch between WiFi-WiFi and WiFi-WiMAX radios, providing a glimpse of the future we envision.

There is more than we could possibly do on this path towards such a network. We would like to engage the broader research community in our expedition. To seed this movement, we developed *OpenRoads* [1]: a mobile wireless platform for experimental research and realistic deployments of networks and services. By making this platform available to the community, we hope to facilitate rapid innovation in the field, through enabling realistic testing and verification of ideas. Further, OpenRoads implements our proposed "flattened" mobile wireless network architecture, allowing us to incorporate novel wireless technologies easily.

In the next section (§2), we will provide primers on technological components of OpenRoads, and describe how our n-casting mobility manager. In §3, we will present an outline of our demonstration, ending with logistics in §4.

2. N-CASTING ON OPENROADS

2.1 OpenRoads' Primers

In OpenRoads, we use OpenFlow that allows switches, routers, WiFi APs, base stations to be controlled by an external controller. Almost all these devices has an internal flow table (originally used for holding firewall ACLs). By exposing an external standardized interface for manipulating the flow table, OpenFlow allows an external controller (in our case, NOX [2]) to control the forwarding of packets in the datapath. This allows for "software defined networking", where the logic for network operation is all done in software. Specifically, the separation of control and datapath means that both the WiMAX and WiFi backhaul can be efficiently programmed in the same controller. The idea of service flows in WiMAX can be easily maintained in the controller, with little change to the datapath (i.e., switches) themselves. This provides a unified backbone for the numerous wireless datapath technologies, allowing for integration of wildly different wireless technologies.

Through this control and datapath separation using OpenFlow, we have also paved the way for an efficient way to share such a network. Since we can now implement network logic in software within an external controller, we can police the messages to and from these controllers to enforce a specified policy in the physical network. This allows multiple controllers/experiments to coexist in the network, each managing a slice of the network (i.e., a certain set of flows or range of headers). For this purpose, we developed the FlowVisor [4], where each controller is allowed read-/write accesses to a certain set of flows. As a transparent proxy, the FlowVisor appears as a controller to the datapath, while appearing as a "private" network to the controllers. Such a capability is critical to a network for experimental research and realistic deployments of networks and services.

In OpenRoads, we went on further to augment the controller (i.e., NOX) with APIs which makes developing mobility managers easier. As a first foray, we deployed this network in our campus and extended OpenFlow into WiFi APs and NEC WiMAX base stations. Here, we have customized the WiMAX base station to behave as a dumb WiMAX AP, providing only the wireless connectivity. This provides a showcase of the ease of incorporating wildly different wireless technologies into our platform.

2.2 *n*-casting

To exemplify the feasibility and desirability of our proposed approach, we developed *n*-casting to showcase how we can use many radios over multiple wireless technologies in this "flattened" network.

An obvious way to use multiple radios in a single handheld is to transmit/receive on all the radios simultaneously. This naive approach can be used to increase bandwidth (i.e., different radios use different frequencies and APs to transmit different packets/flows) or increase robustness (i.e., the same packets are sent down to all the interfaces). Intermediates can be achieved by doing coding (like network coding or fountain code) on the packets sent to the different interfaces. In our case, we would demonstrate the latter, which we call *n*-casting as the same packets (duplicated in the network) are sent to all the *n* interfaces of a device. In this demonstration, we will show *n*-casting being performed over arbitrarily set of WiFi and WiMAX connections.

Our demonstration here is a showcase of the flexibility and capability of our platform. We were able to implement *n*-casting in 227 lines of C/C++ code, something which would take orders of magnitude more effort in the conventional network. Further, no additional code is required for the code to handle WiFi-WiMAX connectivity. In our deployment, the *n*-casting experiment is run on our production network, where multiple controllers can control different slices of the network. All these are only possible due to the simple (and thus powerful) network architecture we have adopted.

3. DEMONSTRATION OF *N*-CASTING

To show the increased robustness due to *n*-casting, we demonstrate how it can be used to improve the quality of a streaming video on a laptop with multiple network interfaces. To enhance the visual perception of the improvement, we will inject loss on our wireless links to emulate a lossy network. Our demonstration consists of two parts: The first part is a remote demonstration of WiFi-WiMAX tricasting on the OpenRoads deployment in our campus. The second part is an onsite demonstration which allows audience to interact and control the flows directly through a GUI. We detail each part in turn.

3.1 *In Vivo* Tricasting

In this *in vivo* demonstration, we would present tricasting over 2 WiFi and 1 WiMAX interfaces on the OpenRoads deployment in Stanford Gates building. This network is being used as the production wireless network for many of us[1]. We will use some of the 30 WiFi APs and 2 WiMAX basestations in our network to run this demonstration. Packets will be routed using OpenFlow-enabled Gigabit Ethernet switches from NEC and HP. The deployment on level 3 is illustrated in Figure 1.

By streaming a video from a video server to the tricasting client over lossy links, we allow the audience to judge the amount of loss in the flow through the quality of the video. By receiving data from multiple interfaces concurrently, an increase in quality of the video is perceived. Through the multiple transmissions received, loss in the video stream can be recovered (as in repetition coding). This corresponds to the increased video quality perceived.

Further, we will also be tricasting over a WiMAX interface on the laptop, demonstrating the integration of multiple wireless technologies within a single platform. Here, the client could arbitrarily switch from one interface to another regardless of the radio technology used. It is important that our platform can encompass many wireless technologies to accomplish our vision of a "flattened" network.

Further, by holding to multiple connections simultaneously, we will show how this mobility manager coded in 227 of C/C++ can achieved seamless make-before-break handover over multiple wireless technologies. We believe this presents a glimpse of our envisioned future wireless networks, and hope it will inspire others.

[1]The text of this paper has been transmitted over our OpenRoads network many times throughout its writing.

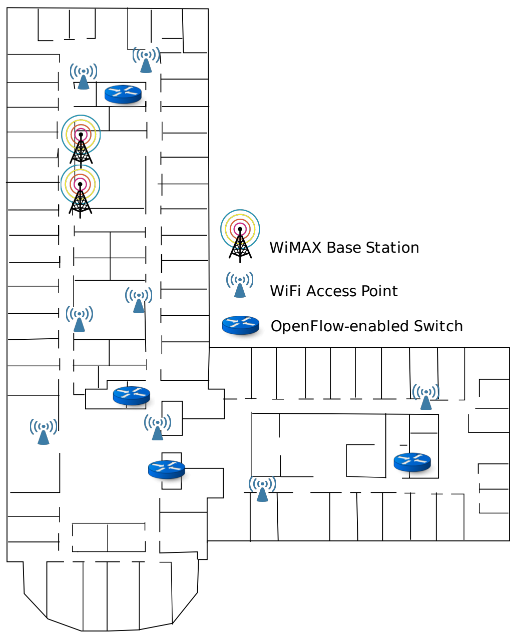

Figure 1: Stanford's OpenRoads deployment on level 3 of Gates

This video quality of this demonstration will be shown in a video, which is already available online[2]. Further, we have developed a GUI to visualize how packets in the video stream transverse our network. This GUI will be connected remotely from the conference site to our campus deployment via Internet. Through this visualization, the audience can see how the flows are redirected in our network during the tricasting, drawing correlation between network events and user experience.

3.2 *In Vitro* Bicasting

In order to let the conference attendees experience n-casting (and therefore the flexibility and capability of OpenRoads) firsthand, we will also present an *in vitro* demonstration at the conference venue. Using a small local setup, as shown in Figure 2, we will showcase bicasting.

Beyond showing the video stream and the network visualization (as in the *in vivo* demonstration), we have also developed a GUI for the conference attendees to control the bicasting. The GUI will allow the audience to manually associate and dissociate each interface on the client to one of the three WiFi APs. The video streaming will be continually shown, allowing the audience to perceive the end-user experience. Through this firsthand experience, we hope to show how easily OpenRoads allows flow to be redirected in the network.

4. LOGISTICS

[2]Watch a video of our demonstration at `http://masayoshi.smugmug.com/gallery/9113061_zhkxK/1/#607385725_4RxFC-A`.

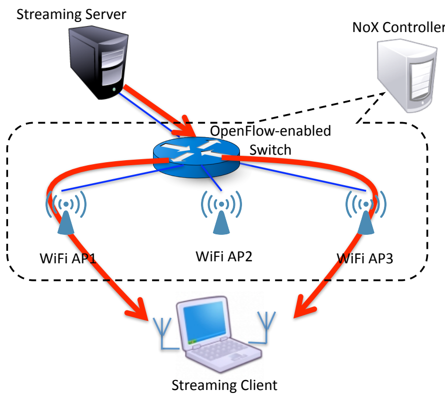

Figure 2: Bicasting over WiFi on-site

The *in vivo* demonstration will involve numerous equipments in the our OpenRoads deployment in Stanford. The following items will be required for our *in vitro* demonstration:

- 4 Laptops — working as the video server, client, network controller and network visualization display for the demonstration. The video server will also act as a display for the tricasting video.
- 3/4 WiFi APs and a software OpenFlow Ethernet switch (using Soekris Net5501) – to form our local network setup.

We will ship these equipments to the conference venue. Further, we will require a stable connection to the Internet (with guaranteed bandwidth of around 500 kbps) that supports SSH tunneling. Power outlets for the 8 devices would also be required, on top of a space of about 180 *cm* by 80 *cm*. A poster stand would also be appreciated. If possible, a large screen for showing the network visualization will be ideal. The setup is estimated to take around 1 to 2 hours.

This demonstration is led by Kok-Kiong Yap and Te-Yuan Huang, both of whom are students in Stanford University.

5. REFERENCES

[1] Kok-Kiong Yap, Masayoshi Kobayashi, Rob Sherwood, Nikhil Handigol, Te-Yuan Huang, Michael Chan, and Nick McKeown. OpenRoads: Empowering research in mobile networks. In *Proceedings of ACM SIGCOMM (Poster)*, Barcelona, Spain, August 2009.
[2] NOX: An OpenFlow Controller. `http://noxrepo.org/wp/`.
[3] The Programmable Open Mobile Internet (POMI) 2020 Project. `http://cleanslate.stanford.edu/research_project_pomi.php`.
[4] Rob Sherwood, Michael Chan,Glen Gibb, Nikhil Handigol,Te-Yuan Huang,Peyman Kazemian,Masayoshi Kobayashi, David Underhill, Kok-Kiong Yap, and Nick McKeown. Carving research slices out of your production networks with OpenFlow. In *Proceedings of ACM SIGCOMM (Demo)*, Barcelona, Spain, August 2009.