Experimental Demonstration of OpenFlow Control of Packet and Circuit Switches

Vinesh Gudla, Saurav Das, Anujit Shastri, Guru Parulkar, Nick McKeown and Leonid Kazovsky
Department of Electrical Engineering, Stanford University, California 94305, USA
vineshg@stanford.edu
Shinji Yamashita
Fujitsu Laboratories Ltd., 10-1, Morinosato-Wakamiya, Atsugi 243-0197, Japan

Abstract: OpenFlow is presented as a unified control plane and architecture for packet and circuit switched networks. We demonstrate a simple proof-of-concept testbed, where a bidirectional wavelength circuit is dynamically created to transport a TCP flow.

OCIS Codes: 060.4253 Networks, circuit-switched, 060.4259 Networks, packet-switched

1. Introduction

The unification of packet and circuit switched network operation, control and management offers significant advantages in the core of the network, in terms of cost efficiency, energy efficiency and network performance. We are not the first to propose such a goal and neither are we the first to highlight its benefits. However, while other proposals have been made over the last decade, none have gained significant traction in the industry. We believe there are two main reasons for this. First, IP and transport networks are two very different networks with different architectures, switching technologies, and control /management mechanisms. The former is automated, dynamic, lightly managed but ultimately best effort, while the latter is largely manual, semi-static, tightly managed and very reliable. Furthermore, while the control plane and data plane are tightly coupled in packet networks, transport networks have traditionally maintained a separation between data and control, albeit at the cost of losing automated operation. In the face of such differences, unification of the two networks is extremely challenging. Second, all previous proposals for unified control, have assumed that the two networks described above remain largely *unaltered*, making the protocols that go across them, overly complex, fragile and ultimately unusable for unification.

OpenFlow [1][2] has been recently proposed as a unified control plane (UCP) and architecture for packet and circuit networks [3]. OpenFlow advocates the separation of data and control planes for circuit *and* packet networks, as well as the treatment of packets as part of *flows*, where a packet flow is defined as any combination of L2/L3/L4 headers. This, together with L1/L0 circuit flows, provides a simple flow abstraction that fits well with both types of networks (Fig. 1). OpenFlow thereby presents a common platform for the control of the *underlying* switching hardware, that switch flows of different granularity, while allowing all of the routing, control and management to be defined in software outside the datapath, in the OpenFlow controller (Fig. 2). Our larger motivations, ideas and goals have been presented in [3]. In this paper we wish to report on a simple proof-of-concept demonstration of OpenFlow unified control of packet and circuit switches.

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

In/Out Port	In/ Out Lambda	VCG	Starting Time-Slot	Signal Type

Fig.1 Unifying abstraction for packet and circuit switches

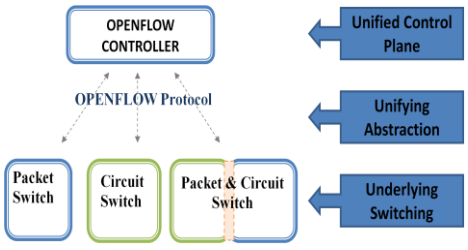

Fig.2 OpenFlow Unified Architecture

2. Experimental setup

We demonstrate OpenFlow unified control in our testbed (Figs 3a & 3b) with the help of OpenFlow enabled packet switches based on the NetFPGA platform, and an OpenFlow enabled circuit switch based on a Wavelength Selective Switch (WSS).

The NetFPGA[4] is a programmable hardware platform that allows researchers to implement electronic packet switching functionality in hardware. It is a PCI card that consists of a Xilinx FPGA and 4 GE ports, and can be easily installed in any PCI slot in a host computer (Fig. 3c). The FPGA can be programmed to behave as a simple Ethernet switch or an IP router that switches packets between the 4 GE ports. But more importantly, it allows building *new* packet processing functionality in hardware, so networking researchers can try out their ideas and not

be limited to functionality provided in commercial packet switching equipment. In our testbed, an OpenFlow packet switch is implemented in the NetFPGA, which switches flows defined on a subset of the 10-tuple shown in Fig. 1. The flow definition is inserted in the switch's hardware flow table, by the controller via the OpenFlow protocol [2].

The WSS is an all-optical lambda switch in a 1X9 configuration. It has the ability to independently switch any of 40 incoming wavelengths at the single input port, to any of the 9 output ports (Fig 3d). The incoming wavelengths (100 GHz spaced ITU DWDM lambdas) are demultiplexed and directed to their respective MEMS mirrors, which can be rotated to direct their wavelength to any of the 9 output ports, where they are multiplexed back into the outgoing fiber. In our setup we used the WSS bi-directionally, such that one of the 9 output ports actually served as an input port for a certain wavelength ($\lambda 2$). The mirrors are controlled with a voltage driver which is sent commands over RS232 from a PC. In turn the PC interacts with the OpenFlow controller which directs the provisioning of cross-connections via the circuit switching features of the OpenFlow protocol [2]. Together the PC, driver and the WSS can be regarded as an OpenFlow enabled circuit switch.

One of the four GE ports on each packet switch was connected to an electrical-to-optical converter from TrendNet (GE to SFP)[5]. We used Fujitsu DWDM SFP 2.5 Gbps transceiver modules in the converter which transmitted wavelengths of 1553.3 nm and 1554.1 nm respectively. These wavelengths traveling in opposite directions were multiplexed at the output/input of the first packet switch (NF1) by an AWG, transported through 25 kms of SM fiber, and then demultiplexed by the WSS before the input/output of the second packet switch (NF2). Together the OpenFlow packet and circuit switches form the underlying switching hardware that switch at different granularities (packet and lambda) and are controlled by an external unified control plane (compare Figs. 2 and 3b).

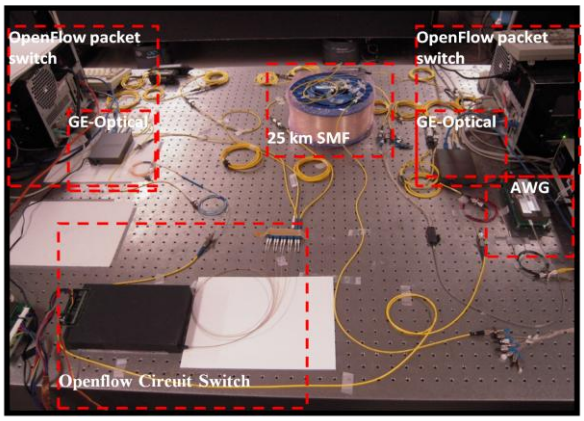

Fig. **3(a) Testbed and components**

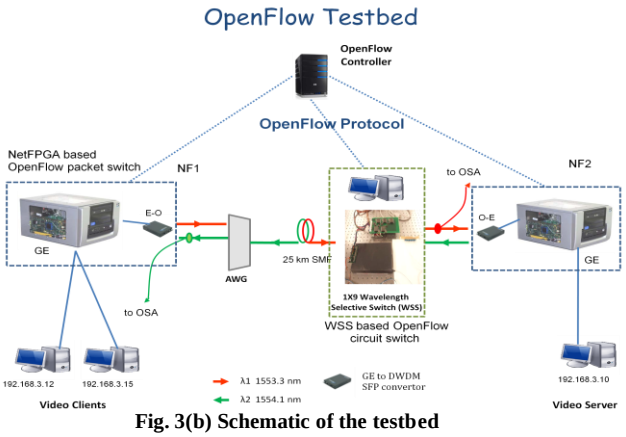

Fig. 3(b) Schematic of the testbed

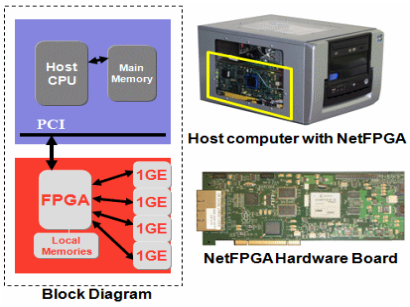

Fig. 3(c) NetFPGA platform

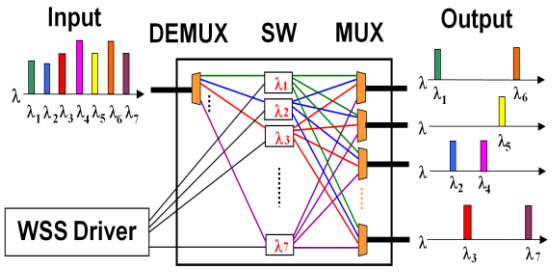

Fig. 3(d) Schematic of Wavelength Selective Switch

Procedure: We connected client and video server PCs (end-hosts) to the packet switches via GE, and used Helix DNA as the video streaming server and Real Player as the client application. A video request is made by the client application to the client PC's TCP/IP stack (in the host OS), by providing the server's IP address, the L4 port it is listening on, as well as the video filename. The client OS initially does not have knowledge of the MAC address corresponding to the server's IP. It therefore generates an ARP request which is received by NF1 and is processed as the first packet of a new flow. Since the packet doesn't match any flow entries in NF1, it gets forwarded to the OpenFlow controller. The controller makes the decision to create the circuit by using the OpenFlow protocol to make the bidirectional wavelength cross-connection in the WSS, thereby bringing up the Ethernet link between NF1 and NF2. It then inserts flow table entries in the latter, to broadcast the ARP request to all interfaces other than the one which received the packet. This results in the ARP request reaching the server PC via the WSS and NF2. The

server PC's OS sends the ARP reply, following which TCP handshaking takes place and a L4 connection is created, over which the video request is made. Finally, the server streams the video data packets over UDP, which are transported over the same bidirectional circuit created by the controller, and are received and displayed by the client.

3. Measurements and Discussion

We measured the time taken by various steps of the process and listed them in Table 1 for different configurations. The measurements were made using Wireshark, a packet analyzing software that monitors network traffic and displays OS time-stamped packets received over an interface. It was run on NF1, NF2 and the WSS driver PC.

The WSS-connect-command-received is the time elapsed between the ARP request from the client recorded on NF1, and the connect command from the OpenFlow controller received by the WSS driver. We recorded this to take only 1ms. Once the connect command is received, the driver commands the WSS to make the connection (over RS-232). The driver PC was programmed to transmit an echo request message to the controller once it receives confirmation over RS-232 that the optical connection has been set up. The time-stamp for this echo message was also recorded in Wireshark, and delay for this message with respect to the connect-command-received message was measured to be 1.3 secs. This number appears large, especially given that the switching time for the MEMS mirrors themselves is on the order of a few ms. The rest of the time is actually taken by the RS-232 communication between the driver PC and the built-in WSS software, which is not optimized for fast setup of connections- for example, while the mirrors can be rotated simultaneously, the cross-connect commands are sent one at a time, and the second command cannot be sent until detailed feedback is read off from the serial port for the first command. We believe that optimizing this interface can reduce this number to tens of ms, if not lower.

Table 1: Time taken for connection to be set up

Steps	GE-Optical-GE link with WSS initially not provisioned	GE-Optical-GE link with WSS cross-connected	GE-GE link (no optics)
WSS connect command received	1 ms	-	-
Cross-connection confirmation	1.3 s	-	-
TrendNet GE-SFP module link-up	4.74 s	4.10 s	-
NetFPGA GE-GE link-up	1 ms	1 ms	1 ms

The next row corresponds to the time it takes for the Ethernet link to come up after the optical connection has been made. This time was measured by running Wireshark on NF2 (connected to the server) and noting when the initial ARP request (forwarded from NF1) makes it through the newly brought-up Ethernet link. From this time we subtracted the time it took for the cross-connection confirmation to arrive, thereby leaving only the time for Ethernet to negotiate a link-up. This was measured at 4.74 secs. Again, this number appears unreasonably large. However there are two components in this delay. One of them is the time it takes for the TrendNet GE-SFP converter boxes to recognize a link-up, and the second is the time it takes for the GE switchports on NF1 and NF2 to recognize/negotiate the link-up. The latter time was measured separately with a pure Ethernet connection (shown in the last column) by sending pings from the client and physically disconnecting and then re-connecting the Ethernet cable from the switchport. As expected the link-up time was near instantaneous. However when the same experiment was performed with the disconnection/re-connection at the convertor box (either SFP or GE side) the large delay was measured again (middle column), clearly indicating an issue with the internal mechanisms of the converter box which has most likely not been optimized for rapid link-up. We believe that a dedicated ASIC/FPGA optimized for fast setup will help in this regard.

4. Conclusions

We demonstrated dynamic control of packet and circuit switches with a unified control plane with the OpenFlow architecture. We measured the latency in creating the circuit and believe that with optimization, the link-up time can be reduced to less than 1sec (as explained above). Our future work will involve expanding the testbed to incorporate other switching types like TDM and fiber switches, and to demonstrate network applications that benefit from a UCP across diverse switching technologies. We also aim to demonstrate unified virtualization [3] of the data plane.

5. References

[1] N. McKeown, et. al., "OpenFlow: Enabling Innovation in Campus Networks", SIGCOMM CCR, Vol. 38, Issue 2, March 2008.
[2] OpenFlow website: http://www.openflowswitch.org
[3] S. Das, G. Parulkar, N. McKeown, "Unifying Packet and Circuit Networks", Below IP Networking Workshop held in conjunction with Globecom, November 2009
[4] NetFPGA website: http://www.netfpga.org
[5] http://www.trendnet.com/products/proddetail.asp?prod=110_TFC-1000MGB&cat=22