OpenFlow Managed Home Gateways

George Cockshull

September 18, 2013

1 Introduction

An attempt will be made to install OpenFlow on various home gateway devices, in order to validate the accessibility of the technology and evaluate the potential for such devices to form the edge of an OpenFlow ISP infrastructure.

2 Hardware Requirements

A build of OpenWrt with OpenFlow on x86 with a full set of qdiscs, tc, iperf and other utilities comes to a 7.25MB squashfs image or 52.5MB ext4 image. Routers with 8MB of flash or more will therefore be enough to verify these tests. A micro version of OpenWrt is available and can be built with squashfs with a minimal set of packages which can come under 2MB (1.7MB.) suitable for reduced functional implementations on space-constrained hardware e.g. low-end home gateways.

As for RAM requirements, OpenFlow implementations 1.1 and later offer multiple flow tables with great flexibility and so flow entries can scale with RAM, even allowing different fields to be handled by different storage means (TCAM is expensive but fast, RAM-based tables are slow but cheap.) This means the performance of an OpenFlow switch is dependent on RAM, but a residential application of OpenFlow need not have particularly large amounts of RAM and the table space can be greatly reduced if using proactive rules. It is only when one cares about performance that the introduction of TCAM (Ternary Content Addressable Memory) and larger table sizes becomes necessary in a small home gateway setup. More information about the scalability of OpenFlow can be found on SDN Central [8].

We will attempt to compare the performance in the following hardware experiments.

3 PC Engines Alix 2d

- 3x VT6105M 10/100M Ethernet
- AMD Geode LXD800 32bit @ 500MHz

- 256MB DDR DRAM

OpenFlow versions 1.0 and 1.3 have been tested to work on this board. This was done with the Stanford reference 1.0 switch *Pantou* [6] `OFsoftswitch13` package[1] and OpenWrt (Barrier Breaker r37384.) OFsoftswitch13 is a user-space implementation by CPqD based off OF11softswitch, based off the Stanford reference implementation. This means the implementation has greater system call overheads than a kernel-space equivalent like Open vSwitch.

As of July 2013, the switch support for later versions of OpenFlow is currently ahead of the support for common controllers; POX (Python version of NOX) only supports OF1.0 as of *carp*[7]. A modified NOX Zaku Controller is available that offers (currently) work-in-progress OF1.3 Support[3].

Measurement	Switch	Value / Mbit/s ± 0.5 Mbit/s	CPU load ± 1%
Theoretical link rate	n/a	100	n/a
LAN link b/w	(k) Linux bridge	91	64%
LAN switch b/w	(k) Linux bridge	88	55%
LAN switch b/w	(u) Pantou Ref (1.0)	74	100%
LAN switch b/w	(u) OFsoftswitch (1.3)	85	100%
LAN switch b/w	(k) Open VSwitch (1.0)	17	100%

Figure 1: Measurements for PC Engines Alix 2d. (k) = uses/is a kernel module, (u) = runs entirely in user space

Fig. 1 Describes the outcome of the measurements taken on this device, testing the various switch implementations. Where OF1.0 is tested, the POX controller is used in a L2 learning switch mode, and where OF1.3 is tested, the NOX Zaku OFlib 1.3 Controller is used in a similar learning switch mode. Each test is performed using `iperf` over 30 seconds to minimise the impact the of the response times of the controller for the initial ARP and rule installation and average over any other transient delays; a emulating what how a user would experience download rates for non-real time applications. Note the surprising result of Open vSwitch performing at only 17Mbit/s compared to other implementations. This result was obtained running the openvswitch kernel module, with user daemons ovs-vswitchd and ovs-controller running locally on the Alix2d. This should intuitively perform at an equal, if not higher rate than a similar user-space setup. I have yet to find out why.

4 Linksys WRT54G v5

Version 5 of the Linksys WRT54G features and Broadcom SoC with 2MB Flash / 8MB RAM and comes pre-loaded with a VxWorks-based operating system using a proprietary bootloader, as opposed to the previous versions that come with Linux-based operating systems and 4MB Flash / 16MB RAM.

Consumer-grade wireless routers tend to have integrated 2MB - 16MB of Flash with 4MB being most commonly featured; the v5 can be considered a low-end consumer device.

The choice of OS and bootloader makes it very difficult to install a Linux distribution as it involves a flash of the bootloader and some effort to restore the MAC address of the device on the new image. While tools have been written to "kill" VxWorks, prepare a new bootloader and scrape the MAC address, the v5 model remains "unsupportable" by OpenWrt. [4] It is not recommended to install OpenWrt on this model as there is a good chance of bricking the device. This change of operating system doesn't introduce additional features, and when combined with the reduction in Flash storage and the firmware update web interface imposing restriction on 3rd party firmware images, it seems clear that for business reasons, the manufacturer intented to cripple the 3rd party support on their cheaper models.

It is worth noting that OpenWrt with OpenFlow would certainly be possible on the WRT54GL and WRT54GS ranges from Linksys, and the limitations described only relate to the newer models (v5 onwards) of the low end WRTs. In particular the Linksys *L models ('L' is for Linux) are intended for third-party firmware use and typically offer higher Flash storage.

5 NetFPGA 2.1 rev 3

- 4x 1Gbit Ethernet
- Virtex-II Pro 50 FPGA
- 4.5MB SRAM
- 64MB DDR2 DRAM

The 1G version of the NetFPGA platform by Stanford [2] was installed in a Dell desktop along with a 4-port 100Mb NIC for performing local loopback regression tests.

Measurement	Netfpga project	Controller	Value Mbit/s	Range Mbit/s
Theoretical link rate	n/a	n/a	1000	n/a
LAN switch b/w	reference_switch	n/a	937	± 0.5
LAN switch b/w	openflow_switch	Localhost POX	930	± 6.0
LAN switch b/w	openflow_switch	In-band POX	931	± 6.5

Figure 2: Measurements for NetFPGA

Each test is performed using `iperf` over 120 seconds where applicable. As expected, due to the long test period, any controller referrals are averaged out, as can be seen by the negligible difference between the localhost and in-band controller values. In this case, they are both in-band, but the localhost value

was with the controller host on the NetFPGA machine and the in-band value means the same controller setup but hosted on another machine (one of the hosts.)

One thing that was noticed with the NetFPGA experiments was regular spikes in the round trip times when pinging across the switch, peaking from 0.1ms to about 40ms. Upon closer inspection, these spikes happened in 30 second intervals, which correspond to the default hard-timeouts for the flows.

```
rtt min/avg/max/mdev = 0.052/1.000/51.756/5.387 ms
```

What makes this worrying is the fact that these flows should not be expiring in a continuous 1 second interval ping test. Indeed performing a flow dump reveals flows with an idle timeout of 10s and a hard timeout of 30s. This indicates a problem with either the NetFPGA openflow_switch implementation or the corresponding OpenFlow software driving it.

Hard-timeouts are a simple solution to prevent a reactive OF switch setup with limited hardware table space from saturating, which could result from a high turnover of connections. In a correct implementation, the penalty would be very rare and therefore negligible in cost due to its high amortisation. In this test however, 40ms is a rather large penalty to occur with such regularity. This penalty could however be further mitigated by a smarter controller implementation that dynamically expires table entries based on least recently used when approaching saturation. This would be made possible with the per-flow counters required at least as early as OpenFlow 1.0 [5].

6 Conclusion

The effect of TCAM on a hardware-accelerated switch (the NetFPGA implementation) on performance is difficult to compare to cheap software implementations due to the large (10x) disparity in line speed in the hardware I have tested. Despite this, both styles of solution fare similarly in terms of percentage of line speed. It should be considered that a hardware-accelerated solution has more performance headroom for flow lookup, by virtue of hardware TCAM. This being said, due to pipelined software implementations being what they are, this should be negligible - unnoticeable in casual use - in the case of a home setup, and only becomes important in high performance or high capacity setups.

References

[1] Fernandes, E.L. OpenFlow 1.0 Softswitch implementation (GitHub) `https://github.com/CPqD/ofsoftswitch13`

[2] NetFPGA 1G Specs `http://netfpga.org/1G_specs.html`

[3] NOX Zaku 1.3 OFlib Controller (GitHub) `https://github.com/CPqD/nox13oflib`

[4] OpenWrt Supported devices table, "Unsupportable" Linksys models http://wiki.openwrt.org/toh/start#linksys3

[5] OpenFlow 1.0.0 Specification. PDF available online [Accessed Sep 2013] https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

[6] Pantou: OpenFlow 1.0 for OpenWrt http://www.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT

[7] POX Controller (GitHub) https://github.com/noxrepo/pox

[8] SDN Central "OpenFlow Can It Scale?" http://www.sdncentral.com/technology/openflow-sdn/2013/06/