Open Source At The Top Of The Rack

John W. Linville

LinuxCon North America

22 August 2014

Who am I?

What is this?

Enterprise packet switching hardware continues to be an open source hold-out in the data center.

- Familiar, tired, old arguments...
- Traditionally under-powered...
- Powerful, new designs on the horizon...
- OCP Networking...
- New platform for networking applications?!?!

Wedge and FBOSS

BLOG OPEN SOURCE EVENTS VIDEOS PUBLICATIONS

Mobile Web Data Infra Culture

Engineering Blog

Yuval Bachar

Adam Simpkins

Introducing "Wedge" and "FBOSS," the next steps toward a disaggregated network

PUBLIÉ MERCREDI

INFRA

We're big believers in the value of disaggregation – of breaking down traditional data center technologies into their core components so we can build new systems that are more flexible, more scalable, and more efficient. This approach has guided Facebook from the beginning, as we've grown and expanded our infrastructure to connect more than 1.28 billion people around the world.

Over the last three years, we've been working within the Open Compute Project (OCP) to apply this principle to open designs for racks, servers, storage boxes, and motherboards. And last year, we kicked off a **new networking project within OCP**, with a goal of developing designs for OS-agnostic top-of-rack (TOR) switches. This was the first step toward disaggregating the network – separating hardware from software, so we can spur the development of more choices for each – and **our progress so far** has exceeded our expectations: Broadcom, Intel, Mellanox, and Accton have already contributed designs for open switches; Cumulus Networks and Big Switch Networks have made software contributions; and the development work and discussions in the **project group** have been highly productive.

Today we're pleased to unveil the next step: a new top-of-rack network switch, code-named "Wedge," and a new Linux-based operating system for that switch, code-named "FBOSS." These projects break down the hardware and software components of the network stack even further, to provide a new level of visibility, automation, and control in the operation of the network. By combining the hardware and software modules together in new ways, "Wedge" and "FBOSS" depart from current networking design paradigms to leverage our experience in operating hundreds of thousands of servers in our data centers. In other words, our goal with these projects was to make our network look, feel, and operate more like the OCP servers we've already deployed, both in terms of hardware and software.

The hardware: more powerful and more modular

One of the big changes we made in designing "Wedge" was to give the switch the same power and flexibility as a server. Traditional network switches often use fixed hardware configurations and non-standard control interfaces, limiting the capabilities of the device and complicating deployments. We chose to leverage our existing "Group Hug" architecture for modular microservers, which enables us to use a wide range of microservers from across the open hardware ecosystem. For our own deployment, we've started with a microserver that we're using elsewhere in our infrastructure. But the open form factor will allow us to use a range of processors, including products from Intel, AMD, or ARM.

Why does this matter? By using a real server module in the switch, we're able to bring switches into our distributed fleet management systems and provision them with our standard Linux-based operating environment. This enables us to deploy, monitor, and control these systems alongside our servers and storage — which in turn allows our engineers to focus more on bringing new capabilities to our network and less on managing the existing systems.

The diagram below highlights the modular approach we've taken to the hardware design. Unlike with traditional closed-hardware switches, with "Wedge" anyone can modify or replace any of the components in our design to better meet their needs. For example, you could use an ARM-based microserver rather than the Intel-based microserver we've selected. Or you could take the electronics and repackage them in a new enclosure, perhaps to solve a different set of problems outside the rack. We're excited to see where the community will take this design in the future.

The software: greater accessibility and better control

On the software side, we wanted "FBOSS" to help us move more quickly. It was designed to allow us to leverage the software libraries and systems we currently use for managing our server fleet, including initial turn-up and decommissioning, upgrades and downgrades, and draining and undraining.

Hardware Designs

Most network switches are essentially embedded systems...

- Minimal CPU...
- Limited RAM...
- Tiny storage...

Software Architecture

Just enough to bring-up the hardware...

- Proprietized OS...
  - Limited functionality RTOS
  - Linux w/ minimal userland
- Static system management capabilities...
- Proprietary upgrade mechanisms...
- Hardware managed solely by application!

Closed Source

Almost every network switch runs closed source software...

- Un-hackable, appliance-like devices...
- Special skills needed even if device is open...
- Hardware documentation is closed to the public...
- Outside contributions are deemed unlikely...
- OpenWRT is an exception??

Better Processors

Devices are appearing with modern CPUs...

- Mainstream architectures (e.g. x86_64)...
- Server-class CPU performance...
- Advanced features (e.g. virtualization)...
- Modern (i.e. featureful) OSes too?

Open Designs

Open Compute Project is driving development of open designs...

- Support from multiple vendors (Broadcom, Mellanox, Intel, Accton, etc)...
- Community participation via the OCP Networking initiative...
- Backing from Facebook give the effort some clout!

Same Old Software

Despite the efforts to change the hardware, the software stack still looks very familiar...

- Linux is used, but still in a limited fashion...
- Switch hardware managed by userland-based "SDK"...
- User interface provided by proprietary management interfaces...
- Ther are a few exceptions (e.g. Cumulus Networks)...

Off The Shelf OSes

Moving towards a standardized hardware platform enables the use of standard OSes...

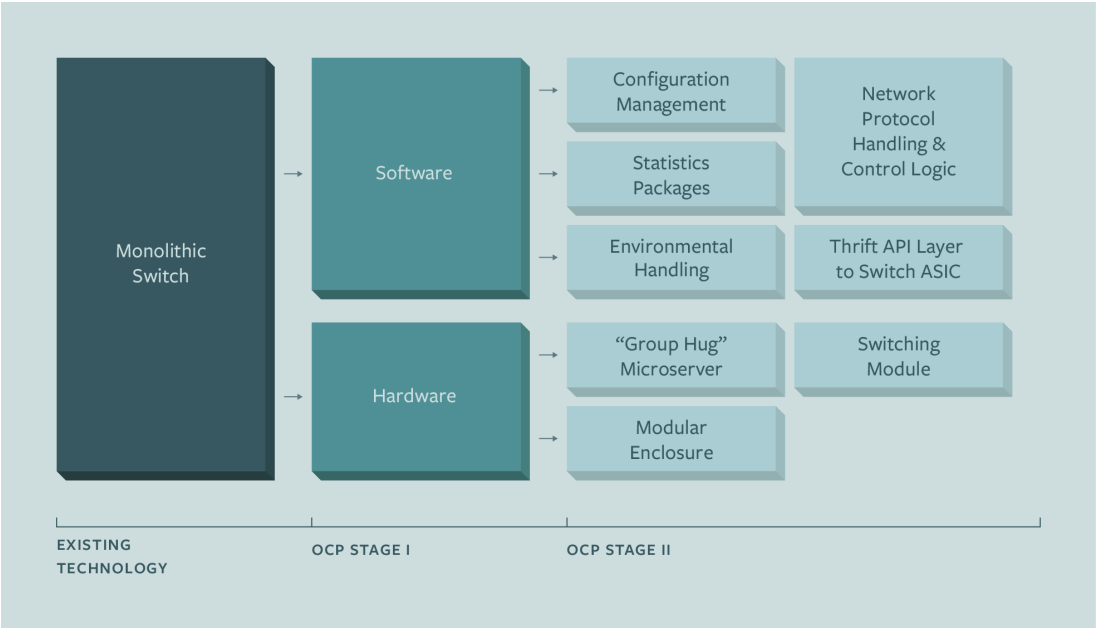
- Less need for specialized training...
- Easier deployment and maintenance...
- More frequent security updates and bug fixes...
- Code is widely used and better tested...

Better Management

Running a server-class OS lets you manage your switch more like a server...

- Incremental updates (by package)...
- Standard tools (Ansible, Chef, Puppet, Thrift, etc)...
- Developer-friendly environment enables custom solutions...

OCP Vision



Original image:
`http://gigaom.com/2014/06/18/facebook-has-built-its-own-switch-and-it-looks-a-lot-like-a-server/`

New Applications

What would you do with an Enterprise switch and a server CPU?

- Applications have "no hop" access to tons of bandwidth...
- Forward deployment of network-attached infrastructure...
- Seems like a good place for an SDN controller?

Hardware Diversity

What does a switch really look like?

- How do you talk to the registers?
- How do you move data?
- Inside-out or outside-in?

It is very easy to build a switch device model that excludes lots of existing switches!

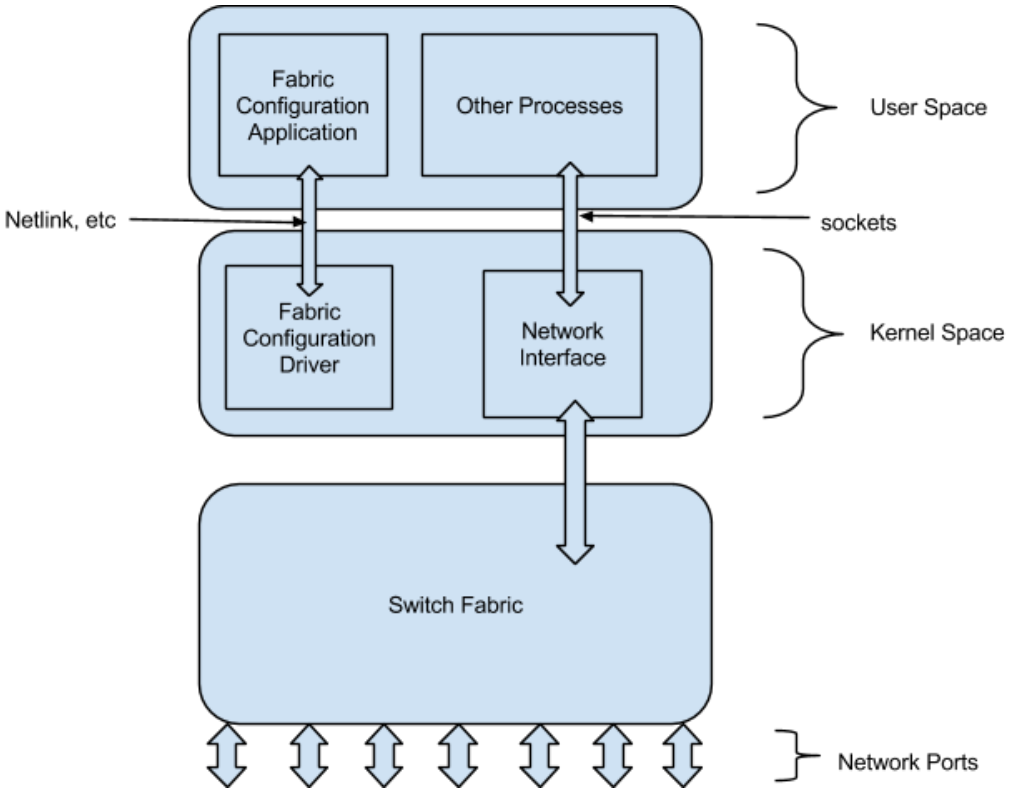
Software Modeling

There are several viable models for modeling switch devices in the kernel...

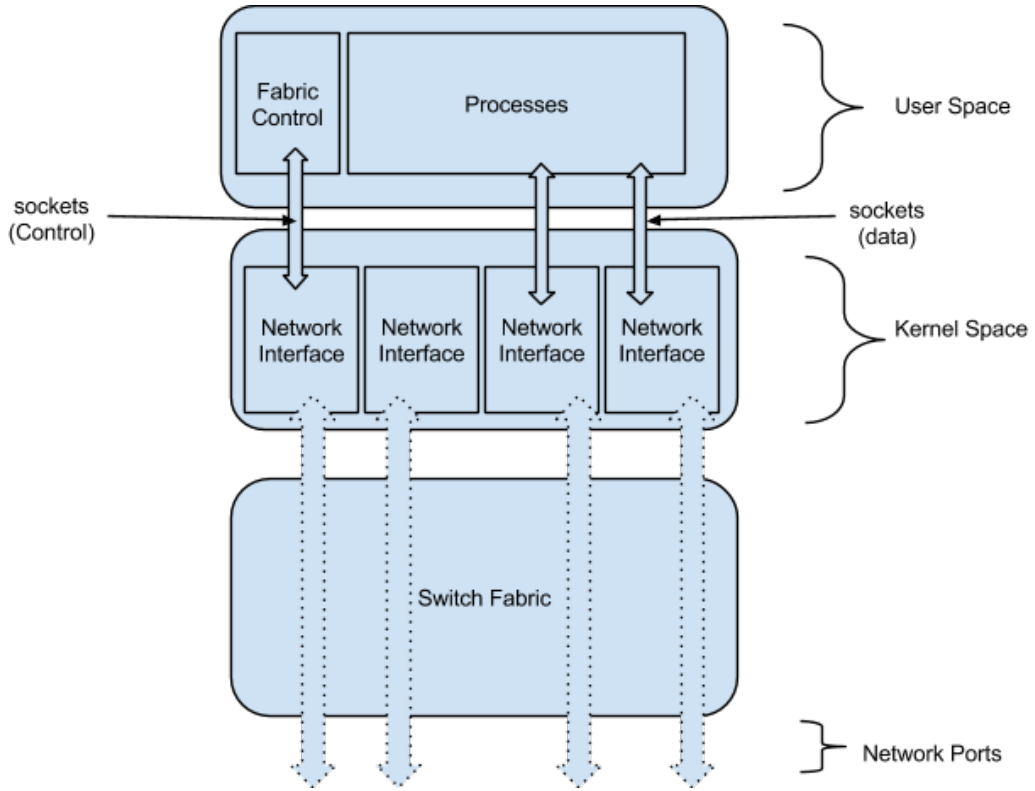
- Single-Port View
- Multi-Port View
- Multi-Port View w/ Dataplane Visibility

What seems to be the obvious choice may depend on your perspective...

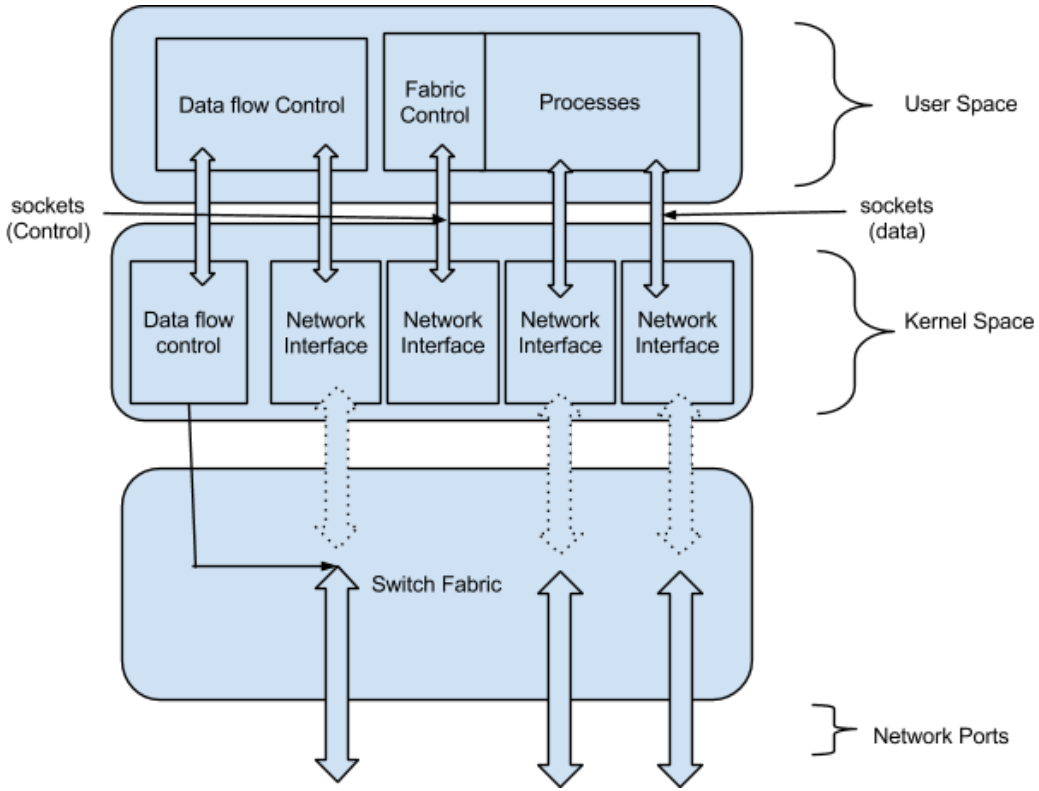
Single-Port View



Multi-Port View



Multi-Port View w/ Dataplane Visibility



Enterprise Vs. Flows

This could be titled "Legacy Vs. SDN"...?

- Is the CPU plugged into the switch?
- Or is the switch like a massive multi-port NIC?
- How much data do we need to be able to see?

Is Open vSwitch the best model?

Start Small

A journey of a thousand miles...

- Millions of switches are in the hands of consumers today...
- Most of these are handled by out-of-tree code in OpenWRT...
- Some infrastructure (i.e. DSA) exists in the kernel already...
- Genericize DSA and start porting drivers...

Be Simple

Enterprise-class switches are full-featured, complicated devices that do not map well to existing kernel concepts...

How do we get this merged?

- Implement a simple, NIC-like driver...
- Enable existing off-loads as appropriate...
- Introduce new off-loads and other features at L2...
- ...and beyond?

Let's Pretend

It's difficult to build a device model, without access to the devices...

- Existing hardware vendors are happy with the way things are...
- Status quo – impossible to demonstrate superior alternatives...
- Rocker Switch is a virtual model of a representative switch...
- User Rocker Switch as a platform for prototyping!

Cracking A Nut

Incumbent vendors own the status quo...

- Are there smaller hardware vendors on the edge of the market?
- Are some of them hungry enough to take a bigger risk on open source?
- Is this enough to crack the market?

Wrap-Up

Around the world and back again...

- Network switches are still closed, not without reason...
- New hardware designs are coming, better support for open source software is possible...
- Open source could bring innovations in the network...
- But what would we do with it?
- And how do we make it happen?

Put your money where your mouth is!

Questions?

Contact

Feel free to contact me!

- Email linville@tuxdriver.com
  - ...@redhat.com
  - ...@gmail.com
  - ...@kernel.org
- IRC linville on FreeNode
- Facebook as “John W. Linville”

Links

OpenWRT

`http://www.openwrt.org/`

OCP Networking Specs And Designs

`http://www.opencompute.org/wiki/Networking/SpecsAndDesigns`

Cumulus Networks

`http://cumulusnetworks.com/`

Centec Networks

`http://www.centecnetworks.com/`