CUMULUS

Building a Better NOS with Linux and switchdev

netdev 0x12 - July 12, 2018

David Ahern | Shrijeet Mukherjee

Agenda

- What is this Whitebox / Disaggregation / Open Networking {r}evolution?
- Evolution of Network Operating Systems
  - Increasing use of Linux
- The Next Step
  - Linux as the OS of the data center with ASIC drivers in the kernel

Legacy Networking

Whitebox Switches and Disaggregation

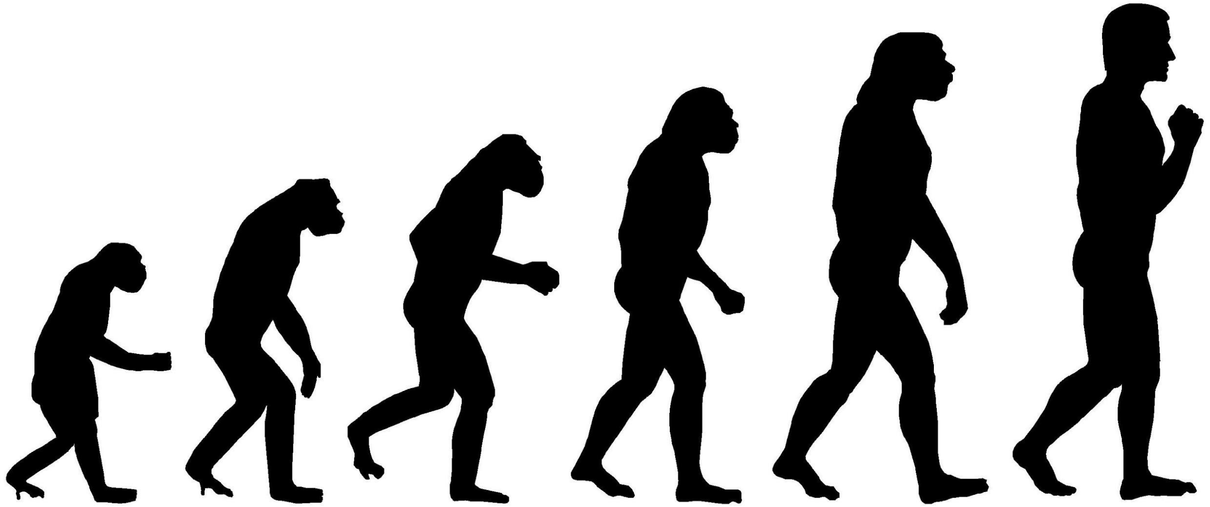
Open Networking

- Networking Operating Systems still silos
  - Can pick a vendor, but still highly dependent on vendor for service and consulting
- What does “Open” mean?
  - Shell access?
  - Able to run Linux commands at a shell prompt?
  - Able to run networking programs in the control plane?

Open Networking with Linux

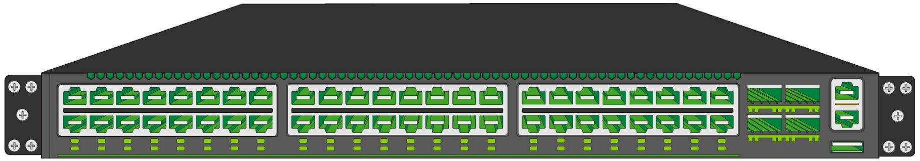
- Enable innovation
  - Solving network problems in ways unimagined by NOS or ASIC vendors
- Flexibility and reconfigurability are important – **essential** characteristics
  - Workloads, performance demands and characteristics can change quickly
  - Need a network that can adapt
- A big blob of highly interdependent processes is not flexible
  - No insights into each component means no flexibility
- Linux is about building blocks
  - Building blocks provide ultimate flexibility and reconfigurability

Networking Solutions Constantly Evolving



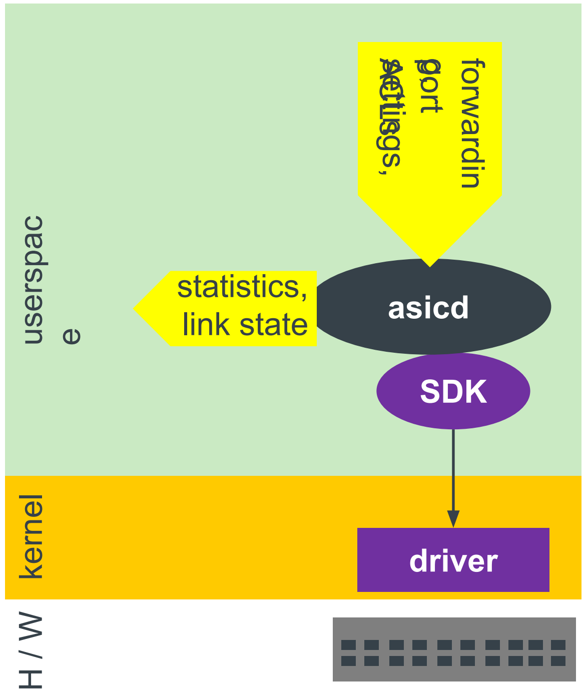
Switches Today Have Similarities to Servers / Hosts

- Hardware components
  - Commodity CPU (x86, arm, ppc)
  - Storage devices – USB, SSD
  - Management NIC
  - Multiple “data plane” ports
- Operational Model
  - Configuring interfaces and services
  - Monitoring – e.g., interface statistics, events, ...
- Servicing
  - Logging in remotely, diagnosing problems with familiar tools
- And, Linux is the primary OS

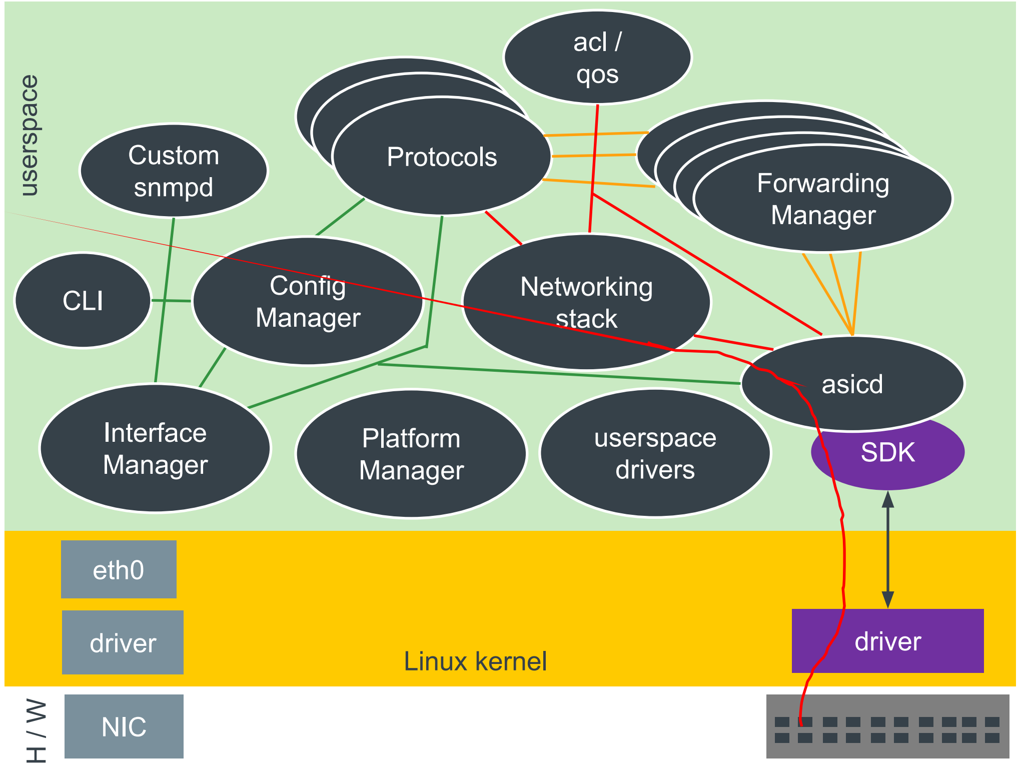


Switch ASICs and SDKs

- Commodity ASICs controlled via SDKs
  - All commands and queries to ASIC must be go through the SDK
- NOS has userspace ASIC driver

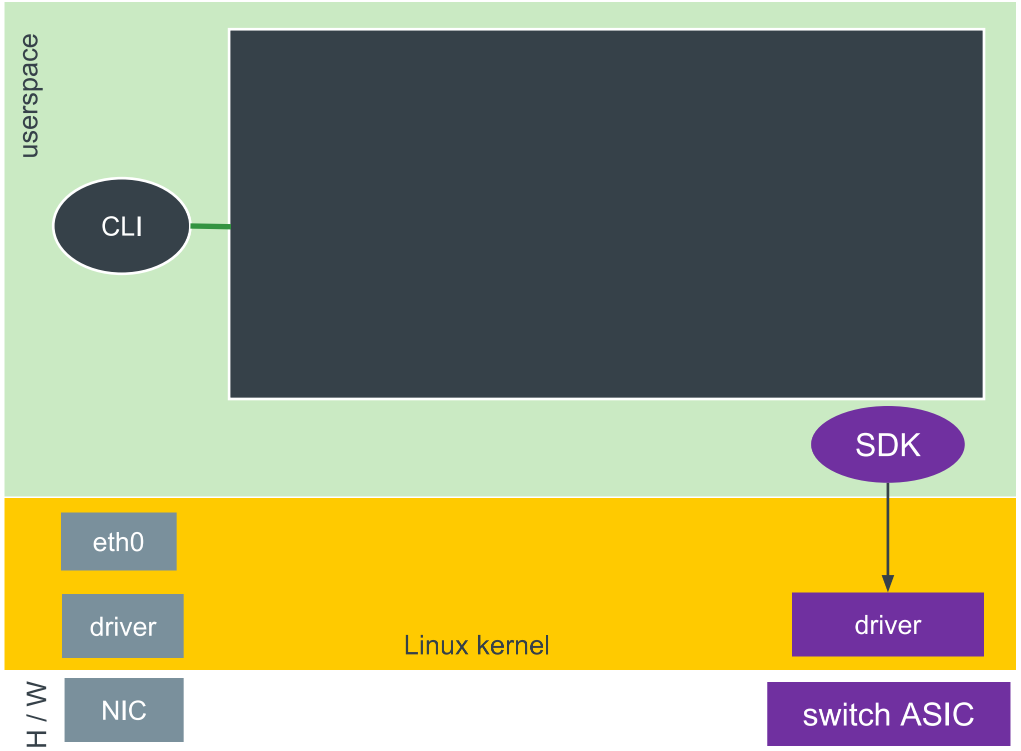


Early NOS Architectures



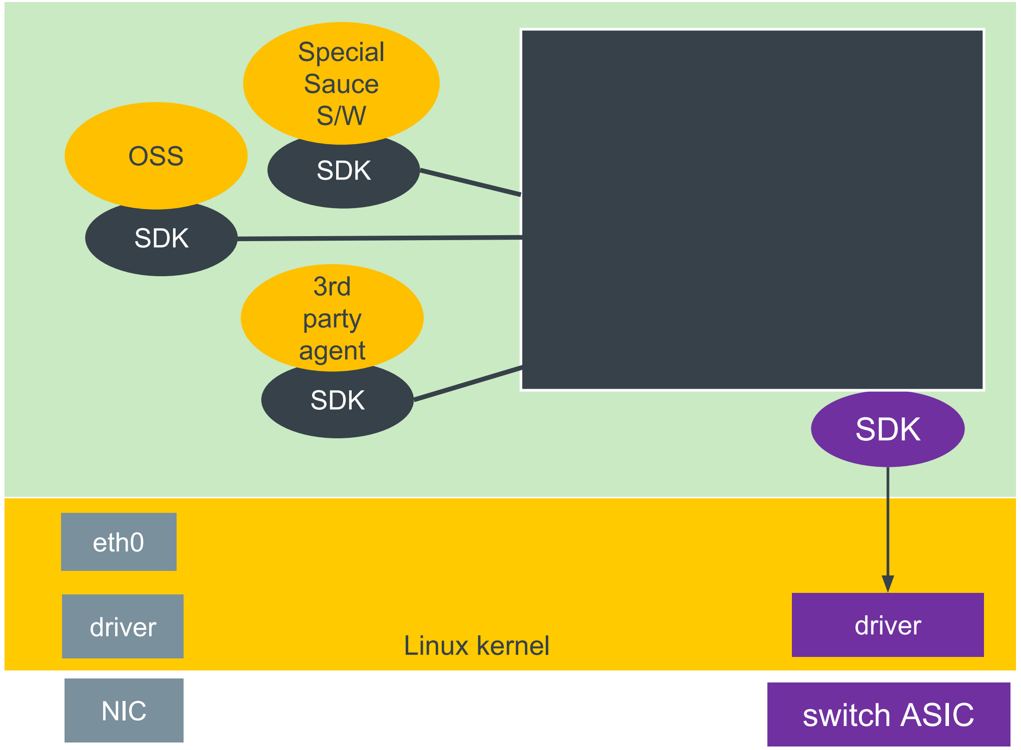
- All userspace, all custom software
  - Custom CLI to configure
  - Custom monitoring APIs
  - Custom diagnostic tools
- No data in the kernel
- Linux has a non-networking role
  - storage devices, process scheduling, management NIC, memory management, etc

The NOS is a Black Box



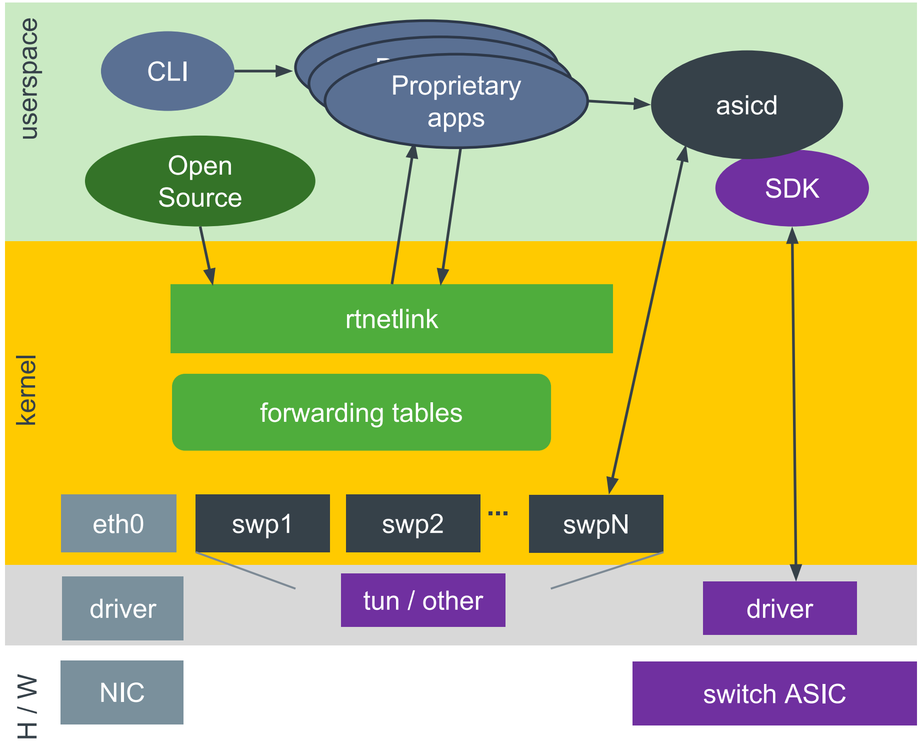
- Networking perspective it is a black box

Legacy NOS and 3rd Party S/W



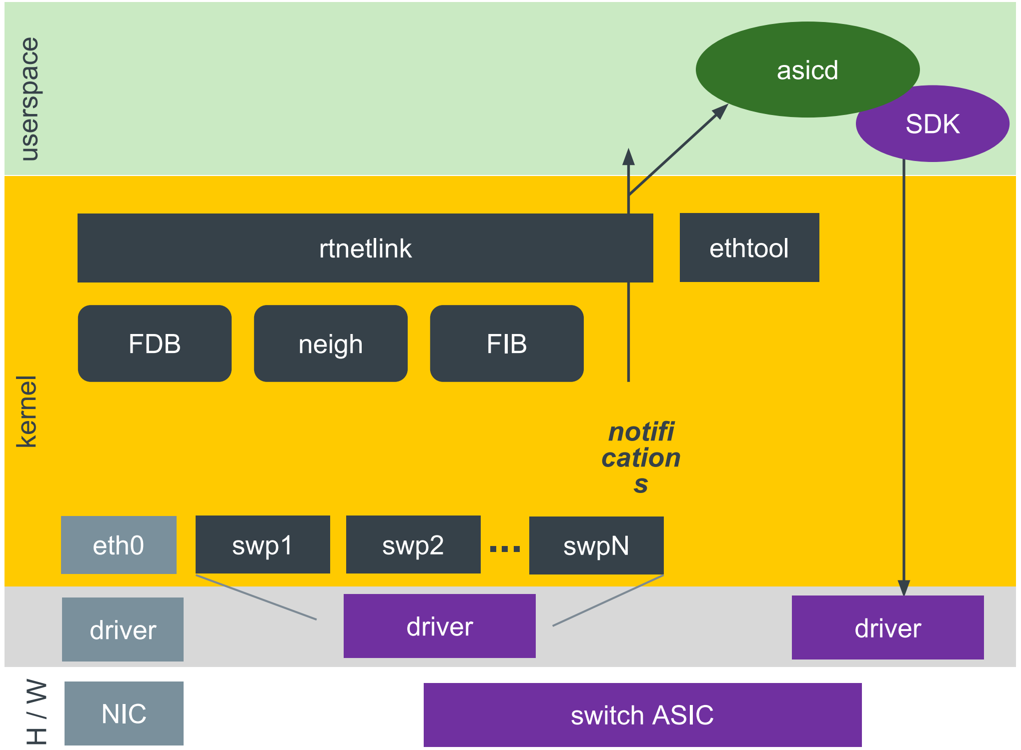
- Vendor Idea!
  - Ask customers to recompile / write their software against our SDK!
- Connect your software to the mother ship
- Seriously?

NOS With Some Linux Networking



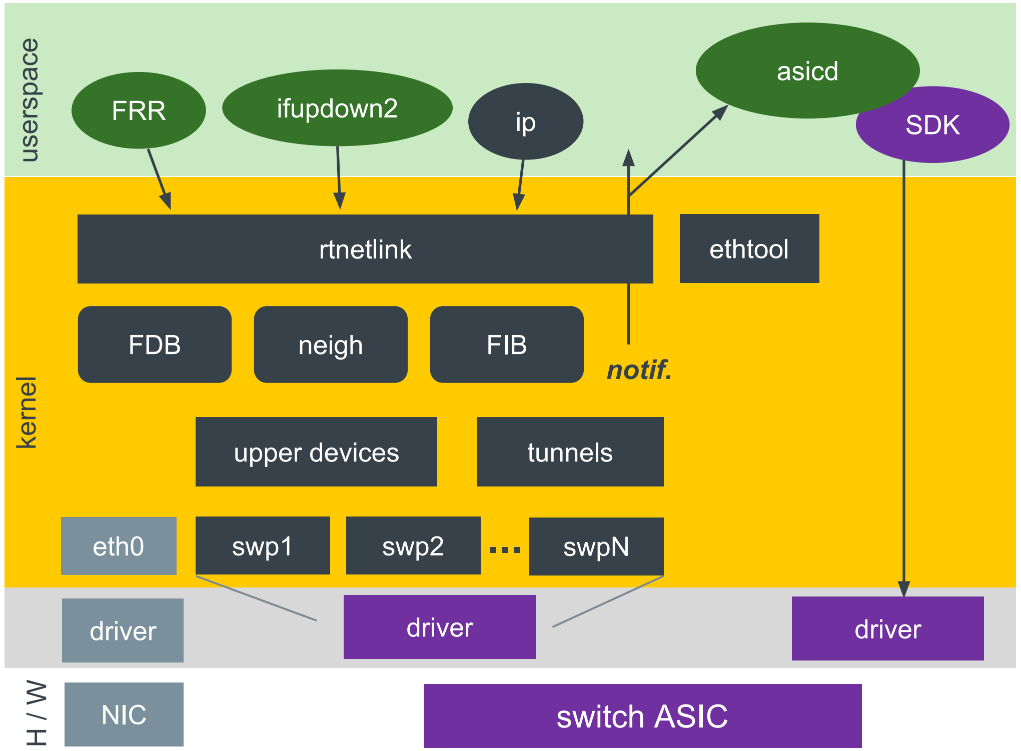
- create netdev's for ports
  - used to relay packets between SDK and Linux stack
- Pandora's box
  - How much data do you put into the kernel?
  - All of the routes? What about features - bridges, bonding, VLANs, vxlans, VRF?
  - Allow Linux APIs to configure networking, physical ports, or retrieve stats?
- Ad-hoc at best

Adding more Linux to the equation



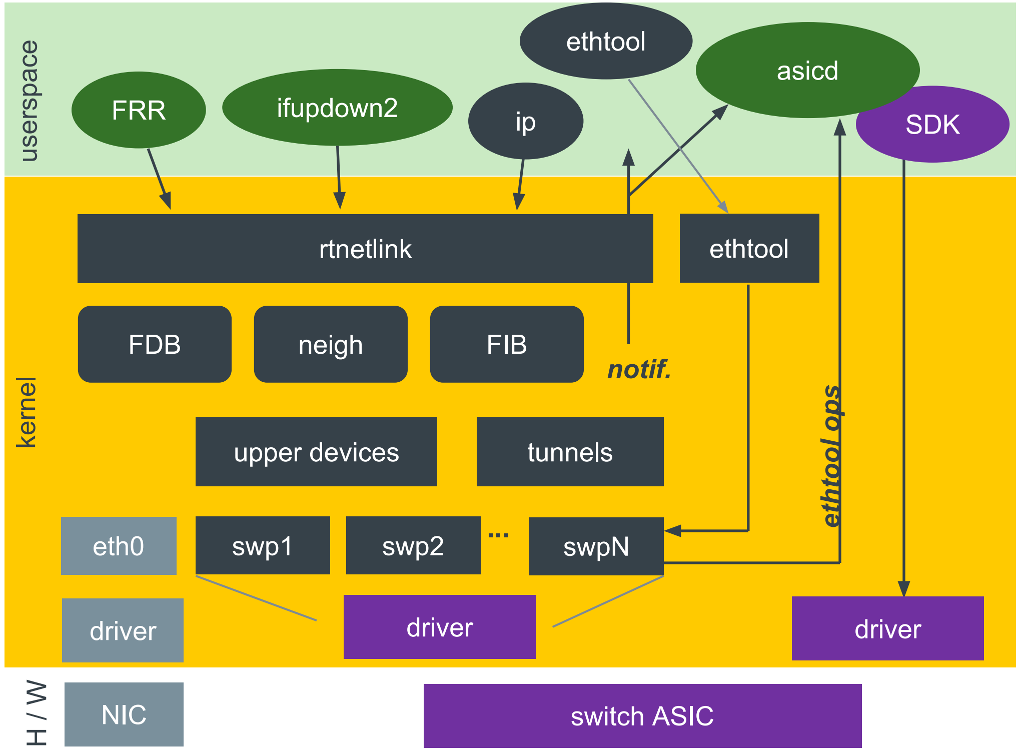
- Still reliant on userspace SDK
- Vendor drivers to create netdevs for front panel ports
  - in-kernel distribution of packets to port netdev
- Build from there with Linux APIs
  - rtnetlink and ethtool
  - notifications for changes to networking config and state

Adding more Linux to the equation



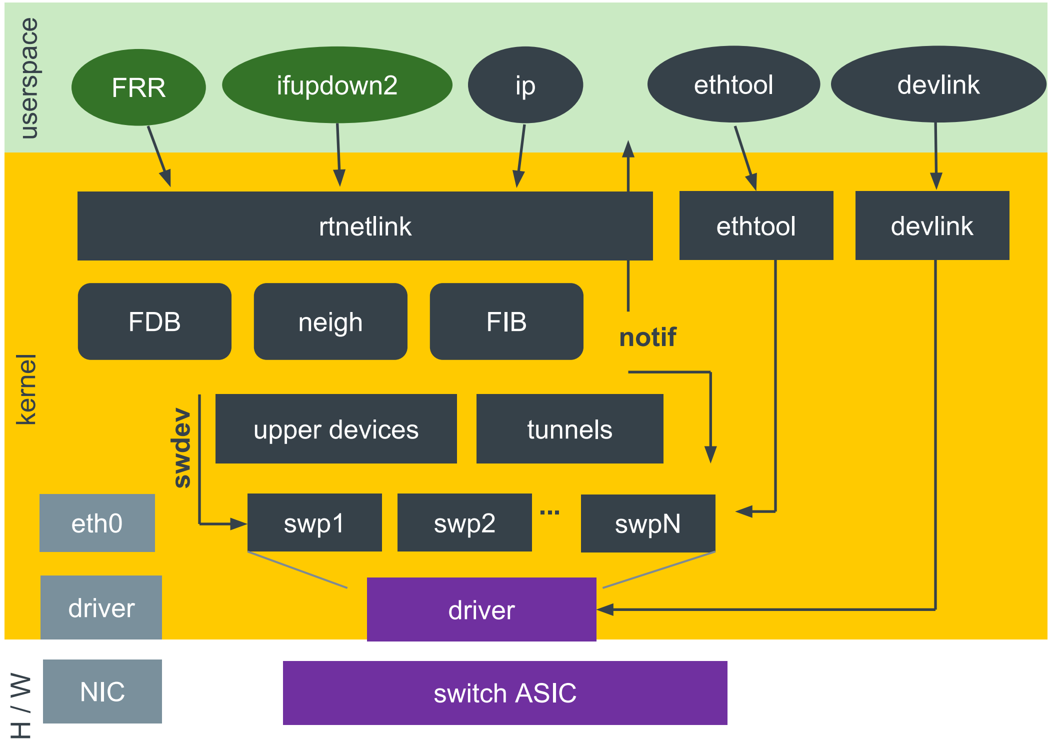
- Now we can use the Linux ecosystem
  - Interface managers to handle complicated topologies
  - Command line tools for static / on-the-fly needs
    - e.g., iproute2
  - Routing suites such as FRR that speak Linux
  - Monitoring and configuration agents of your choice
    - ansible, puppet, chef, collectd, snmpd, ...

Adding more Linux to the equation



- Still a few SDK hassles
  - Configuring settings on front panel ports requires the SDK
    - Need to provide a few custom hooks
  - Error handling
  - S/W feature vs H/W feature
- Overall, much better architecture for Open Networking

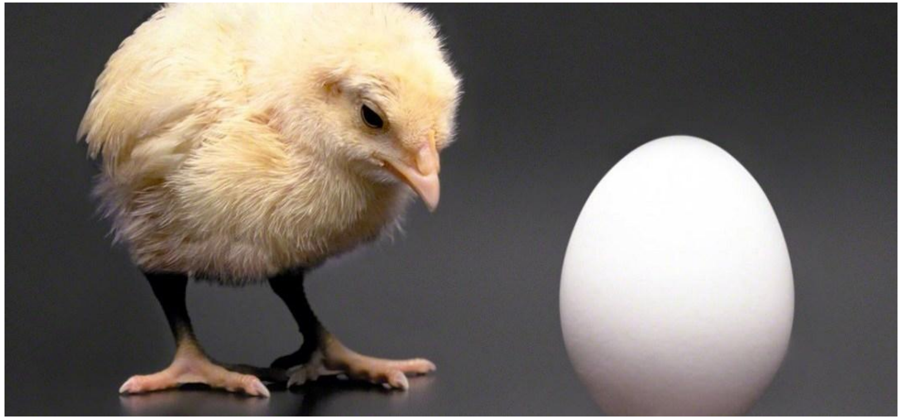
Networking offload with switchdev



- Kernel driver programs ASIC as userspace programs the kernel
- New API for users
  - devlink API for device specific data / control
- Kernel APIs for the driver to learn of changes
  - switchdev operations
  - in-kernel notifiers

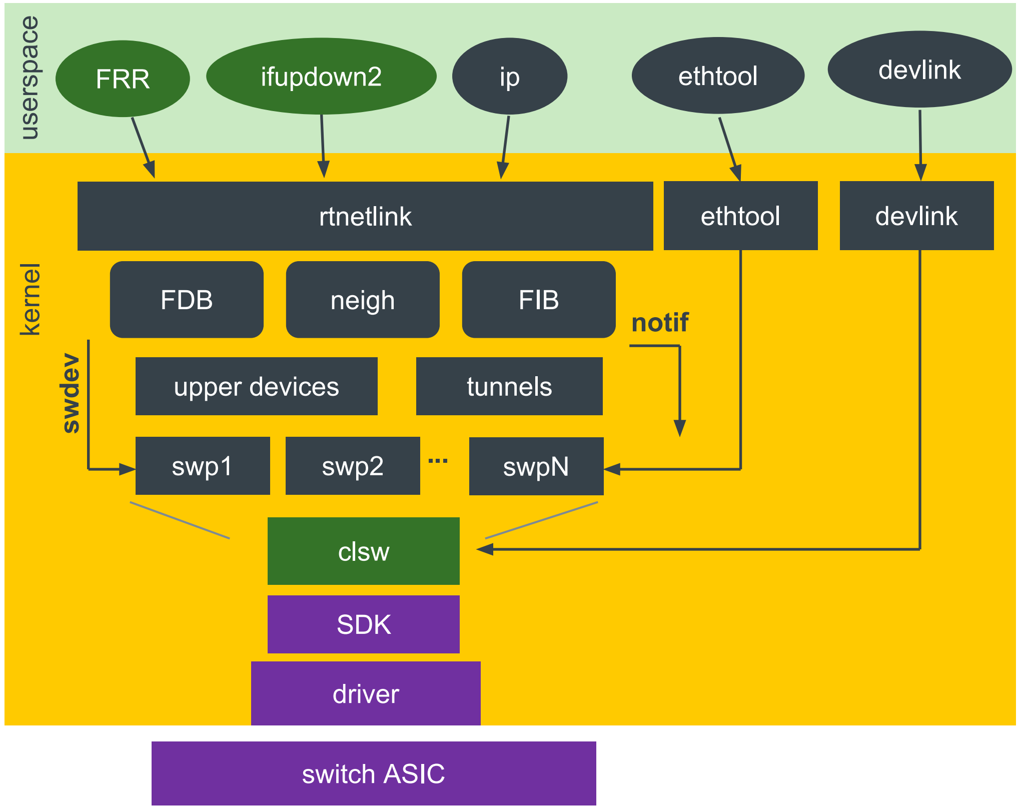
How do we get there?

- Need ASIC vendors to support switchdev model
  - But that’s a lot of work and little incentive to change from SDK
- Need to prove switchdev model is best



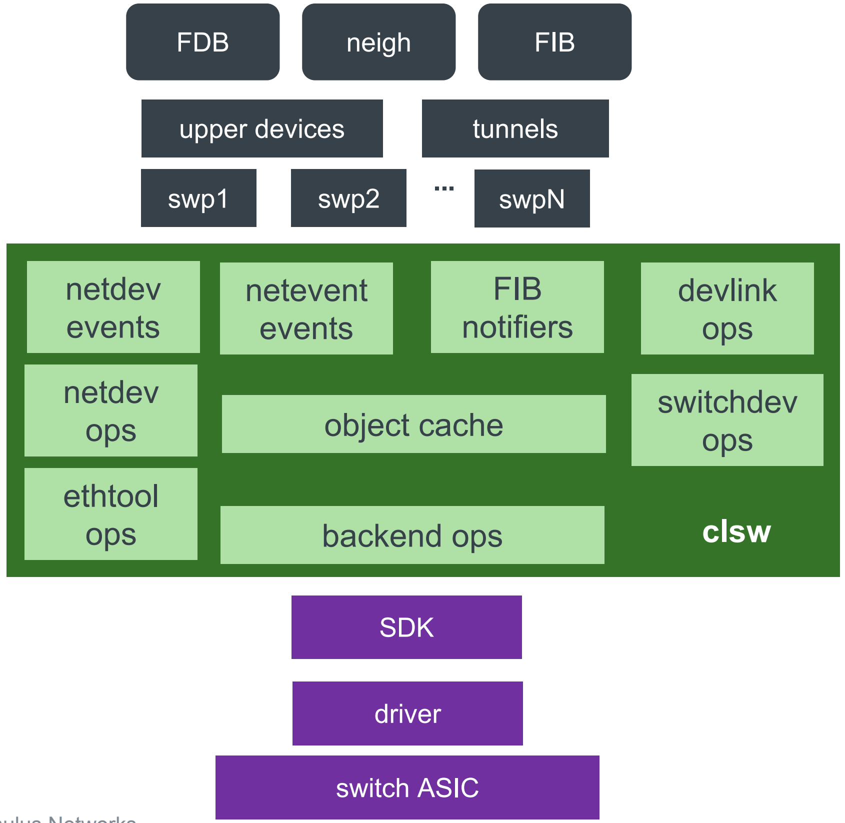
designed by freepik.com

Transitioning SDKs to the switchdev world



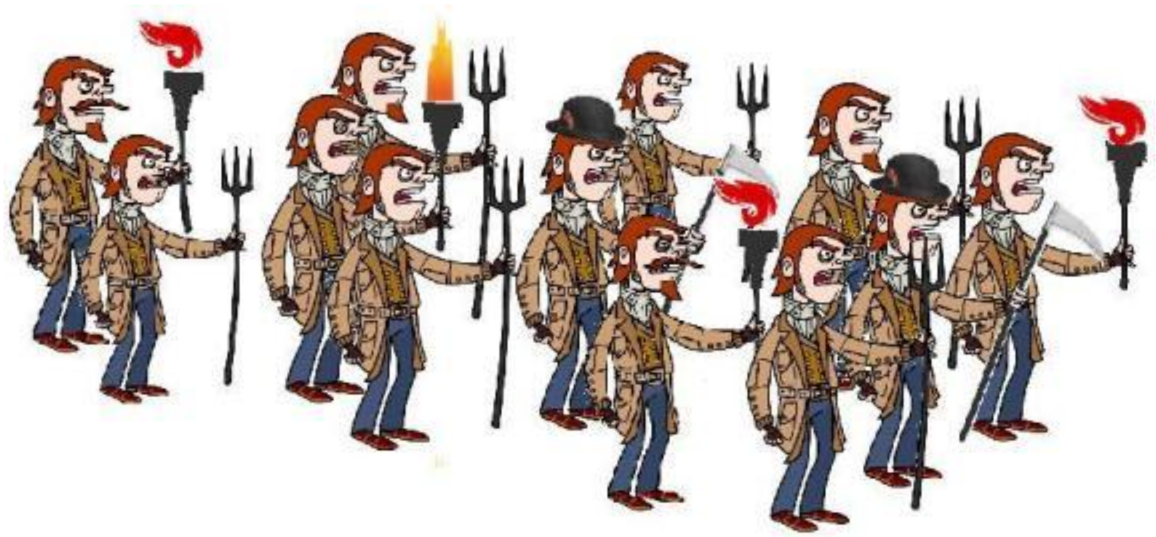
- Move the SDK to the kernel
- A common layer to handle kernel APIs
  - Object cache to map kernel objects to ASIC objects

Common Layer for switchdev



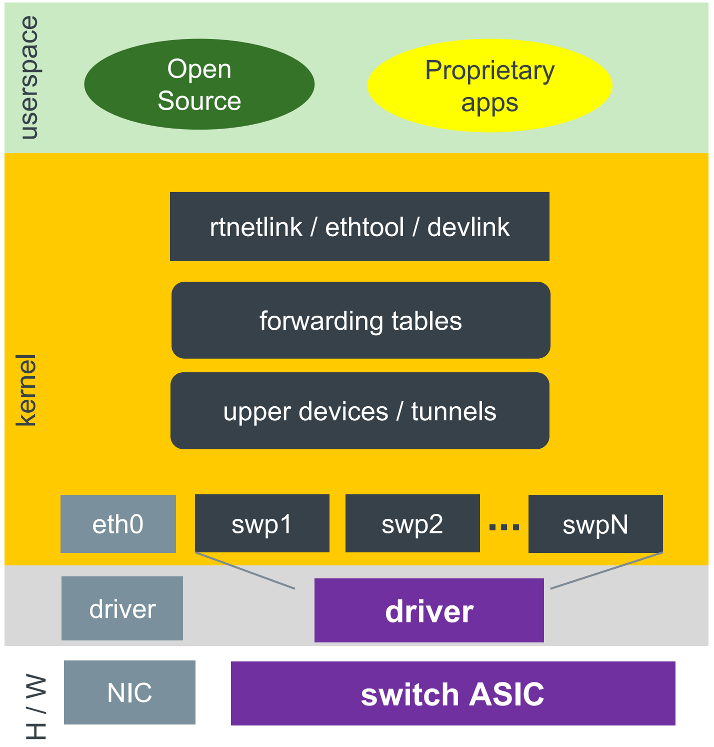
- A lot of the event handling will be common across switch ASICs

Aren’t SDKs Proprietary Blobs !?!?!



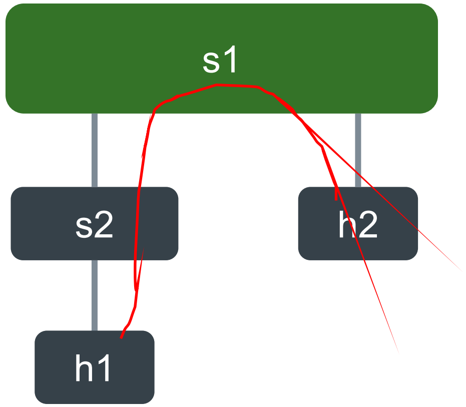
Linux Kernel as the Center of the Open NOS

- Kernel holds networking configuration and state
  - No special process required
  - No custom API required to extract that data
- Less custom software for infrastructure pieces
  - No need for a custom IPC and the performance overhead of ping-ponging between processes
- Simpler design
  - Consistent tools / methodologies for any Linux OS
- Enables true openness and freedom
  - Use any software that speaks Linux APIs
  - Separate proprietary / unique business logic from infrastructure



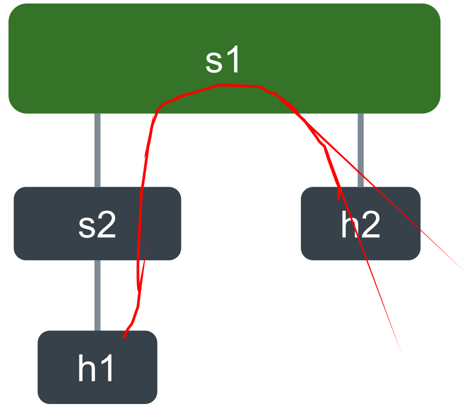
Demonstration / Proof of Concept

- Connectivity between hosts h1 and h2
  - BGP on s1 and s2, peering to exchange connected routes – establishes connectivity
- s1 is one of 3 cases:
  - sdk – userspace SDK driver
  - switchdev – 4.18-rc3 in-kernel driver
  - clsw-sdk – 4.18-rc3 in-kernel SDK based driver with clsw



Reboot via kexec

- Number of missing ping responses between h1 and h2
  - sdk = 72, swdev = 28, clsw-sdk = 27
- Why?
  - Simpler initialization order
- Simpler architecture – no need for complicated “features” (ISSU) that rarely work



CUMULUS

Thank you!

Visit us at cumulusnetworks.com or follow us @cumulusnetworks