Software Defined Fabric for OCP-based Leaf & Spine Switches

Thomas Eklund
VP Marketing and Strategy - Kaloom
Problems with Data Center Networking

- **Lacks automation:** Too labor-intensive and error-prone
- **Lacks programmability:** Prevents developers from driving innovation and customers from adding new services and features themselves
- **Not scalable** to sustain emerging applications and evolving infrastructures
- **Too expensive** and doesn’t leverage white boxes
- **Lacks openness:** Tightly integrated HW and SW, proprietary APIs
- **High end-to-end latency**
- **Unable to guarantee isolated virtual networking slices**
- Lacks proper support for **IPv6**
- **Resource-inefficient:** power, compute, networking resources, engineering personnel
Open Networking
 Standards-based

**Standard Linux**-based
- No kernel patches
- Updates in tandem with compute and storage
- Interfaces towards widely deployed orchestration systems and **SDN controllers**
- Plugins for OpenStack, Kubernetes, and OpenDaylight

**Open APIs**
- NETCONF API based on YANG models

**Open-source** friendly
- Contributing improvements upstream to Linux and Kubernetes

**Open networking** support
- No vendor lock-in
- White box friendly
  - Certified with switches from multiple ODMs
Open Networking HW

- **Disaggregate** the appliance model
  - Separate SW from HW
- Challenges are **standardization** to drive adoption
  - OCP is becoming the leading standard for DC networking
  - $2.5 billion market today (excluding FB and MS), and will grow to $10 billion in three years
- **Commoditizes the networking HW** to drive down cost
  - Commoditizes the networking chipsets, white boxes and PODs
Open Hardware Example

Wedge100BF-32Q/65X Switch Bare Metal Switch from EdgeCore

• OCP Accepted, cost-effective, bare-metal switch infrastructure for data center fabric
• Designed with programmable Tofino switch silicon from Barefoot Networks and XEON-D host processor
• Deploys as Leaf or Spine switch supporting 10GbE, 25 GbE, 50GbE, or 100GbE ports
• Layer 2 or Layer 3 forwarding of 3.2/6.4 Tbps (full duplex)
• Hot-swappable, load-sharing, redundant AC or 48V DC PSUs
• 5/10 redundant, hot-swappable fan modules
OCP Accepted Switch example
# One System Management Approach

*Server like Management*

<table>
<thead>
<tr>
<th></th>
<th>Application Servers</th>
<th>Storage Servers</th>
<th>Leaf Switches</th>
<th>Spine Switches</th>
<th>Fabric Controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remote System Management</strong></td>
<td><img src="image1" alt="OpenBMC" /> Redfish</td>
<td><img src="image2" alt="OpenBMC" /> Redfish</td>
<td><img src="image3" alt="OpenBMC" /> Redfish</td>
<td><img src="image4" alt="OpenBMC" /> Redfish</td>
<td><img src="image5" alt="OpenBMC" /> Redfish</td>
</tr>
<tr>
<td><strong>Network Install</strong></td>
<td><img src="image6" alt="iPXE" /></td>
<td><img src="image7" alt="iPXE" /></td>
<td><img src="image8" alt="onie" /></td>
<td><img src="image9" alt="onie" /></td>
<td><img src="image10" alt="iPXE" /></td>
</tr>
<tr>
<td><strong>Operating System</strong></td>
<td><img src="image11" alt="redhat Enterprise Linux CoreOS" /></td>
<td><img src="image12" alt="redhat Enterprise Linux CoreOS" /></td>
<td><img src="image13" alt="redhat Enterprise Linux CoreOS" /></td>
<td><img src="image14" alt="redhat Enterprise Linux CoreOS" /></td>
<td><img src="image15" alt="redhat Enterprise Linux CoreOS" /></td>
</tr>
</tbody>
</table>
No need for a specialized Linux distribution for switches

<table>
<thead>
<tr>
<th>Feature</th>
<th>Traditional Networking OS</th>
<th>RHEL CoreOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Un-modified Linux Kernel capable of supporting Secure-boot</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Install via ONIE</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Minimum Linux Footprint</td>
<td>NO (&gt; 4GB DDR)</td>
<td>YES (&gt;1GB DDR - lightweight)</td>
</tr>
<tr>
<td>Automatic SW Upgrade with Rollback</td>
<td>NO</td>
<td>YES (RPM_OSTREE)</td>
</tr>
<tr>
<td>Based on SE-Linux</td>
<td>No… for most of them</td>
<td>YES (Secure)</td>
</tr>
<tr>
<td>Optimized for containers</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>DevOps env</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
Fabrics (physical DC) vs vFabrics (virtual DC)

Elastic Network Virtualization and slicing

- A vFabric is a fully elastic isolated network domain
  - Provisioned in software
  - Collection of termination points towards WAN and servers
- A vFabric is a logical switch
  - Delivers integrated NW services
  - Can be part of a virtual data center (vDC)
- A vDC operator offers cloud services
  - Can host millions of cloud service users (e.g. tenants)
Why a programmable data plane?

• It takes too long for the introduction of new functions on traditional fixed functions Ethernet ASICs

• Because there are too many needed functions not supported on current fixed functions Ethernet ASICs
  ▪ Virtual datacenters (e.g. vFabric): Complete isolated broadcast domain
  ▪ In-band Network Telemetry
  ▪ Segment Routing IPv6
  ▪ Geneve (e.g. 24 bits and 32 bits ID)
  ▪ GPRS Tunneling Protocol user-plane for 4G and 5G
  ▪ Etc...

• Because data center operators don’t want to replace hardware to introduce new network capabilities
  ▪ Needs network versioning using slicing
What is P4 and why it matters?

• A high-level programming language intended for packet processors

• Packet processors include Programmable ASICs such as Barefoot Tofino, FPGAs, and CPUs such as Intel XEON

• Keeps the programming language independent of the hardware
  ▪ Contributes to the portability of data plane applications

• P4 is meant to describe/specify the behavior of the data plane application but not how the data plane is actually implemented
Main issues with data plane application

- CPUs introduce **too much latency** for incoming 5G Networks
- CPUs provide **too low throughput** for packet processing applications executing on XEON processors simultaneously serving large number of connected 4G and 5G devices
  - Operators requirement: Over 500K devices/sessions per dual sockets servers
  - Reality: Good performance until there is a maximum of 40K connected devices or active sessions per XEON Scalable
  - Beyond such numbers, CPU is running out-of-cache with a radical drop in packet-rate
- The cost per connected 5G device resulting from a CPU-based Networking Function is **too expensive** for numerous incoming 5G applications
- **Hardware accelerators can provide a significant cost/performance advantage** over CPUs for running data plane applications...at-scale
Emerging Container Network Functions

- Container Network Function (CNF)
- Network Function Control Plane
- Control Plane
  - Go, C, C++ Kubernetes application
  - XEON

Network Function Control Plane

- Barefoot
- Component 1
- P4 Component 2
- Stratix 10/NX
- XEON
- Component 3
- Barefoot

Example

In

Out
DC Fabric Configuration

- Programmable Spine Switches
- Programmable Leaf Switches
- Storage and Application Servers
- SDN Controller
- Edge Switches
- To Other PODs / DCs / Clouds

Data Network
Distributed Fabric Control Plane

- Kubernetes Go-based components
- Scalable cluster
- Fully multi-threaded
- All active nodes
- Redundant
A typical Physical Data Center Fabric Configuration

- Spine Switches
- Edge Switches
- Fabric Controllers
- Networking Rack
- Leaf Switches
- Application servers Rack
Kaloom Software Defined Fabric™ Highlights

1. **Autonomous**
   - Self-Discovering/
     - Self-Forming

2. **Fully virtualizable**
   - Fabric Slicing (vFabric)

3. **Fully Programmable**
   - Future-proof networking

4. **Dataplane Acceleration**
   - vSwitch Offload

5. **Integrated vRouter**

6. **White box**
   - support from multiple vendors, OCP

---

Open. Together.
Upstream contributions in k8s/Linux

• Please join Kaloom™ to work collaboratively in open networking
• Kubernetes and CNI networking improvements in CNF
• KVS and networking improvements in Linux

https://github.com/kaloom/kubernetes-podagent
https://github.com/kaloom/kubernetes-kactus-cni-plugin
Summary of future DC networking requirements

- Open Networking
- OCP based HW
- Programmable
- Fully Automated
- Standard Linux
- Server Style Mgt of networking
- Containerized