Virtualization for NFV/Data Plane SDN

Bob Lantz
Open Networking Laboratory

ONS 2016
Talk Outline

- What is NFV?
- One Way to Do It
- A Better Way
- Control Plane NFV/SDN
- Data Plane SDN
- VM Configuration
- Container Configuration
- Evaluating Virtualization Options
- Conclusions
What is NFV?

For the purpose of this talk, **Network Function Virtualization** is replacing "middlebox" hardware functionality (beyond simple forwarding and routing) with software applications running on a network and/or server OS.

Some example **network/middle box functions** include:

- Firewalls, Load Balancers, NATs, Access Control, Authentication, Intrusion Detection/Mitigation, Packet Inspection, Compression, Caching, VPN/Encryption, Transcoding, Monitoring/Measurement/Statistics, Metering/Billing, etc.
One Way to Do It: Middleboxes -> VMs
A Better Way:
Push functionality to edge and use SDN
Control Plane NFV/SDN

A network function that primarily deals in **packet forwarding decisions** and doesn't require extensive computation or data plane activity can be implemented on a Network OS using existing SDN/OpenFlow switches.

Examples: Firewall, Load Balancer, NAT, Authentication, Simple Statistics/Monitoring/Metering, basic QoS/rate limiting

**Edge switch processing** scales linearly with the number of edge ports.

Centralized control can scale out across multiple nodes of a distributed Network OS.

...but what about **packet processing** in the data plane?
Adding the Data Plane to SDN

Control Program
Network OS
Control Protocol (OpenFlow)
SDN Switch (OpenFlow)

Control Program
Network OS
Control Protocol(s) (OpenFlow?, config? RPC?)

Data Plane Program
Packet Processing Engine
Data Plane SDN: Software + Platform for Network Functions

Data Plane Program (Network Function Application)

P4, Click, OpenFlow Extensions, TPP, Active Networking, C...

Hardware Packet Processor
Programmable pipeline, OpenFlow switch + extensions, P4 parse/match/action hardware, Smart NIC, NPU, FPGA...

Software Packet Processor
OF soft switch + Extensions, P4 Software Switch, EBPF, Click, Software NIC/BESS, kernel, DPDK/netmap, raw sockets...

Virtual or Physical Machine VM, Container, Process...

... but what about overhead and scalability?
VM Configuration for Scalability: “Just Enough OS”

VM

OS + VMM

Network Function Application

init, system processes, daemons, libraries...

OS Kernel

Server

Bare Metal VMM

Network Function Application

Minimal/Library "OS"

VM

Server
# Linux containers: *a-la-carte* OS-level virtualization/isolation

<table>
<thead>
<tr>
<th>Resource</th>
<th>Virtualization/Isolation Method</th>
<th>Linux Command(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU, I/O bandwidth, Memory</td>
<td>processes + cgroups (process control groups)</td>
<td>cgreate, cgset, cgclassify...</td>
</tr>
<tr>
<td>Network Devices</td>
<td>network namespaces</td>
<td>ip netns</td>
</tr>
<tr>
<td>Process IDs, user IDs, hostname</td>
<td>pid, user, uts namespaces</td>
<td>unshare</td>
</tr>
<tr>
<td>Disk/File System</td>
<td>virtual disk, chroot, mount namespaces, overlayfs, union FS</td>
<td>mount, chroot, unshare</td>
</tr>
</tbody>
</table>
Container Configuration for Scalability:
"Just Enough OS + Just Enough Virtualization"

- Network Function App
- System processes
- init
- cgroups
  - CPU
  - Memory
  - I/O
- namespaces
  - User
  - PID
  - Mount
  - UTS
  - Network
- File System
- Virtual Disk Image
- Read-only FS Image
- Private r/w FS (aufs, overlayfs, mnt...)
- Private r/w
<table>
<thead>
<tr>
<th>Method</th>
<th>Software Support</th>
<th>Scalability/Overhead</th>
<th>Complexity</th>
<th>Performance</th>
<th>Isolation</th>
<th>Other: multi-kernel, live migration...</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMs</td>
<td>virsh, OpenStack...</td>
<td>Possibly Poor</td>
<td>High</td>
<td>Good (PCI passthrough, ELI, DPDK / netmap...)</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>Containers</td>
<td>lxc, docker, OpenVZ, OpenStack...</td>
<td>Probably Good</td>
<td>Variable</td>
<td>Good</td>
<td>OK/variable</td>
<td>No.</td>
</tr>
</tbody>
</table>

Summary: Evaluating Virtualization Options for NFV
Conclusions

Control plane network functions (firewall, load balancer, NAT, simple stats etc.) can and should be "virtualized" by implementing them as SDN apps on a Network OS.

Packet processing data plane network functions should use switch features or hardware where available, and will eventually (P4, future OpenFlow, Click, etc.) become data plane SDN apps on a Network OS!

There is a large space of virtualization and container options - what is best will depend on each specific use case. Overhead and complexity are likely to be reduced by a "Just Enough OS + Just Enough virtualization" approach.

The future of data plane SDN is exciting!
Backup slides
Alternatives: VMs, Containers, and Processes

Server

OS kernel + VMM

Container

OS Kernel

Network Function App

init, system processes, daemons, libraries...

Network Function App

Server OS

Server

Network Function App

Server OS

Server
# Evaluating Container Configurations for NFV

<table>
<thead>
<tr>
<th>Container Features</th>
<th>Resembles</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS Disk Image (COW?) + User/pid/UTS/Mount/Network namespaces + cgroups + init</td>
<td>lxc-ish</td>
<td>can boot in VMM; full OS; orchestration support</td>
<td>bulky OS image; extra processes, CPU and memory usage</td>
</tr>
<tr>
<td>OS File System Image (overlayfs) + User/pid/UTS/Mount/Network namespaces + cgroups + init</td>
<td>Docker-ish</td>
<td>full OS; orchestration support</td>
<td>bulky OS image + libraries (might be required anyway?) extra processes, CPU and memory usage</td>
</tr>
<tr>
<td>Underlying r/w file system + Network namespaces + cgroups (optional) bash</td>
<td>Mininet-ish</td>
<td>zero file overhead; no init/OS processes; minimal per-container admin and configuration</td>
<td>minimal isolation/security; Mininet not really designed for NFV</td>
</tr>
<tr>
<td>Underlying read-only File system + mount namespace + cgroups</td>
<td>&quot;Just Enough OS + Just Enough Virtualization for NFV?&quot;</td>
<td>zero file overhead; no init/OS processes; minimal per-container admin and configuration</td>
<td>no good orchestration support (yet?); privilege isolation may require OS configuration (e.g. user IDs); perturbs server OS</td>
</tr>
</tbody>
</table>