fd.io vpp and containers

Keith Burns

@alagalah
kb@cisco.com | alagalah@gmail.com
github.com/alagalah

fd.io Foundation
fd.io intro: why, what, how
Evolution of Programmable Networking
fd.io projects

Legend:
- New Projects
- Core Projects

Network IO

Management Agent

Packet Processing

Honeycomb

NSH_SFC
ONE
VPP Sandbox
TLDK

VPP

Testing/Performance/Support

deb_dpdk

CSIT

fd.io Foundation
## VPP Feature Summary at launch 2016-02-11

<table>
<thead>
<tr>
<th>IPv4/IPv6</th>
<th>IPv4</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>14+ MPPS, single core</td>
<td>GRE, MPLS-GRE, NSH-GRE, VXLAN</td>
<td>VLAN Support</td>
</tr>
<tr>
<td>Multimillion entry FIBs</td>
<td>IPSEC</td>
<td>Single/ Double tag</td>
</tr>
<tr>
<td>Source RPF</td>
<td>DHCP client/proxy</td>
<td>L2 forwarding with EFP/ BridgeDomain concepts</td>
</tr>
<tr>
<td>Thousands of VRFs</td>
<td>CG NAT</td>
<td>VTR – push/pop/Translate (1:1,1:2, 2:1,2:2)</td>
</tr>
<tr>
<td>Controlled cross-VRF lookups</td>
<td></td>
<td>Mac Learning – default limit of 50k addresses</td>
</tr>
<tr>
<td>Multipath – ECMP and Unequal Cost</td>
<td>Neighbor discovery</td>
<td>Bridging – Split-horizon group support/ EFP Filtering</td>
</tr>
<tr>
<td>Multiple million Classifiers – Arbitrary N-tuple</td>
<td>Router Advertisement</td>
<td>Proxy Arp</td>
</tr>
<tr>
<td>VLAN Support – Single/Double tag</td>
<td>DHCPv6 Proxy</td>
<td>Arp termination</td>
</tr>
<tr>
<td>Mandatory Input Checks:</td>
<td>L2TPv3</td>
<td>IRB – BVI Support with RouterMac assignment</td>
</tr>
<tr>
<td>TTL expiration</td>
<td>Segment Routing</td>
<td>Flooding</td>
</tr>
<tr>
<td>header checksum</td>
<td>MAP/LW46 – IPv4aas</td>
<td>Input ACLs</td>
</tr>
<tr>
<td>L2 length &lt; IP length</td>
<td>iOAM</td>
<td>Interface cross-connect</td>
</tr>
<tr>
<td>ARP resolution/snooping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARP proxy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Mandatory Input Checks:
- TTL expiration
- Header checksum
- L2 length < IP length
- ARP resolution/snooping
- ARP proxy

### VLAN Support
- Single/ Double tag
- L2 forwarding with EFP/ BridgeDomain concepts
- VTR – push/pop/Translate (1:1,1:2, 2:1,2:2)
- Mac Learning – default limit of 50k addresses
- Bridging – Split-horizon group support/ EFP Filtering
- Proxy Arp
- Arp termination
- IRB – BVI Support with RouterMac assignment
- Flooding
- Input ACLs
- Interface cross-connect
# VPP Feature Summary at launch

**IPv4/IPv6**
- 14+ MPPS, single core
- Multimillion entry FIBs
- Source RPF
- Thousands of VRFs
  - Controlled cross-VRF lookups
- Multipath – ECMP and Unequal Cost
- Multiple million Classifiers –
  - Arbitrary N-tuple
- VLAN Support – Single/Double tag
- Mandatory Input Checks:
  - TTL expiration
  - header checksum
  - L2 length < IP length
  - ARP resolution/snooping
  - ARP proxy

**IPv4**
- GRE, MPLS-GRE, NSH-GRE, VXLAN
- IPSEC
- DHCP client/proxy
- CG NAT

**IPv6**
- Neighbor discovery
- Router Advertisement
- DHCPv6 Proxy
- L2TPv3
- Segment Routing
- MAP/LW46 – IPv4aas
- iOAM

**MPLS**
- MPLS-o-Ethernet –
  - Deep label stacks
  - supported

**L2**
- VLAN Support
  - Single/ Double tag
- L2 forwarding with EFP/
  - L2 features
- Bridging – Split-horizon group support/
  - EFP Filtering
- Proxy Arp
- Arp termination
- IRB – BVI Support with RouterMac
  - assignment
- Flooding
- Input ACLs
- Interface cross-connect

---

Counters for everything!!!

fd.io Foundation
VPP 16.06 Release

- Enhanced Switching & Routing
  - IPv6 Segment Routing multicast support
  - LISP xTR support
  - VXLAN over IPv6 underlay
  - per interface whitelists
  - shared adjacencies in FIB

- Expanded Hardware and Software Support
  - Support for ARM 32 targets
  - Support for Raspberry Pi
  - Support for DPDK 16.04

- New and improved interface support
  - jumbo frame support for vhost-user
  - Netmap interface support
  - AF_Packet interface support

- Programmability
  - Python API bindings
  - Enhanced JVPP Java API bindings
  - Enhanced debugging cli
VPP 16.09 Release

- Enhanced LISP support for
  - L2 overlays
  - Multi-tenancy
  - Multi-homing
  - Re-encapsulating Tunnel Routers (RTR) support
  - Map-Resolver failover algorithm

- New “in-tree” plugins for
  - SNAT
  - MagLev-like Load Balancer
  - Identifier Locator Addressing (ILA)

- High performance port range ingress filtering

- Dynamically ordered subgraphs
  - Allows registration of node ‘before’ another node
Honeycomb 16.09 Release

• Infrastructure
  • Data processing pipeline
  • Extensible translation layer (SPI)
  • Configuration and context persistence

• Yang models exposing VPP features:
  • Interfaces:
    • Base interface management – ietf-interface + ietf-ip models
    • vhost-user, Linux tap interface management
  • Bridge domain management
  • Overlays / Encapsulations
    • VLAN, VXLAN, VXLAN-GPE, GRE management
  • NSH_SFC plugin support
  • ACLs
    • L2/L3 ACL management – ietf-acl
    • LISP – mapping server configuration
    • Bit level granularity classifier interface
NSH_SFC 16.09 Release

• SFF functionality
• NSH Proxy for SF
• Transport:
  • VXLAN-GPE
  • GRE
• API
  • Automatically generated jar for java bindings
• Integrated with OpenDaylight SFC
Implementation Example: VPP as a vRouter/ vSwitch

- **vSwitch/vRouter**
  - Including CLI
- **Switching**
  - Bridge Domains
  - BVI interfaces
  - Split-Horizon Groups
  - Program ARP termination
- **Routing**
  - VRFs (FIBs) - thousands
  - IPv4 / IPv6 routes – millions
  - 700K updates per second
VPP Architecture

- Instruction/Data cache efficiency
- Graph composed at runtime
- Easy to create and incorporate new features
- All in user space

Packet vector

Plug-in to create new nodes

Custom-A

Custom-B
Plugins

- First-class graph node citizens
  - Introduce new graph nodes
  - Graph composed at runtime, nodes discovered
- Rearrange packet processing graph
- Can be built independently of VPP source tree
- Ability to take advantage of diverse hardware when present
VPP vRouter/vSwitch: Local Programmability

Low Level API
- Complete
- Feature Rich
- High Performance
  - Example: 900k routes/s
- Shared memory/message queue
- Box local
- All CLI tasks can be done via API

Generated Low Level Bindings - existing today
- C, Java and Python API bindings
- Others can be done
Containers
**Chunky case - video**

Uncompressed data rate = color depths * vertical resolution * horizontal resolution * refresh frequency

<table>
<thead>
<tr>
<th>UHDTV 4K formats</th>
<th>Uncompressed</th>
<th>TICO compression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10-bit-422 Gbps</td>
<td>10-bit-444 Gbps</td>
</tr>
<tr>
<td>Samples per Line</td>
<td>Lines</td>
<td>fps</td>
</tr>
<tr>
<td>3840</td>
<td>2160</td>
<td>50</td>
</tr>
<tr>
<td>3840</td>
<td>2160</td>
<td>59,94</td>
</tr>
<tr>
<td>3840</td>
<td>2160</td>
<td>120</td>
</tr>
</tbody>
</table>

**Ethernet Type**

<table>
<thead>
<tr>
<th>Type</th>
<th>10GbE</th>
<th>40GbE</th>
</tr>
</thead>
</table>

*August 27, 2015, TVTechnology*
“Encoding in **multiple formats in parallel** including raw uncompressed per camera”

- BBC: IP Studio project

… “Destination-timed switching is probably the simplest way to switch video on commodity Ethernet switches, but it generally requires **twice the bandwidth** of a single video signal to be reserved.”

- Thomas Edwards of Fox, June 10, 2015, TVTechnology.com
Death by a thousand cuts case - distributed

“Compute is going to everywhere, …

... compute will be distributed from end points and in layers of networks before data is even shipped back into the datacenter...

..there could be as much processing outside the “server” and the “datacenter” as inside of it. These terms could become somewhat meaningless.”

- Peak X86, The Next Platform, Sep 15, 2016
Implications of reality...

100Gbps NICs are reality

Use cases exist *today* for 100Gbps per workload

3D XPoint and Memristor technology is reality

Machine learning algorithms being held back by lack of real-time streaming instrumentation
ONE DOES NOT SIMPLY WALK INTO MORDOR
Implications of containers

Containers are small
Implications of containers

Containers are small

- Quicker to start/stop on demand
- Can afford to have more of them
- Easier to develop with
Implications of containers

- Quicker to start/stop on demand
- Can afford to have more of them
- Easier to develop with
- New ways of designing applications
- New deployment models
Implications of containers

New ways of designing applications

New deployment models
Implications of containers

New ways of designing applications

New deployment models

Microservices
Implications of containers

New ways of designing applications

New deployment models

Microservices

Application components run in own process

Simpler API - REST
Implications of containers

- New ways of designing applications
- New deployment models
Implications of containers

New ways of designing applications
New deployment models

Described by container metadata
Horizontally scaled
Kubernetes 1.3 - 2,000 node 60,000 pod clusters

Implications of containers

Lots of individually addressed elements to manage

Lots of sessions to scale

Predictable performance under load

Instrumentation - Do you know who is talking to who? How much?
If REST is the new SOAP....

... TCP and IPC becomes even more important.
How is container networking done today?
How is container networking done today?
Why not this?
control plane
Project Calico – key Principles

**BGP**
Distribute routes using proven Border Gateway Protocol*, with route reflectors for scale

**IP**
Perform layer 3 forwarding at each compute node

**Lock**
Separate policy decisions from routing information
Translate global policy into distributed firewall on each host, enabling tenant isolation & more
How do I provision it?
How do I provision it?

calicootl pool add 192.168.0.0/16

docker run --net=none --name workload-A -tid

busybox

sudo calicoctl container add workload-A 192.168.0.1

calicootl profile add PROF_A

calicootl container workload-A profile append

PROF_A
Calico Network Policy

container, VM, bare metal

(virtual) network interface

reusable policy

Workload

Endpoint

Profile

Rules

Tag

Profile

Tag

Profile

Endpoint

Workload
Putting it all together
You said that one does not simply walk into Mordor
But I totally DID
Just sayin’
Next Steps – Get Involved

We invite you to Participate in fd.io

- Get the Code, Build the Code, Run the Code
- Try the vpp user demo
- Install vpp from binary packages (yum/apt)
- Read/Watch the Tutorials
- Join the Mailing Lists
- Join the IRC Channels
- Explore the wiki
- Join fd.io as a member
Thanks.

Keith Burns

@alagalah

krb@cisco.com | alagalah@gmail.com

github.com/alagalah