The Universal Dataplane
FD.io: The Universal Dataplane

- Project at Linux Foundation
  - Multi-party
  - Multi-project
- Software Dataplane
  - High throughput
  - Low Latency
  - Feature Rich
  - Resource Efficient
  - Bare Metal/VM/Container
  - Multiplatform

Fd.io Scope:
- Network IO - NIC/vNIC <-> cores/threads
- Packet Processing – Classify/Transform/Prioritize/Forward/Terminate
- Dataplane Management Agents - ControlPlane

Bare Metal/VM/Container
- Dataplane Management Agent
- Packet Processing
- Network IO
Multiparty: Broad Membership

Service Providers
- AT&T
- Comcast

Network Vendors
- Cisco
- Ericsson
- Huawei
- ZTE
- Metaswitch
- Brocade

Chip Vendors
- Intel
- Cavium Networks

Integrators
- Red Hat
- Inocybe
Multiparty: Broad Contribution
In the period since its inception, fd.io has more commits than OVS and DPDK combined, and more contributors than OVS.

<table>
<thead>
<tr>
<th></th>
<th>Fd.io</th>
<th>OVS</th>
<th>DPDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commits</td>
<td>6283</td>
<td>2395</td>
<td>3289</td>
</tr>
<tr>
<td>Contributors</td>
<td>163</td>
<td>146</td>
<td>245</td>
</tr>
<tr>
<td>Organizations</td>
<td>42</td>
<td>52</td>
<td>78</td>
</tr>
</tbody>
</table>

![Commits](image1)

![Contributors](image2)

![Organizations](image3)
Multiproject: Fd.io Projects

Dataplane Management Agent
- Honeycomb
- hc2vpp

Packet Processing
- NSH_SFC
- ONE
- TLDK
- CICN
- odp4vpp
- VPP Sandbox
- VPP

Network IO
- deb_dpdk
- rpm_dpdk

Testing/Support
- CSIT
- puppet-fdio
- trex
Fd.io Integrations

Integration work done at OPNFV
Vector Packet Processor - VPP

- Packet Processing Platform:
  - High performance
  - Linux User space
  - Run’s on commodity CPUs: Intel/ARM
  - Shipping at volume in server & embedded products since 2004.
VPP Architecture: Packet Processing

Packet Vector of n packets

ethernet-input

dpdk-input vhost-user-input ... af-packet-input

ip4-input mpls-input

ip6-input

ip6-lookup ip4-lookup

ip6-rewrite ip6-local ip4-local ip4-rewrite

Packet Processing Graph

Input Graph Node

Graph Node

Packet

Input Graph Node

Vector of n packets
VPP Architecture: Splitting the Vector

Packet Processing Graph

Input Graph Node

Graph Node

Packet

Vector of n packets

Vector of n packets

dpdk-input
vhost-user-input
af-packet-input

ethernet-input

ip6-input
ip4-input
mpls-input
arp-input

ip6-lookup
ip4-lookup

ip6-rewrite
ip6-local
ip4-local
ip4-rewrite
VPP Architecture: Plugins

Packet Vector of n packets

ethernet-input

dpdk-input

vhost-user-input

... af-packet-input

ip4-input

ip6-input

mpls-input

arp-input

ip6-lookup

ip4-lookup

ip6-rewrite

ip6-local

ip4-rewrite

ip4-local

custom-1

custom-2

custom-3

Hardware Plugin

Plugins are: First class citizens That can: Add graph nodes Add API Rearrange the graph

Can be built independently of VPP source tree

Plugin
/usr/lib/vpp_plugins/foo.so

Packet Processing Graph

Input Graph Node

Graph Node

Vector of n packets

Skip sftw nodes where work is done by hardware already

Hardware Plugin

hw-accel-input

dpdk-input

vhost-user-input

...
VPP Architecture: Programmability

**Architecture**

- Linux Hosts
- Shared Memory
  - Request Queue
  - Response Queue
- Async Response Message
- Can use C/Java/Python/Lua Language bindings
- Control Plane Protocol
- Request Message 900k request/s

**Example: Honeycomb**

- Linux Hosts
- Shared Memory
  - Request Queue
  - Response Queue
- Async Response Message
- Netconf/Restconf/Yang
- Request Message
- Honeycomb Agent
- VPP
Universal Dataplane: Performance at Scale

**IPv6, 24 of 72 cores**

- [Gbps]
  - 12 routes: 480Gbps zero frame loss
  - 1k routes: 500Gbps zero frame loss
  - 100k routes: 500Gbps zero frame loss
  - 500k routes: 500Gbps zero frame loss
  - 1M routes: 500Gbps zero frame loss
  - 2M routes: 500Gbps zero frame loss

**IPv4+ 2k Whitelist, 36 of 72 cores**

- [Gbps]
  - 1k routes: IMIX => 342 Gbps, 1518B => 462 Gbps
  - 500k routes: 500Gbps zero frame loss
  - 1M routes: 500Gbps zero frame loss
  - 2M routes: 500Gbps zero frame loss
  - 4M routes: 500Gbps zero frame loss
  - 8M routes: 500Gbps zero frame loss

- [Mpps]
  - 12 routes: 200Mpps zero frame loss
  - 1k routes: 200Mpps zero frame loss
  - 100k routes: 200Mpps zero frame loss
  - 500k routes: 200Mpps zero frame loss
  - 1M routes: 200Mpps zero frame loss
  - 2M routes: 200Mpps zero frame loss

**Hardware:**
- Cisco UCS C460 M4
  - Intel® C610 series chipset
  - 4 x Intel® Xeon® Processor E7-8890 v3 (18 cores, 2.5GHz, 45MB Cache)
  - 2133 MHz, 512 GB Total
  - 9 x 2p40GE Intel XL710
  - 18 x 40GE = 720GE !!

**Latency**
- 18 x 7.7trillion packets soak test
- Average latency: <23 usec
- Min Latency: 7…10 usec
- Max Latency: 3.5 ms

**Headroom**
- Average vector size ~24-27
- Max vector size 255
- Headroom for much more throughput/features
- NIC/PCI bus is the limit not vpp
Universal Dataplane: Features

**Hardware Platforms**
- Pure Userspace - X86, ARM 32/64, Power
- Raspberry Pi

**Interfaces**
- DPDK/Netmap/AF_Packet/TunTap
- Vhost-user - multi-queue, reconnect, Jumbo Frame Support

**Language Bindings**
- C/Java/Python/Lua

**Tunnels/Encaps**
- GRE/VXLAN/VXLan-GPE/LISP-GPE/NSH
- IPSEC
- Including HW offload when available

**MPLS**
- MPLS over Ethernet/GRE
- Deep label stacks supported

**Routing**
- IPv4/IPv6
- 14+ MPPS, single core
- Hierarchical FIBs
- Multimillion FIB entries
- Source RPF
- Thousands of VRFs
- Controlled cross-VRF lookups
- Multipath – ECMP and Unequal Cost

**Segment Routing**
- SR MPLS/IPv6
- Including Multicast

**LISP**
- LISP xTR/RTR
- L2 Overlays over LISP and GRE encaps
- Multitenancy
- Multihome
- Map/Resolver Failover
- Source/Dest control plane support
- Map-Register/Map-Notify/RLOC-probing

**Switching**
- VLAN Support
- Single/ Double tag
- L2 forwd w/EFP/BridgeDomain concepts
- VTR – push/pop/Translate (1:1,1:2, 2:1,2:2)
- Mac Learning – default limit of 50k addr
- Bridging
- Split-horizon group support/EFP Filtering
- Proxy Arp
- Arp termination
- IRB - BVI Support with RouterMac assignmt
- Flooding
- Input ACLs
- Interface cross-connect
- L2 GRE over IPsec tunnels

**Security**
- Mandatory Input Checks:
  - TTL expiration
  - header checksum
  - L2 length < IP length
  - ARP resolution/snooping
  - ARP proxy
  - SNAT
  - Ingress Port Range Filtering
  - Per interface whitelists
  - Policy/Security Groups/GBP (Classifier)

**Network Services**
- DHCPv4 client/proxy
- DHCPv6 Proxy
- MAP/LW46 – IPv4aas
- MagLev-like Load
- Identifier Locator Addressing
- NSh SFC SFF’s & NSH Proxy
- LLDP
- BFD
- Policer
- Multiple million Classifiers – Arbitrary N-tuple

**Monitoring**
- Simple Port Analyzer (SPAN)
- IP Flow Export (IPFIX)

**Counters for everything**
- Lawful Intercept

**Inband iOAM**
- Telemetry export infra (raw IPFIX)
- iOAM for VXLAN-GPE (INGENA)
- SRv6 and iOAM co-existence
- iOAM proxy mode / caching
- iOAM probe and responder

**Monitoring**
- Simple Port Analyzer (SPAN)
- IP Flow Export (IPFIX)
- Counters for everything
- Lawful Intercept

fd.io Foundation
Rapid Release Cadence – ~3 months

16-02
Fd.io launch

16-06
Release - VPP

16-06 New Features
Enhanced Switching & Routing
IPv6 SR multicast support
LISP xTR support
VXLAN over IPv6 underlay
per interface whitelists
shared adjacencies in FIB
Improves interface support
vhost-user – jumbo frames
Netmap interface support
AF_PACKET interface support
Improved programmability
Python API bindings
Enhanced JVPP Java API bindings
Enhanced debugging cli
Hardware and Software Support
Support for ARM 32 targets
Support for Raspberry Pi
Support for DPDK 16.04

16-09
Release:
VPP, Honeycomb,
NSH_SFC, ONE

16-09 New Features
Enhanced LISP support for
L2 overlays
Multitenancy
Multihoming
Re-encapsulating Tunnel Routers (RT)
Map-Resolver failover algorithm
New plugins for
SNAT
MagLev-like Load
Identifier Locator Addressing
NSH SFC SFF’s & NSH Proxy
Port range ingress filtering
Dynamically ordered subgraphs

17-01
Release:
VPP, Honeycomb,
NSH_SFC, ONE

17-01 New Features
Hierarchical FIB
Performance Improvements
DPDK input and output nodes
L2 Path
IPv4 lookup node
IPSEC
Software HWCrypto Support
HQoS support
Simple Port Analyzer (SPAN)
BFD
IPFIX Improvements
L2 GRE over IPSec tunnels
LLDP
LISP Enhancements
Source/Dest control plane
L2 over LISP and GRE
Map-Register/Map-Notify
RLOC-probing
ACL
Flow Per Packet
SNAT – Multithread, Flow Export
LUA API Bindings
New in 17.04 – Due Apr 19

**VPP Userspace Host Stack**
- TCP stack
- DHCPv4 relay multi-destination
- DHCPv4 option 82
- DHCPv6 relay multi-destination
- DHCPv6 relay remote-id
- ND Proxy

**Security Groups**
- Routed interface support
- L4 filters with IPv6 Extension Headers

**API**
- Move to CFFI for Python binding
- Python Packaging improvements
- CLI over API
- Improved C/C++ language binding

**Segment Routing v6**
- SR policies with weighted SID lists
- Binding SID
- SR steering policies
- SR LocalSIDs
- Framework to expand local SIDs w/plugins

**SNAT**
- CGN: Configurable port allocation
- CGN: Configurable Address pooling
- CPE: External interface
- DHCP support
- NAT64, LW46

**iOAM**
- UDP Pinger w/path fault isolation
- IOAM as type 2 metadata in NSH
- IOAM raw IPFIX collector and analyzer
- Anycast active server selection

**IPFIX**
- Collect IPv6 information
- Per flow state
Continuous Quality, Performance, Usability

Built into the development process – patch by patch

Submit → Automated Verify → Code Review → Merge → Publish Artifacts

**Build/Unit Testing**
- **120 Tests/Patch**
- Build binary packaging for Ubuntu 14.04, Ubuntu 16.04, CentOS 7
- Automated Style Checking
- **Unit test:**
  - IPFIX
  - BFD
  - Classifier
  - DHCP
  - GRE
  - IPv4
  - IPv4 IRB
  - IPv4 multi-VRF
  - IPv6
  - IP Multicast
  - L2 FIB
  - L2 Bridge Domain
  - MPLS
  - SNAT
  - SPAN
  - VXLAN

**System Functional Testing**
- **252 Tests/Patch**
- DHCP – Client and Proxy
- GRE Overlay Tunnels
- L2BD Ethernet Switching
- L2 Cross Connect Ethernet Switching
- LISP Overlay Tunnels
- IPv4-in-IPv6 Softwire Tunnels
- Cop Address Security
- IPSec
- IPv6 Routing – NS/ND, RA, ICMPv6
- uRPF Security
- Tap Interface
- Telemetry – IPFIX and Span
- VRF Routed Forwarding
- IPv4 Routing
- QoS Policer Metering
- VLAN Tag Translation
- VXLAN Overlay Tunnels

**Performance Testing**
- **144 Tests/Patch, 841 Tests**
- L2 Cross Connect
- L2 Bridging
- IPv4 Routing
- IPv6 Routing
- IPv4 Scale – 20k,200k,2M FIB Entries
- IPv4 Scale - 20k,200k,2M FIB Entries
- VM with vhost-user
  - PHYS-VPP-VM-VPP-PHYS
- L2 Cross Connect/Bridge
- VXLAN w/L2 Bridge Domain
- IPv4 Routing
- COP – IPv4/IPv6 whiteless
- iACL – ingress IPv4/IPv6 ACLs
- LISP – IPv4-0-IPv6/IPv6-0-IPv4
- VXLAN
- QoS Policer
- L2 Cross over
- L2 Bridging

**Usability**
- Merge-by-merge:
  - apt installable deb packaging
  - yum installable rpm packaging
  - autogenerated code documentation
  - autogenerated cli documentation
- Per release:
  - autogenerated testing reports
  - report perf improvements
- Puppet modules
- Training/Tutorial videos
- Hands-on-usecase documentation

Run on real hardware in fd.io Performance Lab

Merge-by-merge packaging feeds Downstream consumer CI pipelines
Universal Dataplane: Infrastructure

- **Bare Metal**
  - Server
  - FD.io
  - Kernel/Hypervisor

- **Cloud/NFVi**
  - Server
  - VM
  - VM
  - VM
  - FD.io
  - Kernel/Hypervisor

- **Container Infra**
  - Server
  - Con
  - Con
  - Con
  - FD.io
  - Kernel
Universal Dataplane: VNFs

FD.io based VNFs

Server

VM
FD.io

VM
FD.io

FD.io

Kernel/Hypervisor

FD.io based VNFs

Server

Con
FD.io

Con
FD.io

FD.io

Kernel/Hypervisor
Universal Dataplane: Embedded

Embedded Device
- Device
  - FD.io
  - Kernel/Hypervisor
  - Hw Accel

SmartNic
- Server
  - Kernel/Hypervisor
  - SmartNic
  - FD.io
  - Hw Accel

fd.io Foundation
Universal Dataplane: CPE Example

- **Physical CPE**
  - Device
  - Kernel/Hypervisor
  - Hw Accel
  - FD.io

- **vCPE in a VM**
  - Server
  - VM
    - FD.io
  - VM
    - FD.io
  - Kernel/Hypervisor
  - FD.io

- **vCPE in a Container**
  - Server
  - Con
    - FD.io
  - Con
    - FD.io
  - Kernel/Hypervisor
  - FD.io
Opportunities to Contribute

- Firewall
- IDS
- Hardware Accelerators
- Integration with OpenCache
- Control plane – support your favorite SDN Protocol Agent
- Spanning Tree
- DPI
- Test tools
- Cloud Foundry Integration
- Container Integration
- Packaging
- Testing

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)
- Install Honeycomb from binary packages
- Read/Watch the Tutorials
- Join the Mailing Lists
- Join the IRC Channels
- Explore the wiki
- Join fd.io as a member