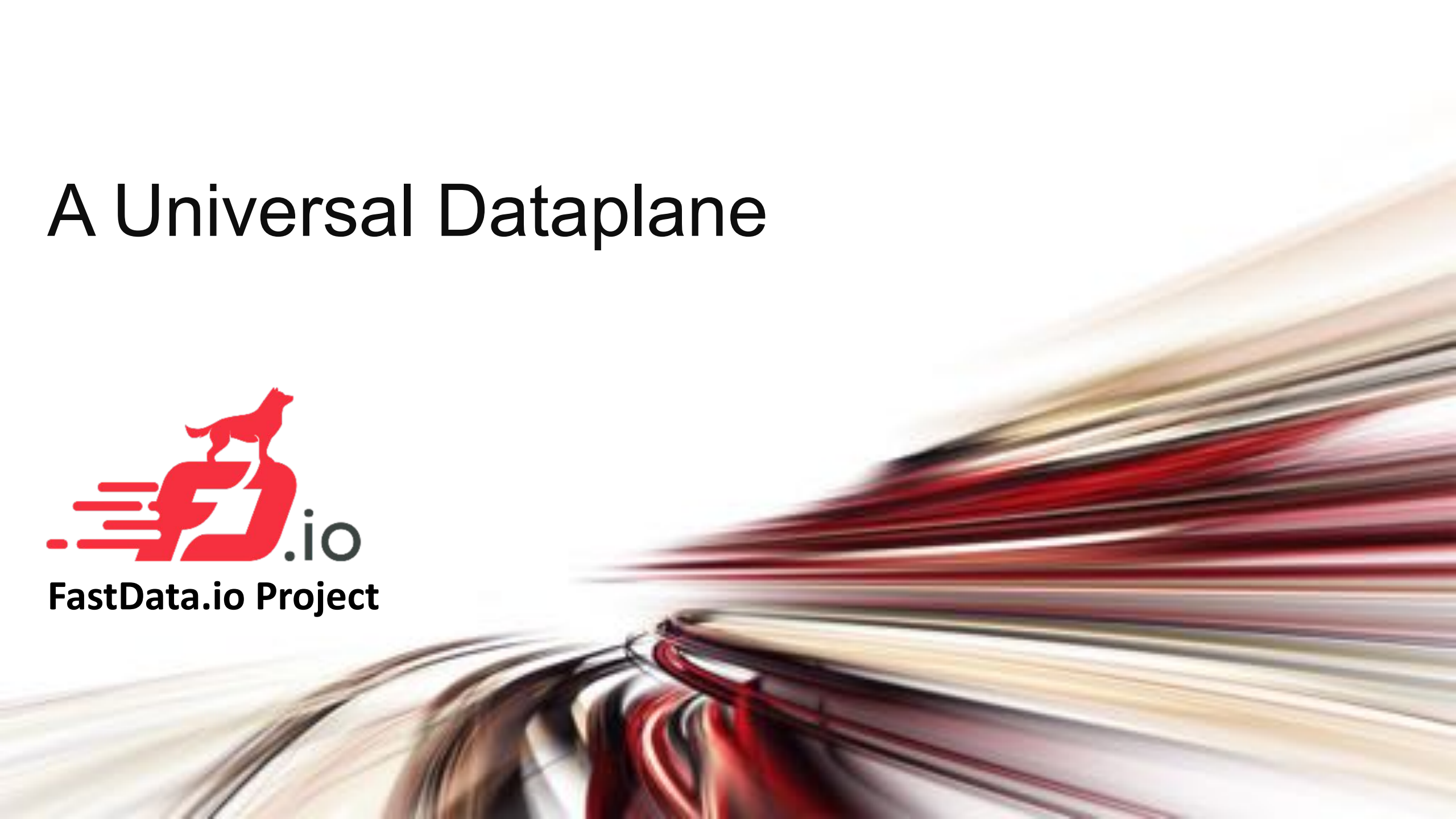
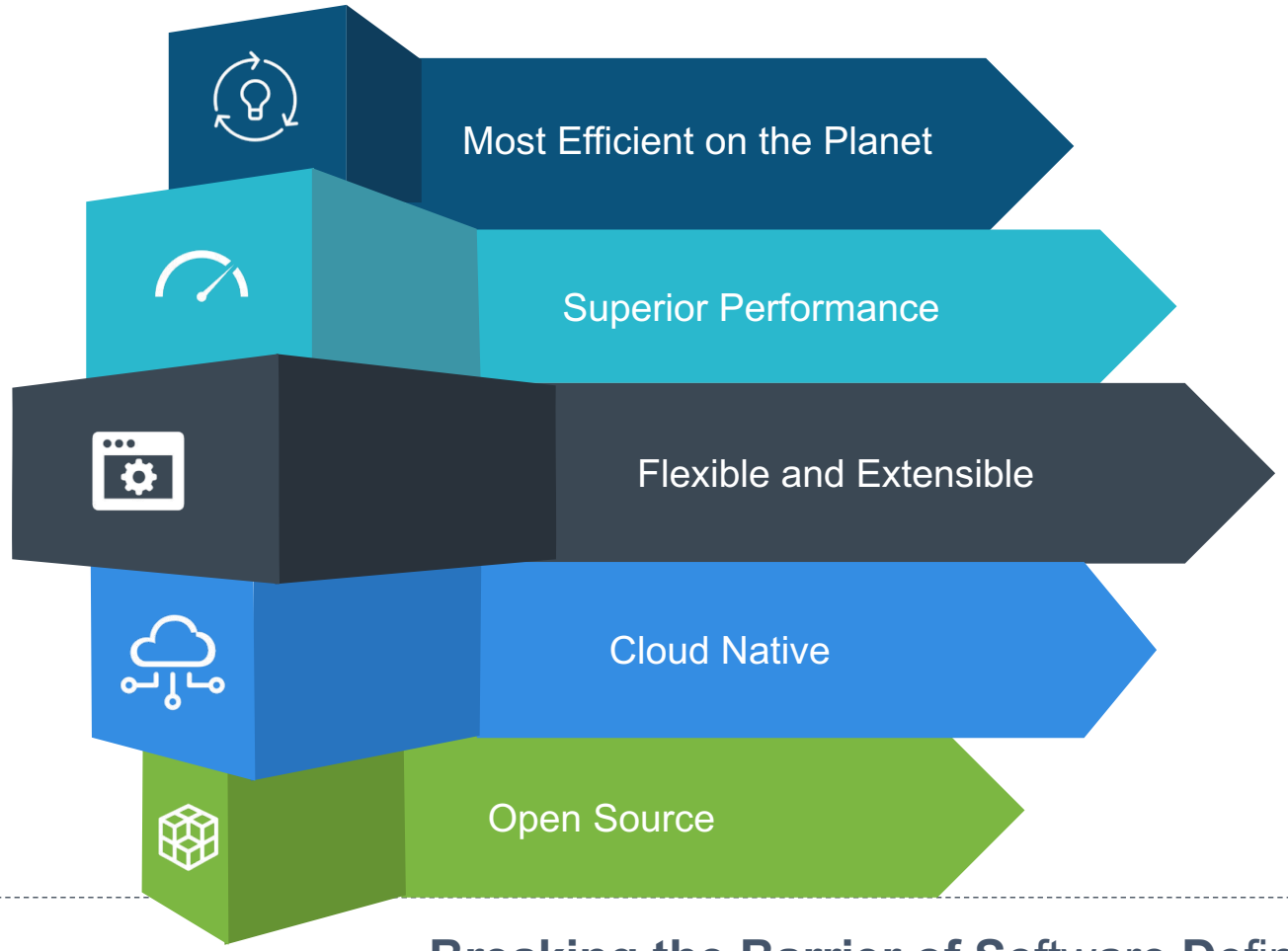
A Universal Dataplane



FastData.io Project

FD.io: A Universal Dataplane

Platform for Native Cloud Network Services



EFFICIENCY

The most efficient software data plane Packet Processing on the planet

PERFORMANCE

FD.io on x86 servers outperforms specialized packet processing HW

SOFTWARE DEFINED NETWORKING

Software programmable, extendable and flexible

CLOUD NETWORK SERVICES

Foundation for cloud native network services

LINUX FOUNDATION

Open source collaborative project in Linux Foundation

Breaking the Barrier of Software Defined Network Services
1 Terabit Services on a Single Intel® Xeon® Server !!!

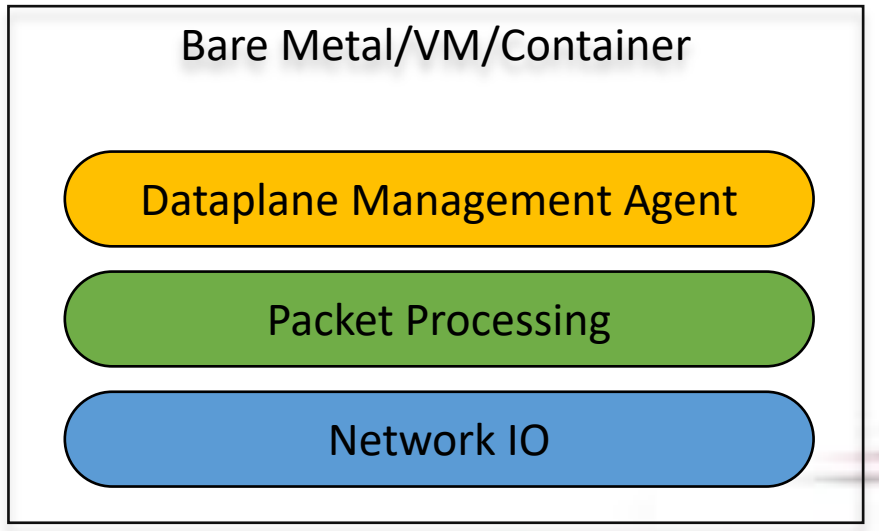
What is FD.io VPP

- Is it a software router? A virtual switch?
- A virtual network function? Or, something else?
- In fact it is all of the above and a whole lot more.
- It is a modularized and extensible software framework for building bespoke network data plane applications.
- And equally importantly, VPP code is written for modern CPU compute platforms (x86_64, ARMv8, PowerPC, to name a few), with a great deal of care and focus given to optimizing the software-hardware interface for real-time, network I/O operations and packet processing.

FD.io: A Universal Dataplane

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

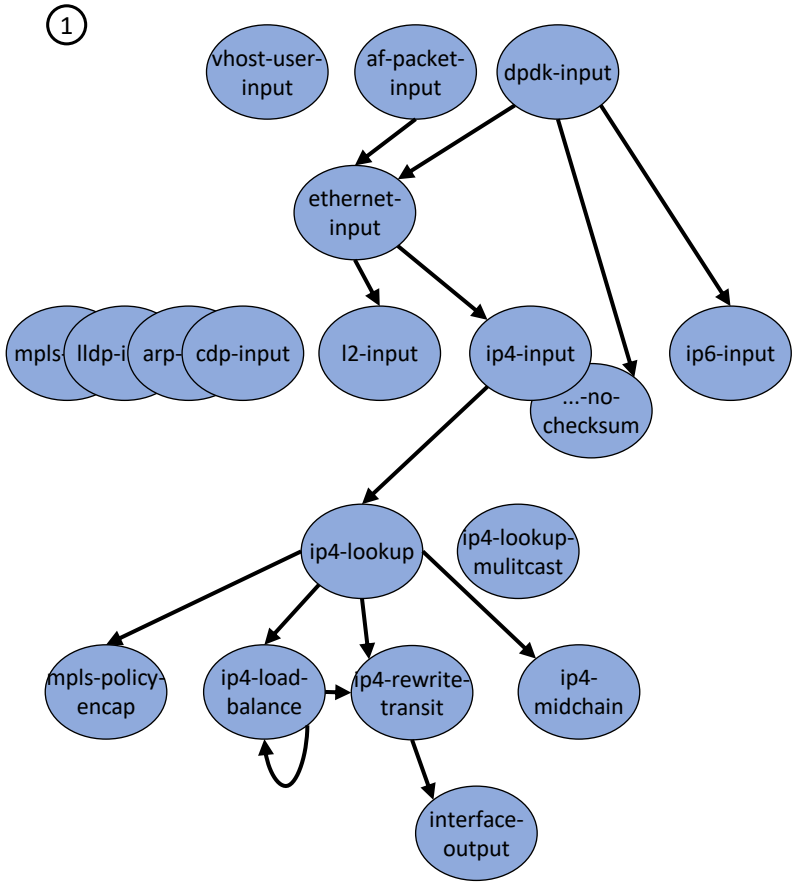
- Fd.io Scope:
  - **Network IO -** NIC/vNIC <-> cores/threads
  - **Packet Processing –** Classify/Transform/Prioritize/Forward/Terminate
  - **Dataplane Management Agents -** ControlPlane



Multiparty: Broad Contribution

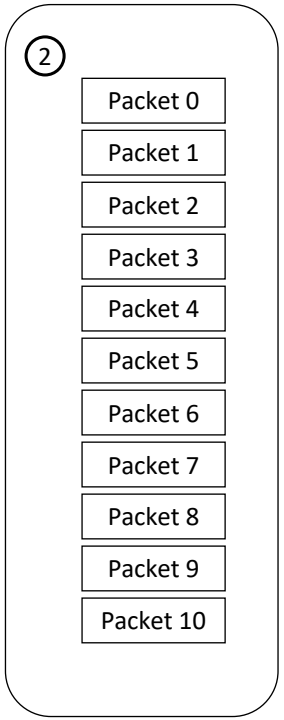
CISCO
intel
ERICSSON
Metaswitch Networks
IBM
6WIND
Я
Yandex
midokura
Qiniu
HUAWEI
COMCAST
QOSMOS
ubuntu
Supported by Canonical
Linaro
PROJECT CALICO
SUPPORTED BY METASWITCH
NXP
Allegro Packets
Network Software
Netgate
ZTE
UPC
KALRAY
SUSE
INOCYBE
OPENDAYLIGHT DISTRIBUTION
Universitat Politècnica de Catalunya (UPC)
redhat

VPP: How does it work?



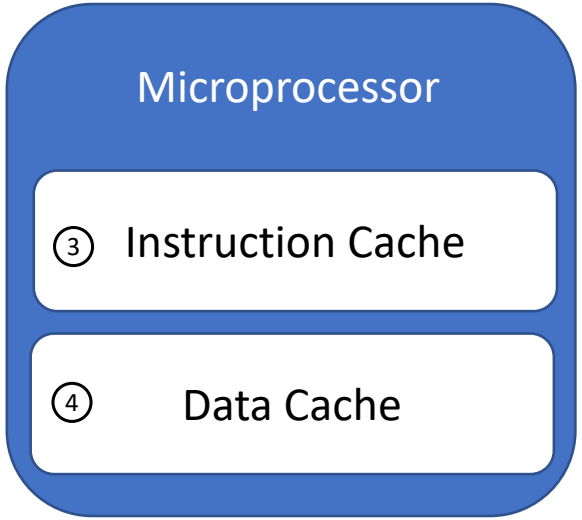
Packet processing is decomposed into a directed graph node …

* approx. 173 nodes in default deployment



… packets moved through graph nodes in vector …

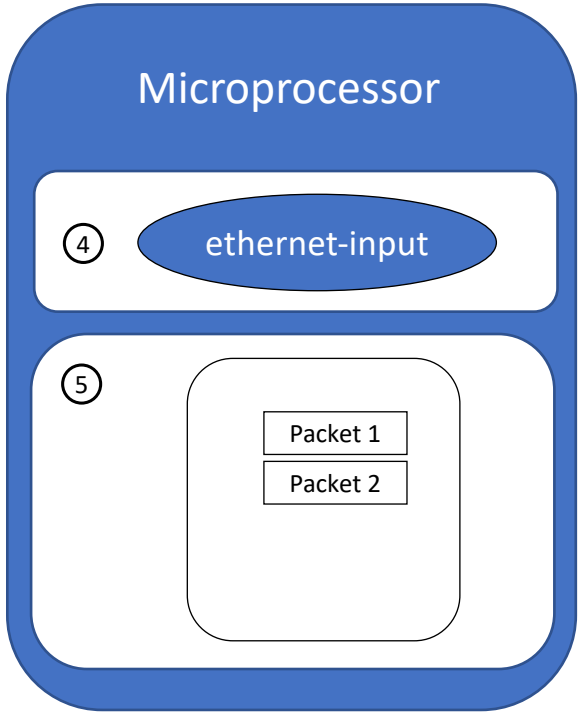
… graph nodes are optimized to fit inside the instruction cache …



… packets are pre-fetched, into the data cache …

VPP: How does it work? ⑥

… instruction cache is warm with the instructions from a single graph node …

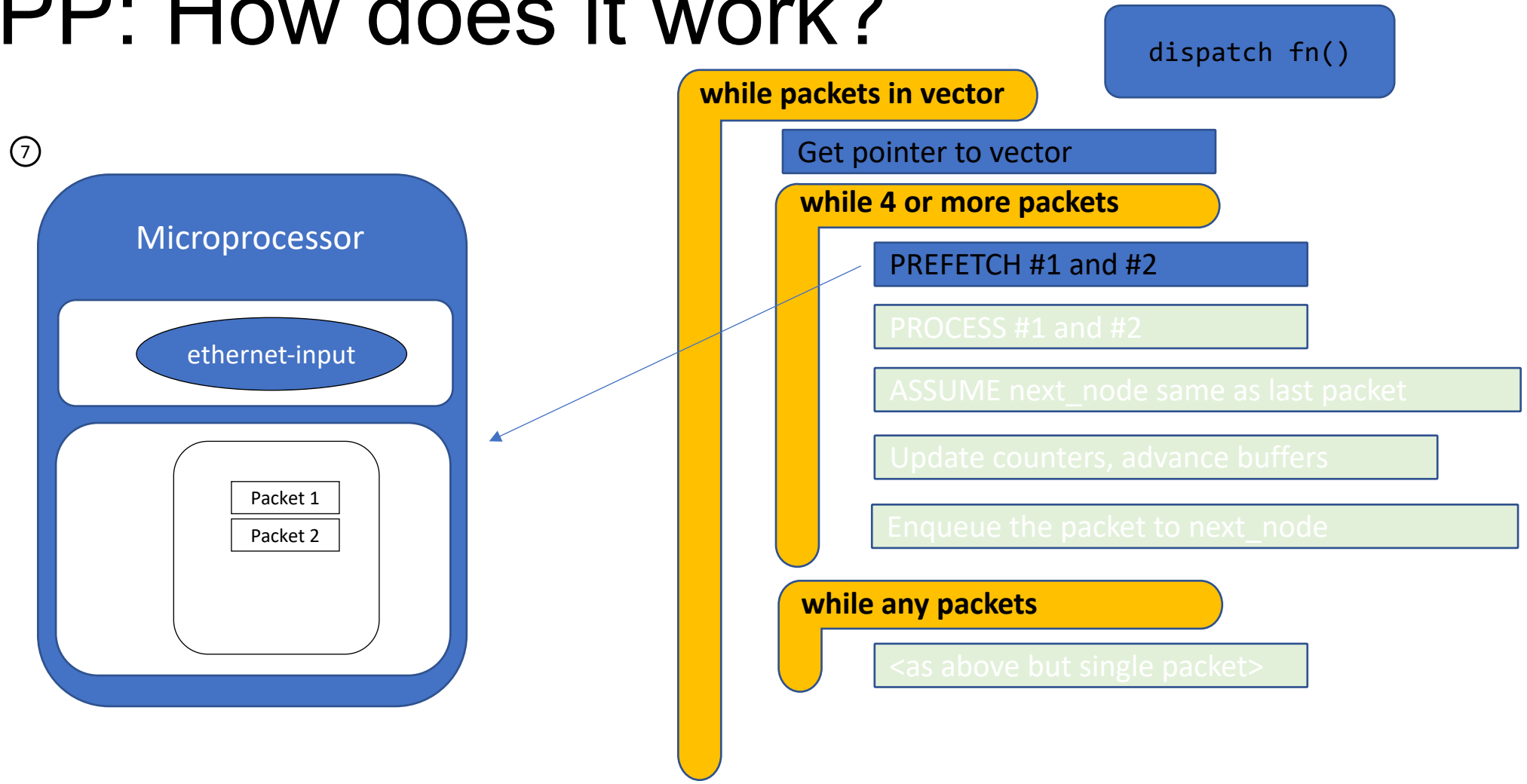


… data cache is warm with a small number of packets ..

```
dispatch fn()
while packets in vector
    Get pointer to vector
    while 4 or more packets
        PREFETCH #3 and #4
        PROCESS #1 and #2
        ASSUME next_node same as last packet
        Update counters, advance buffers
        Enqueue the packet to next_node
    while any packets
        <as above but single packet>
```

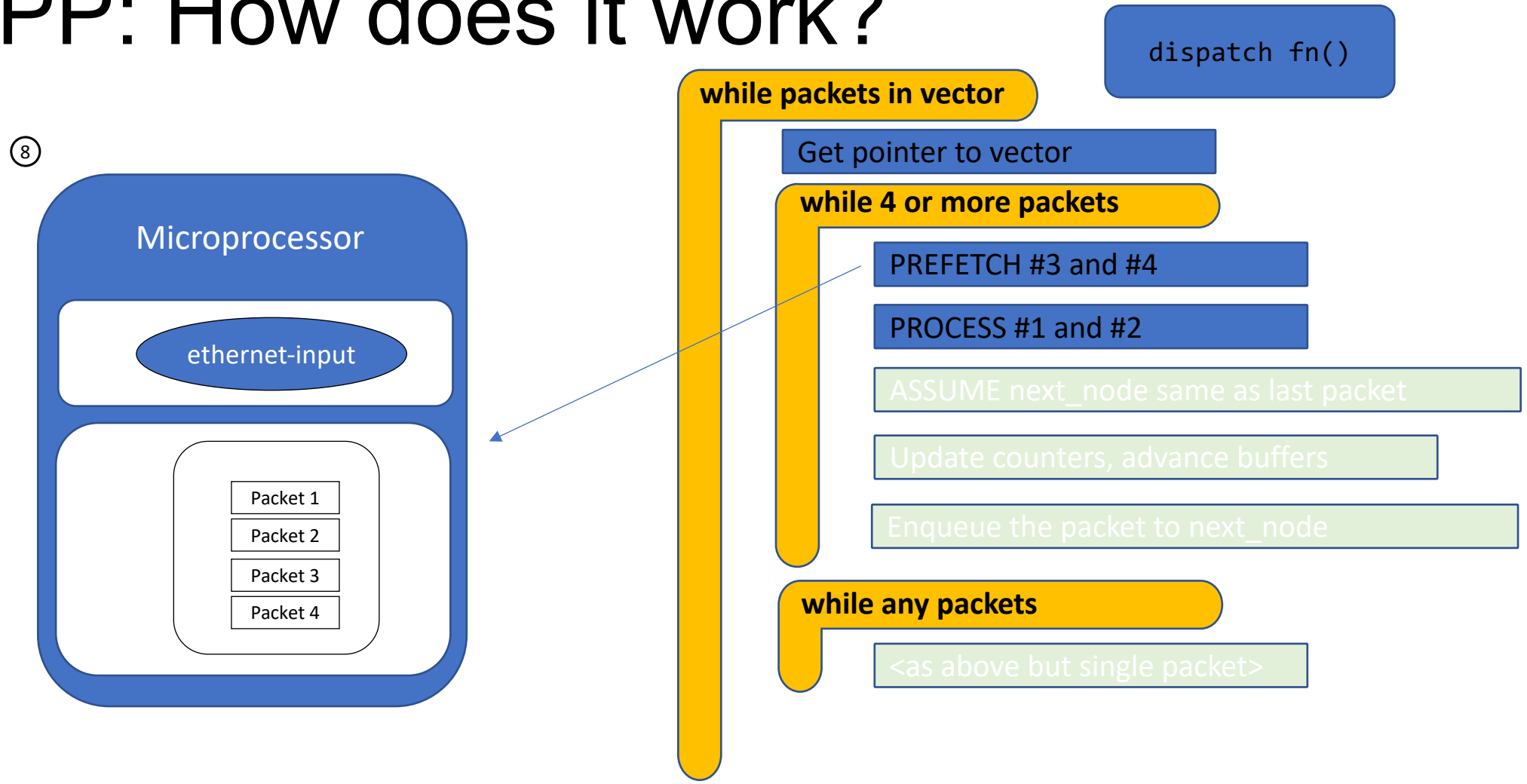
… packets are processed in groups of four, any remaining packets are processed on by one …

VPP: How does it work?



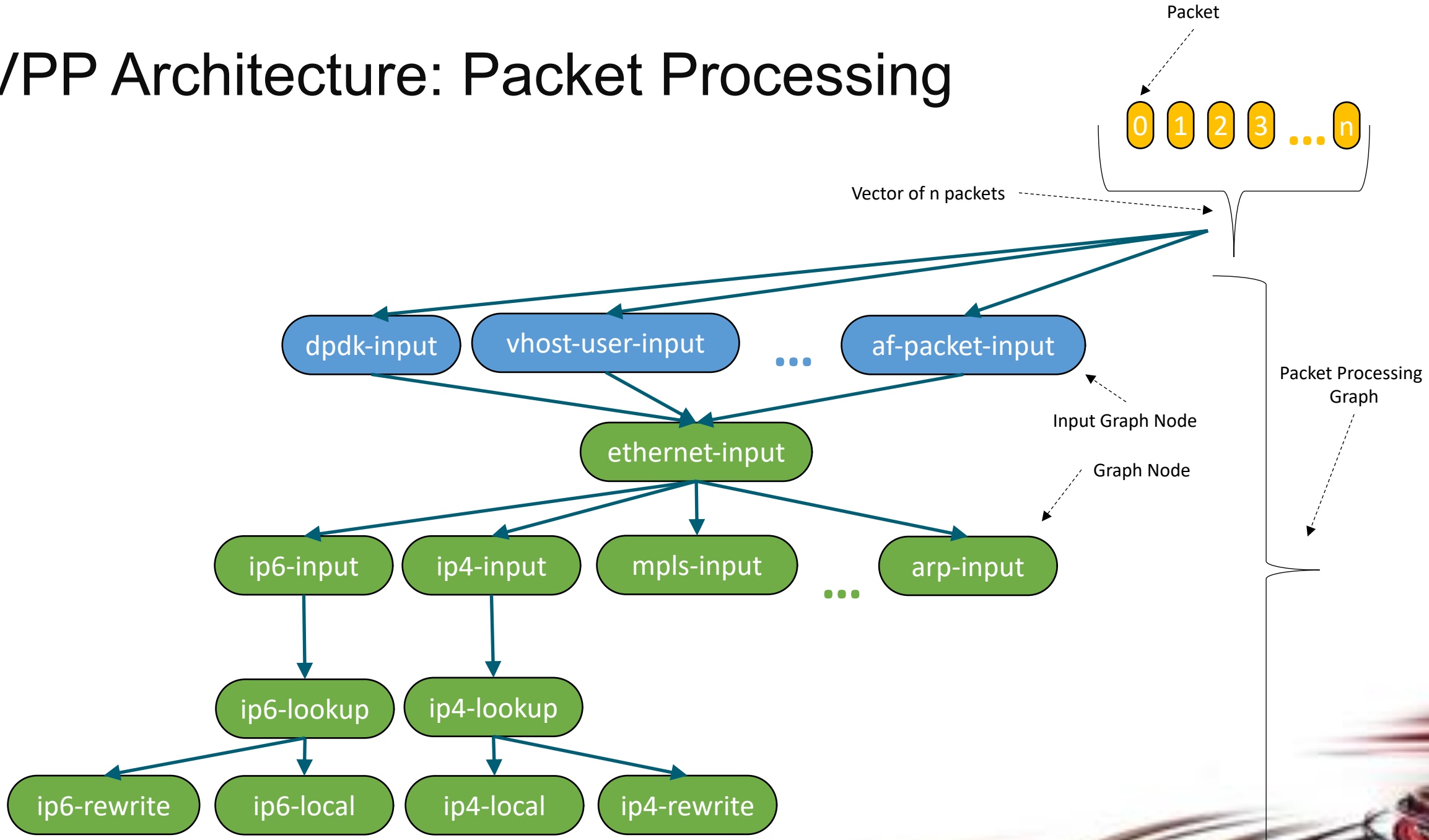
… prefetch packets #1 and #2 …

VPP: How does it work?

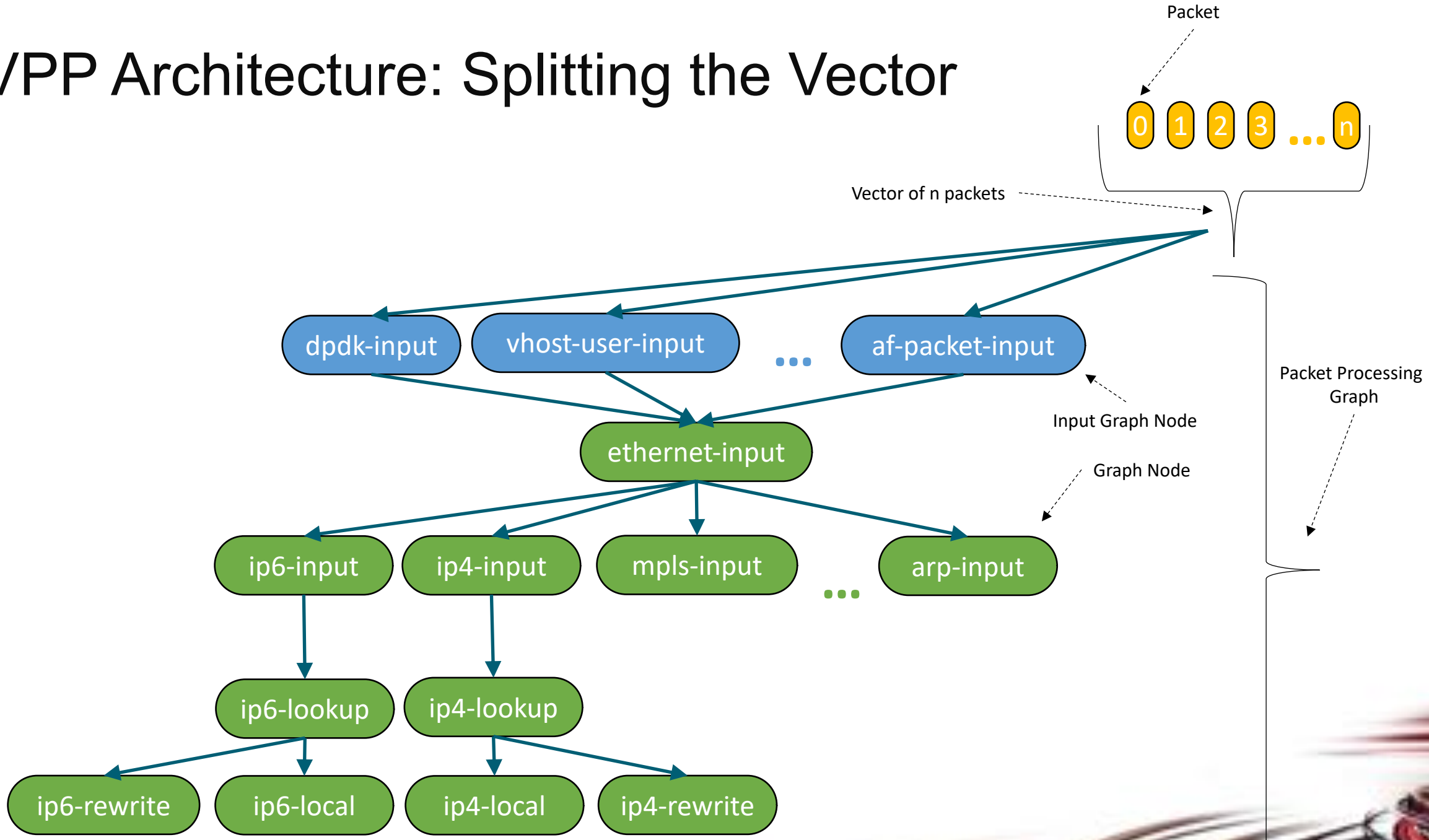


… process packet #3 and #4 …
… update counters, enqueue packets to the next node …

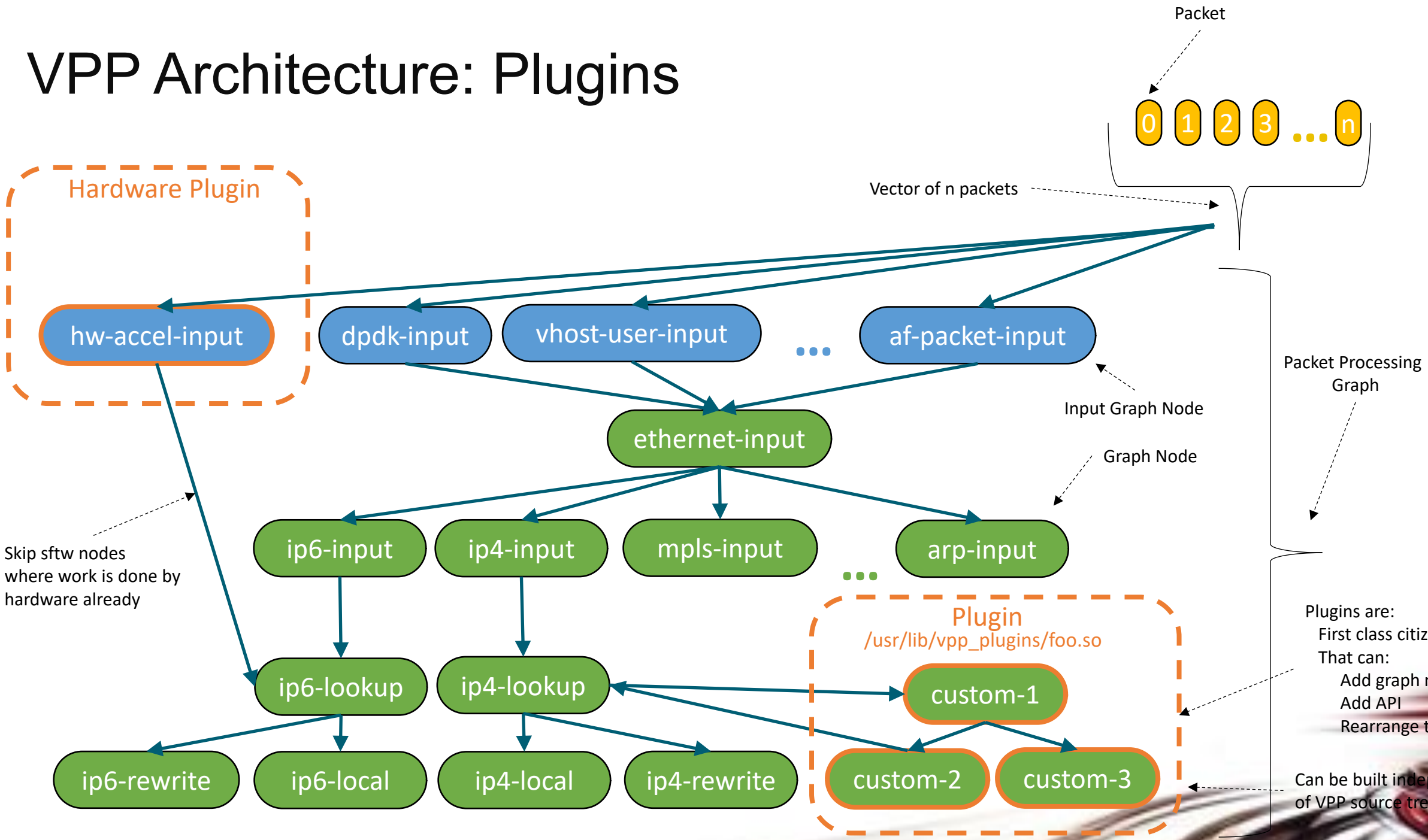
VPP Architecture: Packet Processing



VPP Architecture: Splitting the Vector



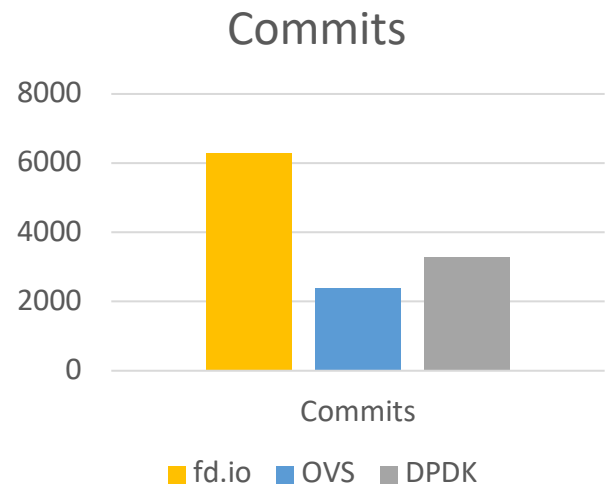
VPP Architecture: Plugins

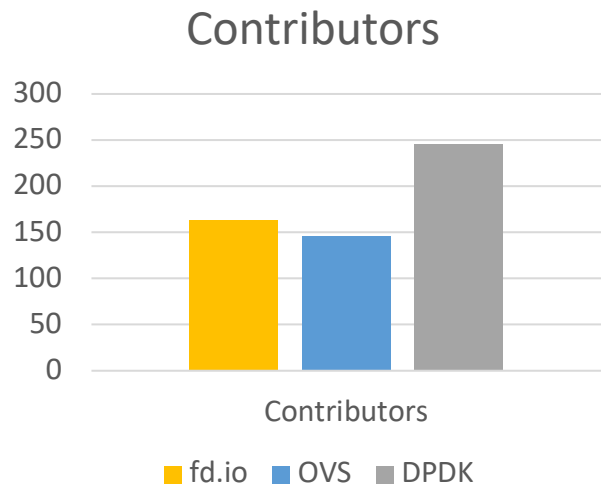


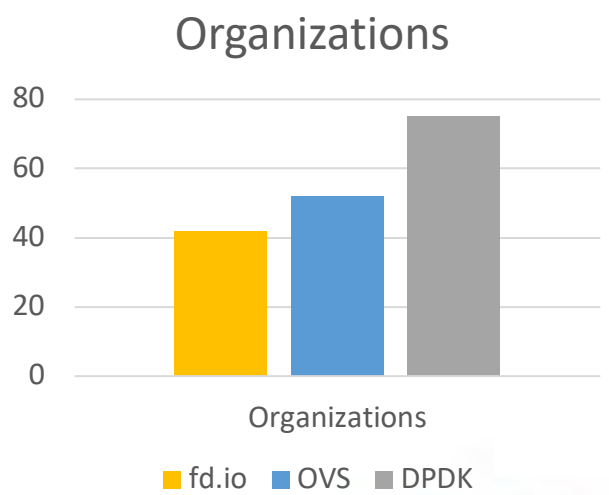
Code Activity

- In the period since its inception, fd.io has more commits than OVS and DPDK combined, and more contributors than OVS

2016-02-11 to 2017-04-03	Fd.io	OVS	DPDK
Commits	6283	2395	3289
Contributors	163	146	245
Organizations	42	52	78



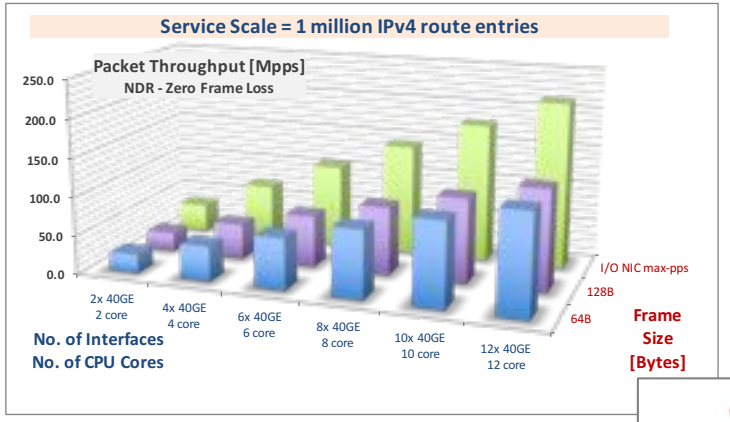


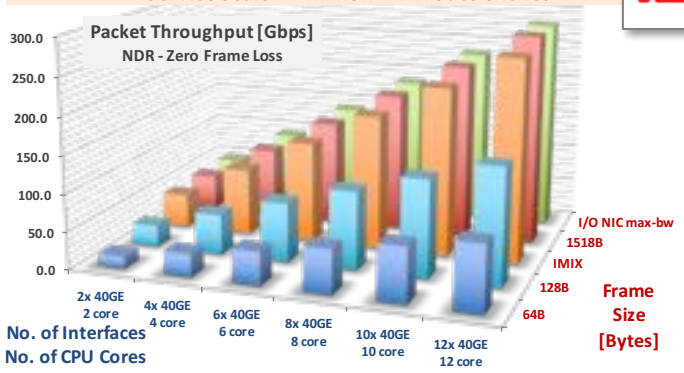


VPP Universal Fast Dataplane: Performance at Scale [1/2]

Per CPU core throughput with linear multi-thread(-core) scaling

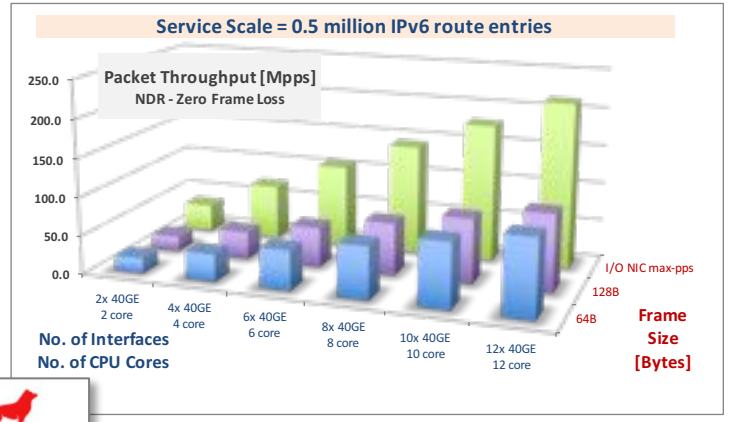
IPv4 Routing

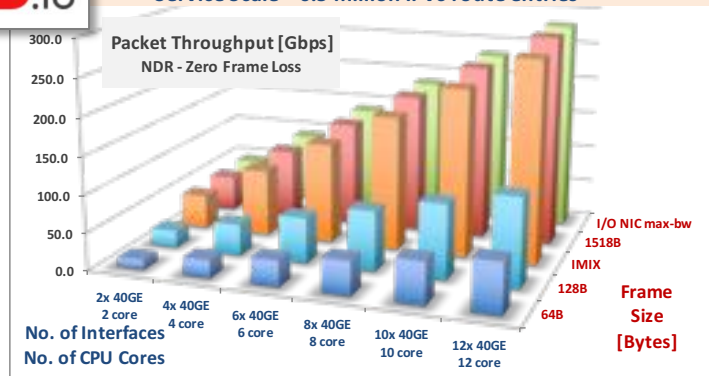




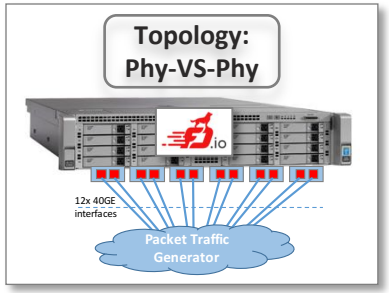
IPv4 Thput [Mpps]	2x 40GE 2 core	4x 40GE 4 core	6x 40GE 6 core	8x 40GE 8 core	10x 40GE 10 core	12x 40GE 12 core
64B	24.0	45.4	66.7	88.1	109.4	130.8
128B	24.0	45.4	66.7	88.1	109.4	130.8
IMIX	15.0	30.0	45.0	60.0	75.0	90.0
1518B	3.8	7.6	11.4	15.2	19.0	22.8
I/O NIC max-pps	35.8	71.6	107.4	143.2	179	214.8
NIC max-bw	46.8	93.5	140.3	187.0	233.8	280.5

IPv6 Routing





IPv6 Thput [Mpps]	2x 40GE 2 core	4x 40GE 4 core	6x 40GE 6 core	8x 40GE 8 core	10x 40GE 10 core	12x 40GE 12 core
64B	19.2	35.4	51.5	67.7	83.8	100.0
128B	19.2	35.4	51.5	67.7	83.8	100.0
IMIX	15.0	30.0	45.0	60.0	75.0	90.0
1518B	3.8	7.6	11.4	15.2	19.0	22.8
I/O NIC max-pps	35.8	71.6	107.4	143.2	179	214.8
NIC max-bw	46.8	93.5	140.3	187.0	233.8	280.5



Hardware: Cisco UCS C240 M4
Intel® C610 series chipset
2 x Intel® Xeon® Processor E5-2698 v3 (16 cores, 2.3GHz, 40MB Cache)
2133 MHz, 256 GB Total
6 x 2p40GE Intel XL710=12x40GE

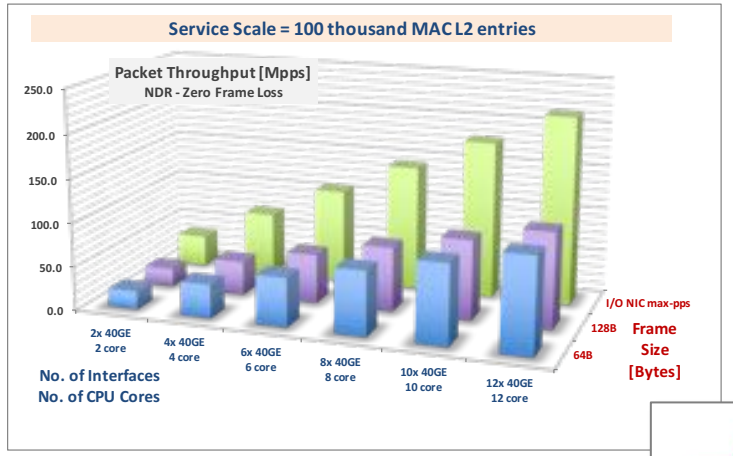
Software
Linux: Ubuntu 16.04.1 LTS **Kernel:** ver. 4.4.0-45-generic
FD.io VPP: VPP v17.01-5~ge234726 (DPDK 16.11)

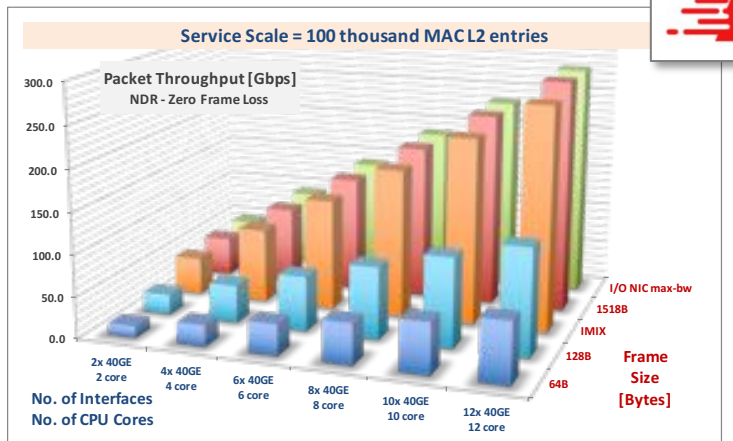
Resources
1 physical CPU core per 40GE port
Other CPU cores available for other services and other work
20 physical CPU cores available in 12x40GE seupt
Lots of Headroom for much more throughput and features

VPP Universal Fast Dataplane: Performance at Scale [2/2]

Per CPU core throughput with linear multi-thread(-core) scaling

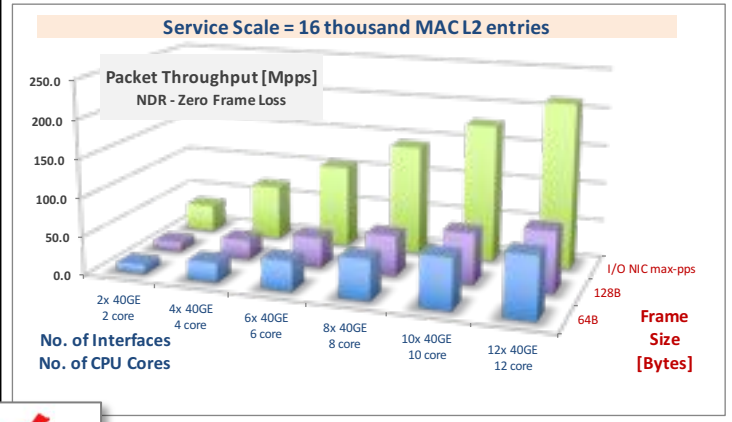
L2 Switching

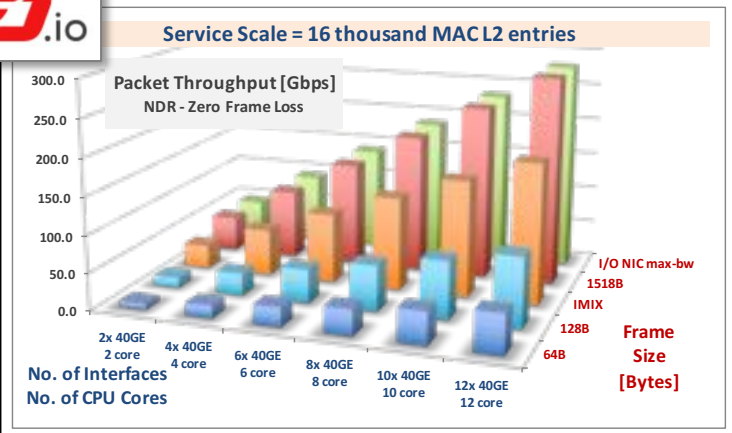




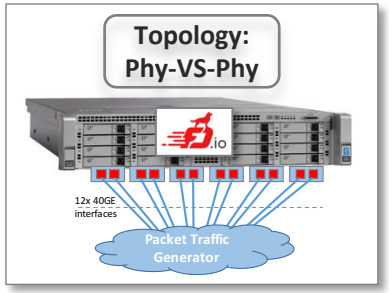
MAC Thput [Mpps]	2x 40GE 2 core	4x 40GE 4 core	6x 40GE 6 core	8x 40GE 8 core	10x 40GE 10 core	12x 40GE 12 core
64B	20.8	38.4	55.9	73.5	91.0	108.6
128B	20.8	38.4	55.9	73.5	91.0	108.6
IMIX	15.0	30.0	45.0	60.0	75.0	90.0
1518B	3.8	7.6	11.4	15.2	19.0	22.8
I/O NIC max-pps	35.8	71.6	107.4	143.2	179	214.8
NIC max-bw	46.8	93.5	140.3	187.0	233.8	280.5

L2 Switching with VXLAN Tunneling





MAC Thput [Mpps]	2x 40GE 2 core	4x 40GE 4 core	6x 40GE 6 core	8x 40GE 8 core	10x 40GE 10 core	12x 40GE 12 core
64B	11.6	25.1	38.6	52.2	65.7	79.2
128B	11.6	25.1	38.6	52.2	65.7	79.2
IMIX	10.5	21.0	31.5	42.0	52.5	63.0
1518B	3.8	7.6	11.4	15.2	19.0	22.8
I/O NIC max-pps	35.8	71.6	107.4	143.2	179	214.8
NIC max-bw	46.8	93.5	140.3	187.0	233.8	280.5



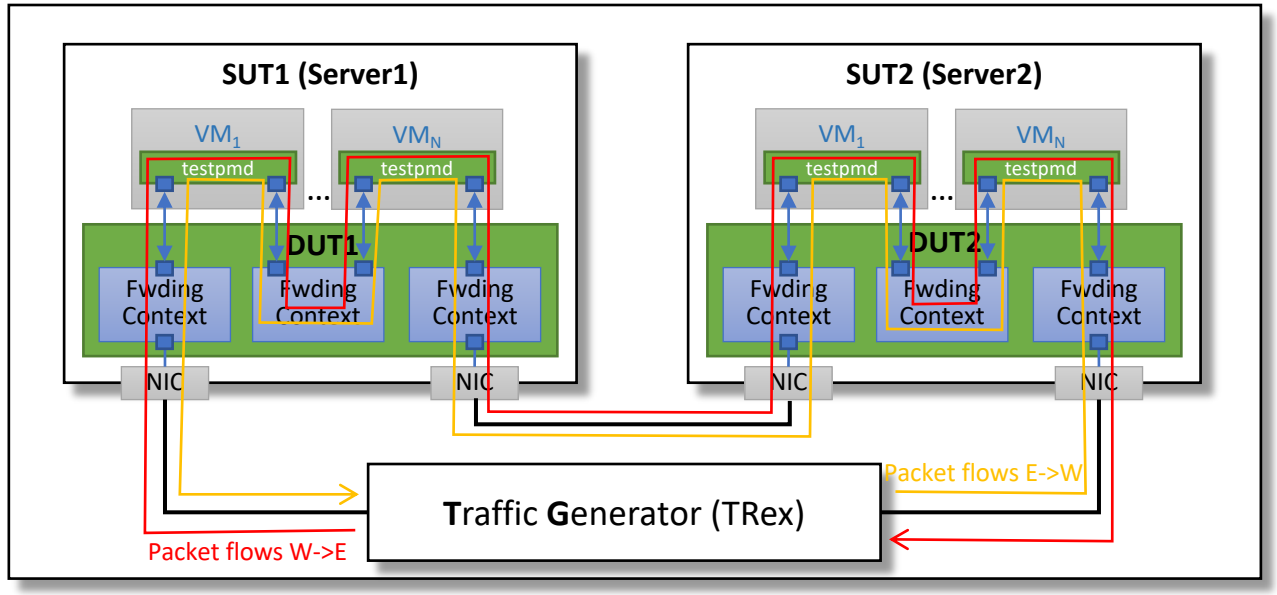
Hardware: Cisco UCS C240 M4
Intel® C610 series chipset
2 x Intel® Xeon® Processor E5-2698 v3 (16 cores, 2.3GHz, 40MB Cache)
2133 MHz, 256 GB Total
6 x 2p40GE Intel XL710=12x40GE

Software
Linux: Ubuntu 16.04.1 LTS
Kernel: ver. 4.4.0-45-generic
FD.io VPP: VPP v17.01-5~ge234726 (DPDK 16.11)

Resources
1 physical CPU core per 40GE port
Other CPU cores available for other services and other work
20 physical CPU cores available in 12x40GE seupt
Lots of Headroom for much more throughput and features

VPP VM-to-VM vhostuser

Per CPU core throughput performance – CSIT rls1704

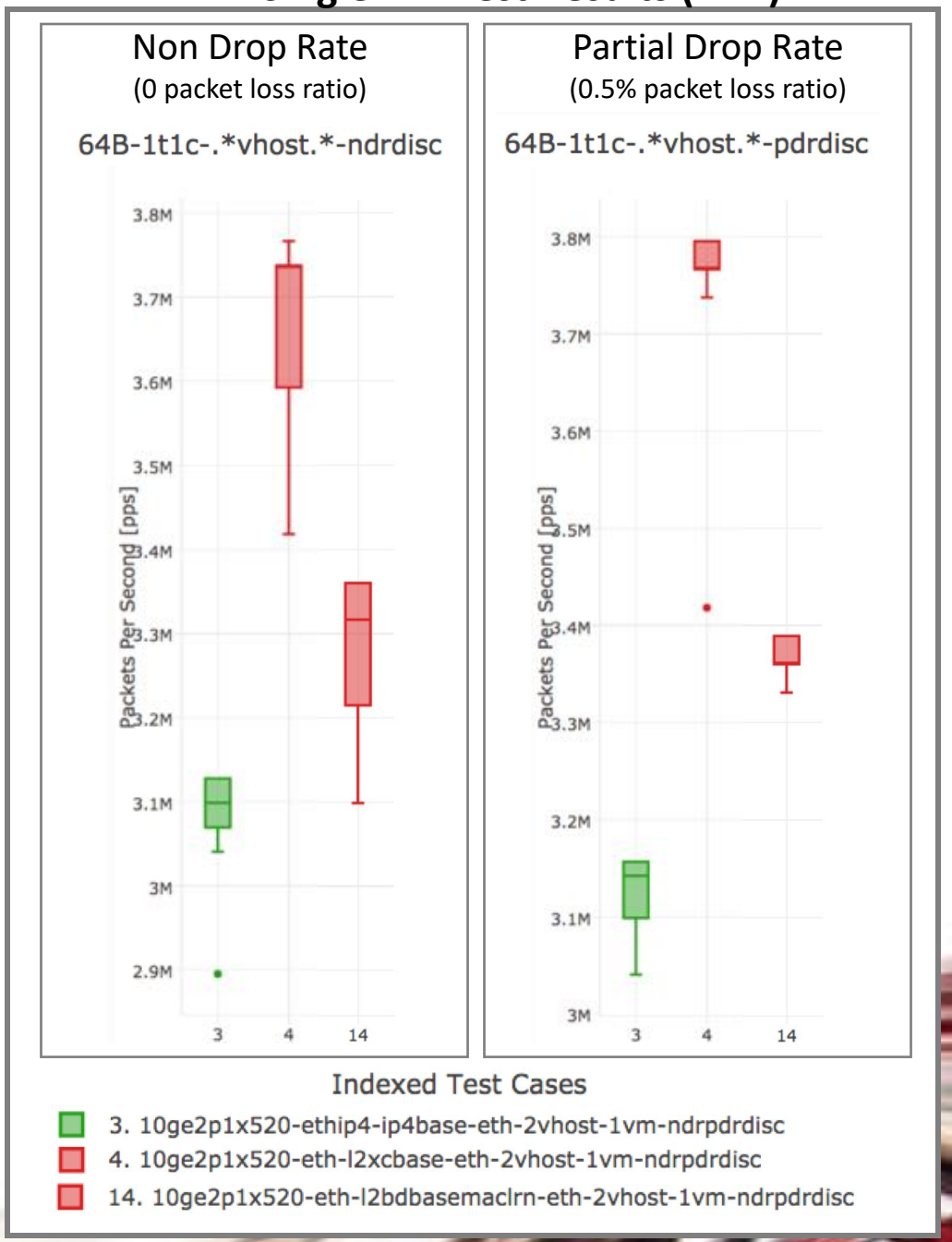


- For VM tests, packets are switched by DUT (VPP) multiple times
  - twice for a single VM, three times for two VMs in chain;
  - external throughput rates measured by TG and listed in CSIT report must be multiplied by (N+1) to represent the actual DUT aggregate packet forwarding rate, N stands for number of VMs;
- CSIT rls1704 reported throughput for VPP vhostuser:

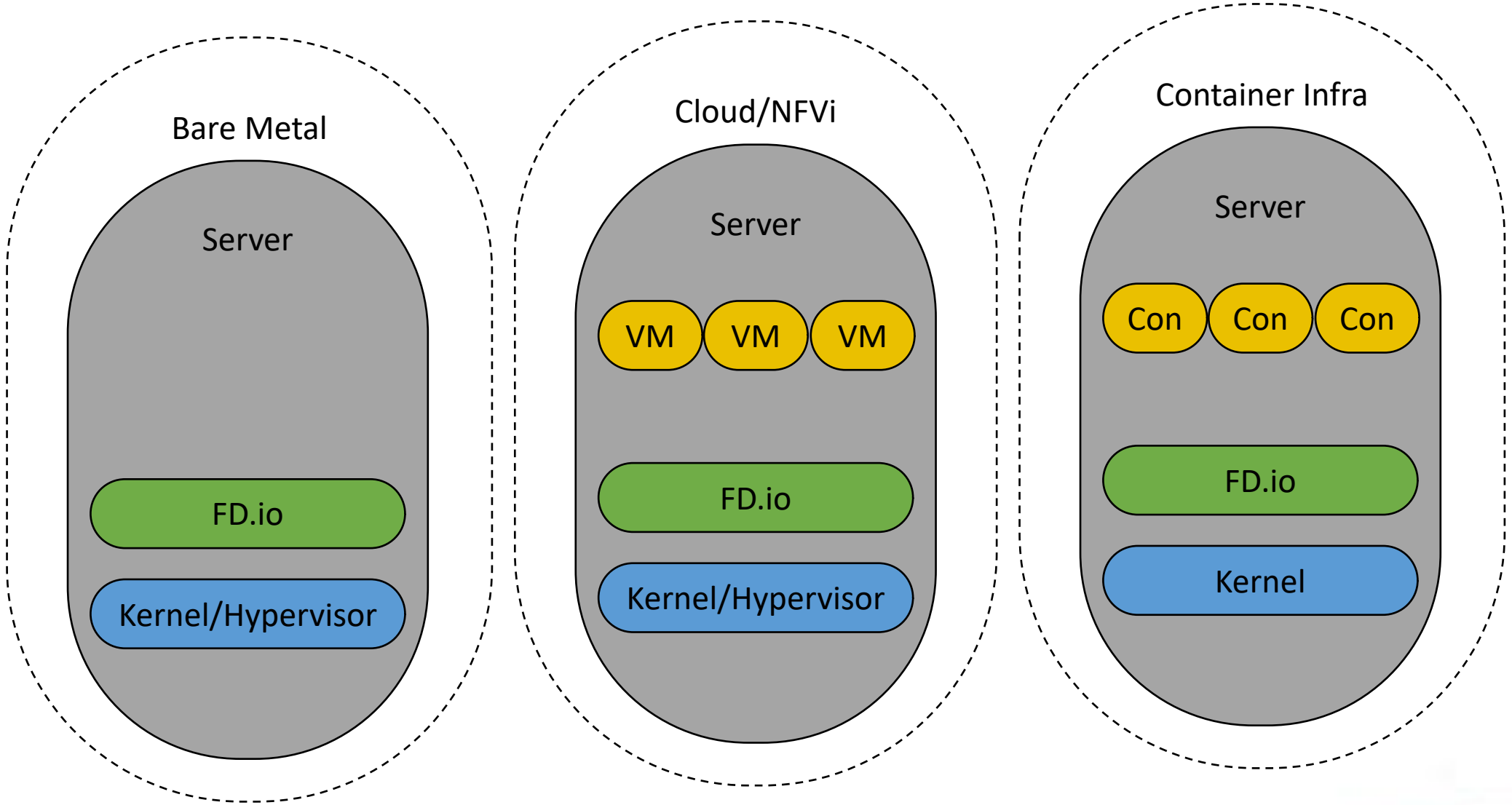
Fwding Context Type	NDR [Mpps]	PDR [Mpps]
IPv4 routing (ip4base)	**6.1** (2*3.05)	**6.1** (2*3.05)
L2 crossconnect (l2xcbase)	**6.8** (2*3.4)	**7.5** (2*3.75)
L2 bridging (l2bdbasemaclrn)	**6.2** (2*3.1)	**6.7** (2*3.35)

Source: https://docs.fd.io/csit/rls1704/report/vpp_performance_tests/overview.html
https://docs.fd.io/csit/rls1704/report/vpp_performance_tests/packet_throughput_graphs/vm_vhost.html

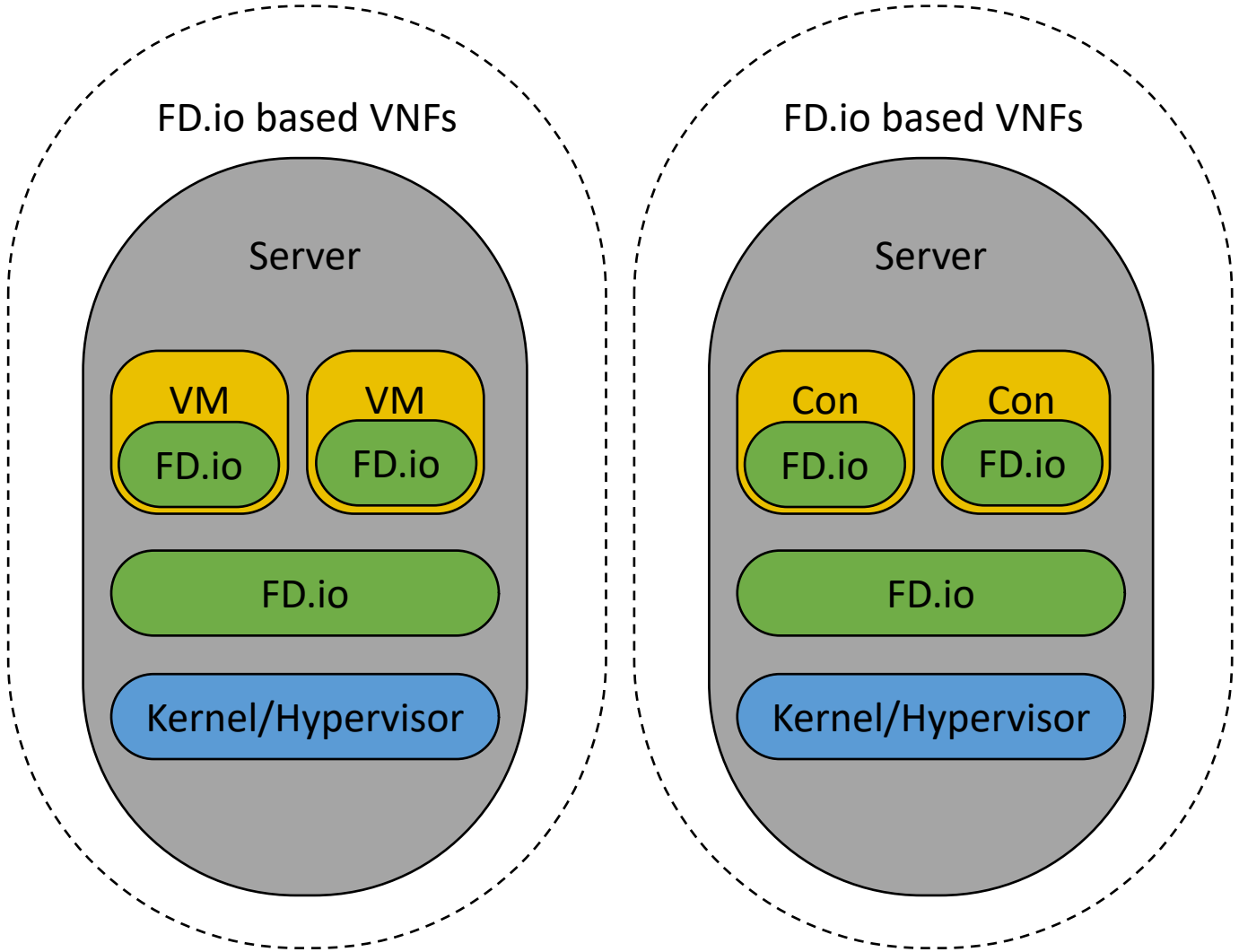
Single VM Test Results (N=1)



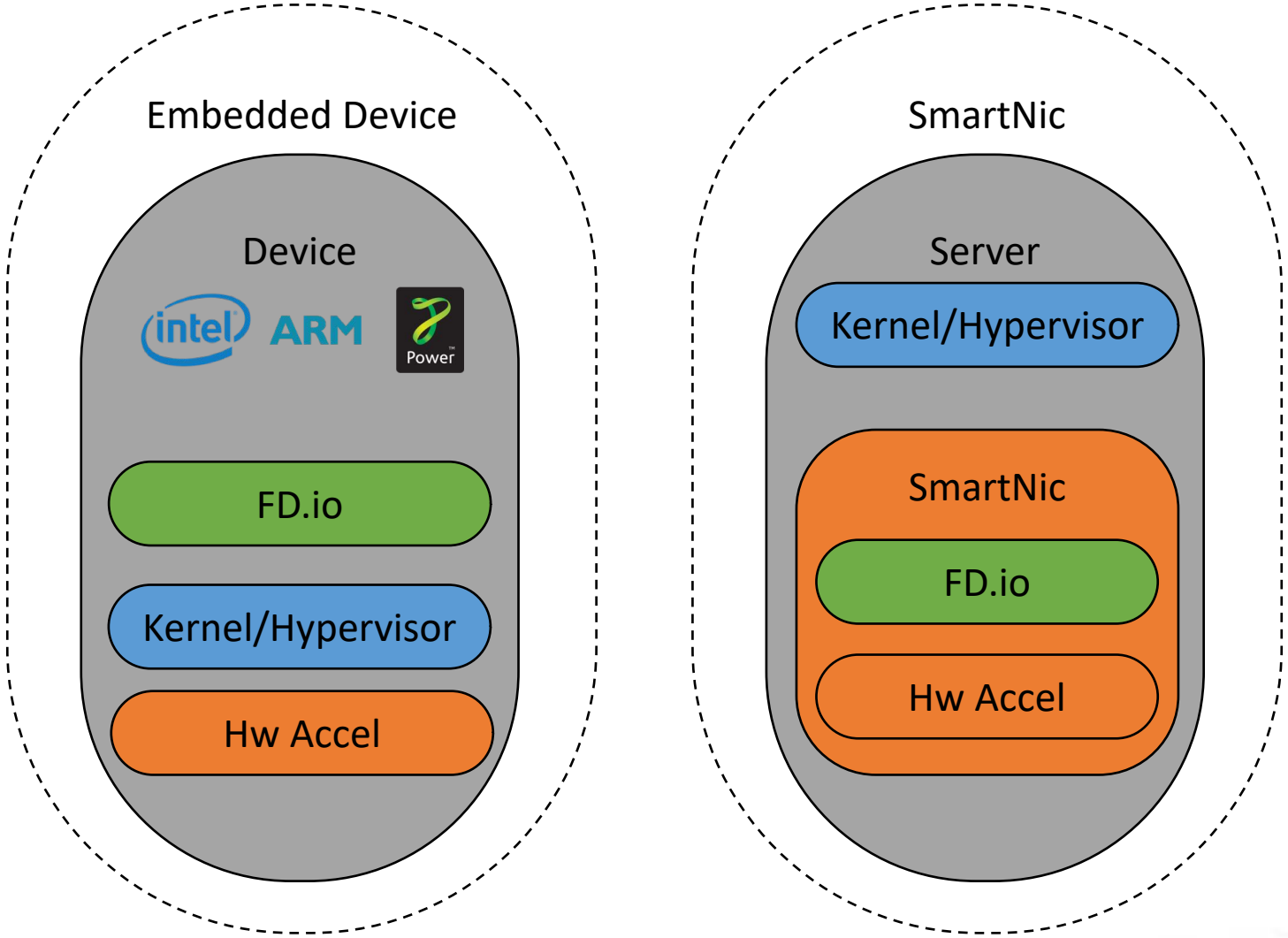
Universal Dataplane: Infrastructure



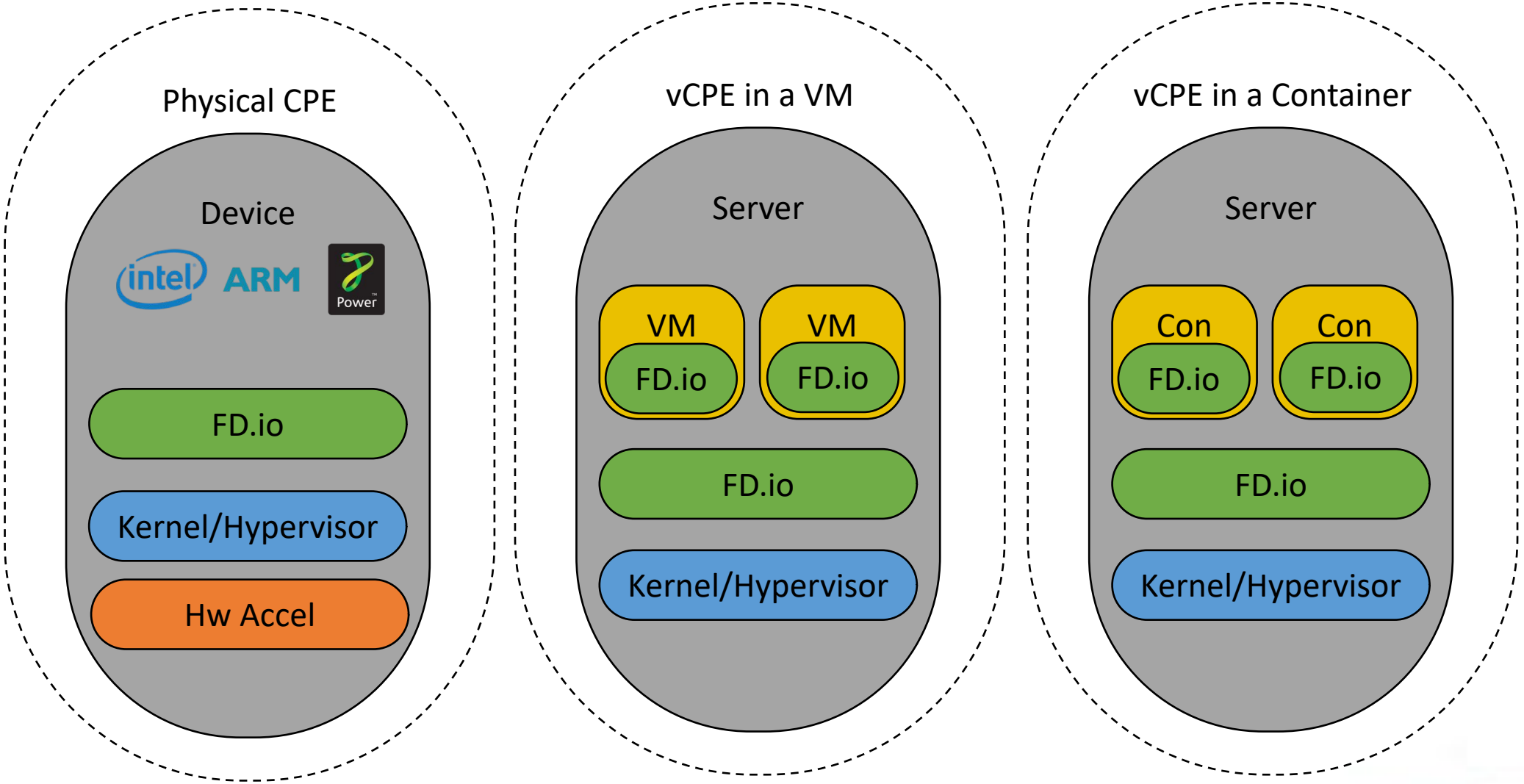
Universal Dataplane: virtual NFs



Universal Dataplane: Embedded

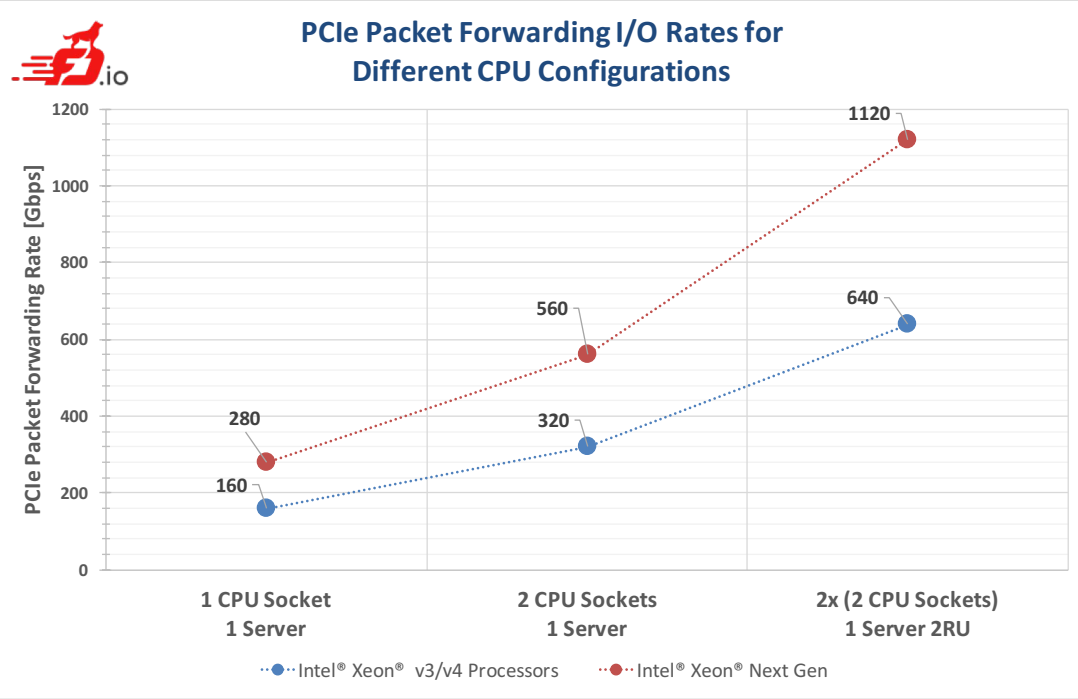


Universal Dataplane: CPE Example



Scaling Up The Packet Throughput with FD.io VPP

Can we squeeze more from a single 2RU server ?



1. **Today's Intel® XEON® CPUs (E5 v3/v4):**
   a. Per socket have 40 lanes of PCIe Gen3
   b. 2x 160Gbps of packet I/O per socket

2. **Tomorrow's Intel® XEON® CPUs:**
   a. Per socket support More lanes of PCIe Gen3
   b. 2x 280Gbps of packet I/O per socket

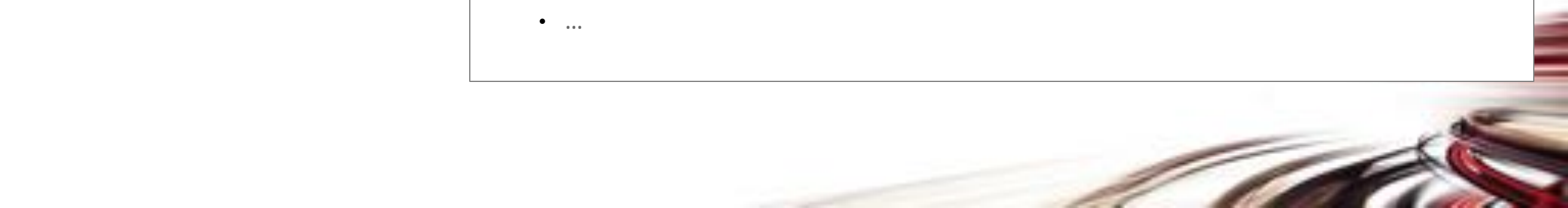
VPP enables linear multi-thread(-core) scaling
up to the packet I/O limit per CPU => on a path to **one terabit software router** (**1TFR**).

So FD.io VPP is great, what can You do with it?

Slide from Coseners 2016 ..

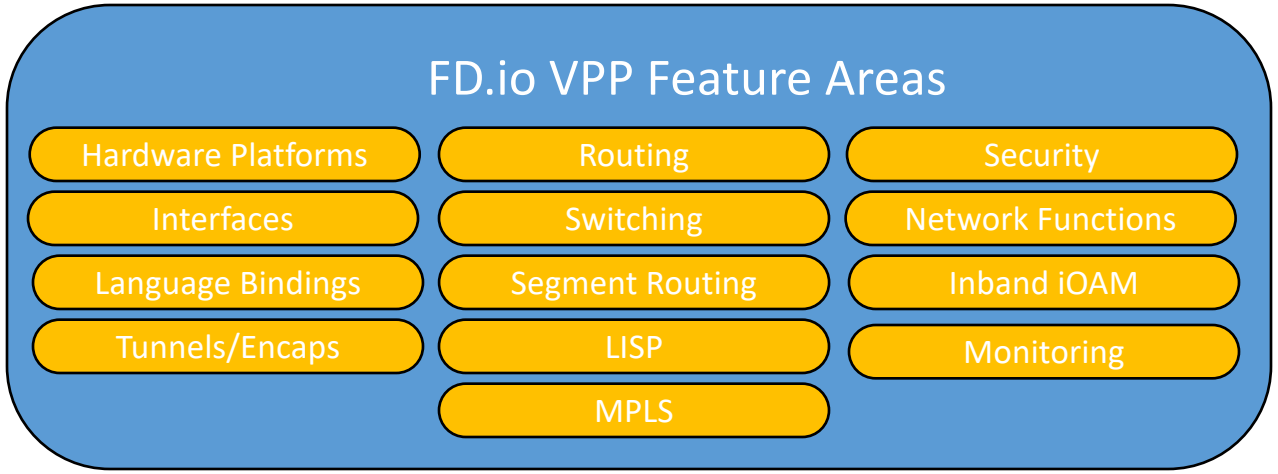
FD.io VPP – A Platform for Interesting Work …

- Modern IP router data plane out-of-the-box
  - Advanced, modular, scales, optimized SW-HW interface
  - A platform to build on
- All sorts of crypto and tunneling things
- ILA at IETF96 Hackathon in Berlin
  - ILA in XDP and VPP, https://tools.ietf.org/html/draft-herbert-nvo3-ila-02
- Telemetry – apps, users, flows, …
- Modern TCP stack anyone?
  - E.g. TCP ex-machina, Keith Winstein https://github.com/tcpexmachina/remy
- …



So FD.io VPP is great, what can You do with it?

- SW NFV platform - Make any NFs you can dream of
- Modular architecture with graph nodes
- Plugin architecture enables extending the system without touching main repository code
- Experiment, research, develop
  - data structures: cuckoo hash, poptrie, bloom filters
  - lock-free stateful stores
  - …



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