



ons

EUROPE

OPEN NETWORKING //
Integrate, Automate, Accelerate

September 25 - 27, 2018
Amsterdam, The Netherlands

High Performance Cloud-native Networking

K8s Unleashing FD.io

 **Giles Heron**

Principal Engineer, Cisco
giheron@cisco.com

 **Maciek Konstantynowicz**

FD.io CSIT Project Lead
Distinguished Engineer, Cisco
mkonstan@cisco.com

 **Jerome Tollet**

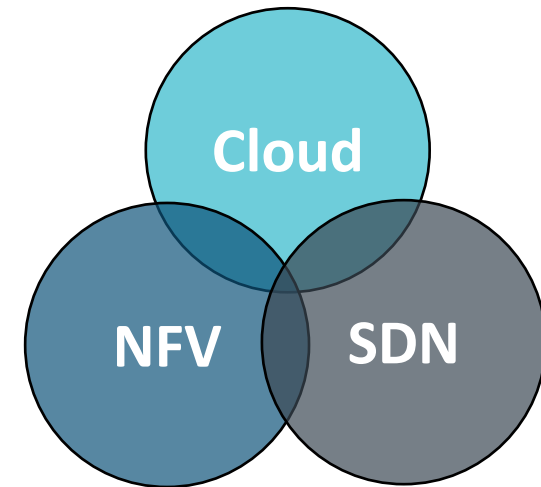
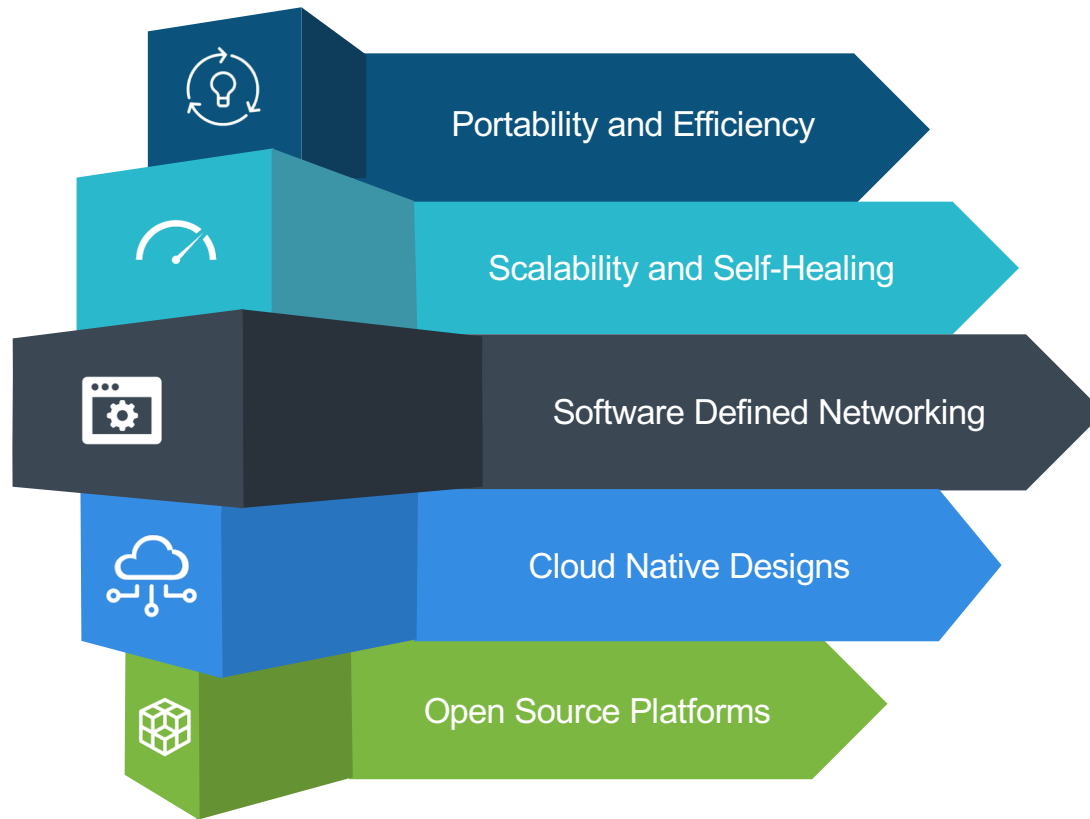
Distinguished Engineer, Cisco
jtollet@cisco.com

DISCLAIMERs

- **'Mileage May Vary'**
 - Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your opinion and investment of any resources. For more complete information about open source performance and benchmark results referred in this material, visit <https://wiki.fd.io/view/CSIT> and/or <https://docs.fd.io/csit/rls1807/report/>.
- **Trademarks and Branding**
 - This is an open-source material. Commercial names and brands may be claimed as the property of others.



Internet Mega Trends



5 Pillars of Next Generation Software Data Planes

Blazingly Fast



- Process the *massive explosion* of East-West traffic
- Process *increasing* North-South traffic

Software First



- Cloud means *running everywhere*
- Cloud means hardware and physical *infra agnostic*

Truly Extensible



- Foster *pace of innovation* in cloud-native networking
- *No compromise* on performance (zero-tolerance)

Predictable performance



- Dataplane performance must be deterministic
- Predictable for a number of VMs, Containers, virtual topology and (E-W, N-S) traffic matrix

Measureable



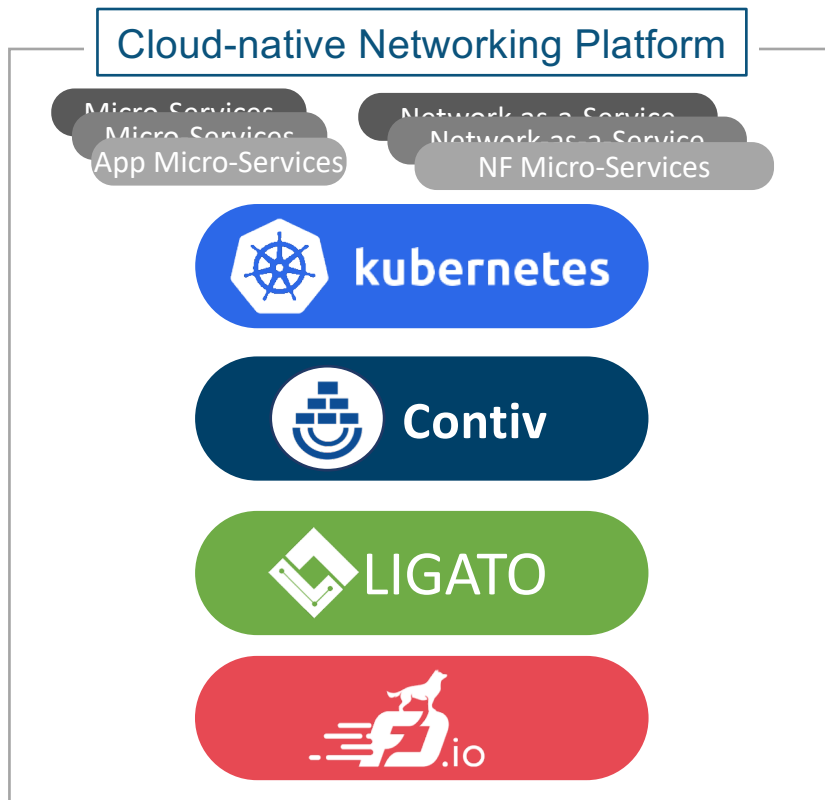
- Counters everywhere to *count everything* for detailed cross-layer operation and efficiency monitoring
- Enables feedback loop to drive optimizations

FD.io VPP meets these challenges

How can one use it in large scale Cloud-native networks ?

High Performance Cloud-native Networking

K8s Unleashing FD.io



Production-Grade Container Orchestration

- De facto standard for container orchestration
- Superior extensibility and self-healing



Performance Container Networking

- Seamless cloud-native integration
- Extends K8s with userspace networking



Cloud-native Network Micro-services

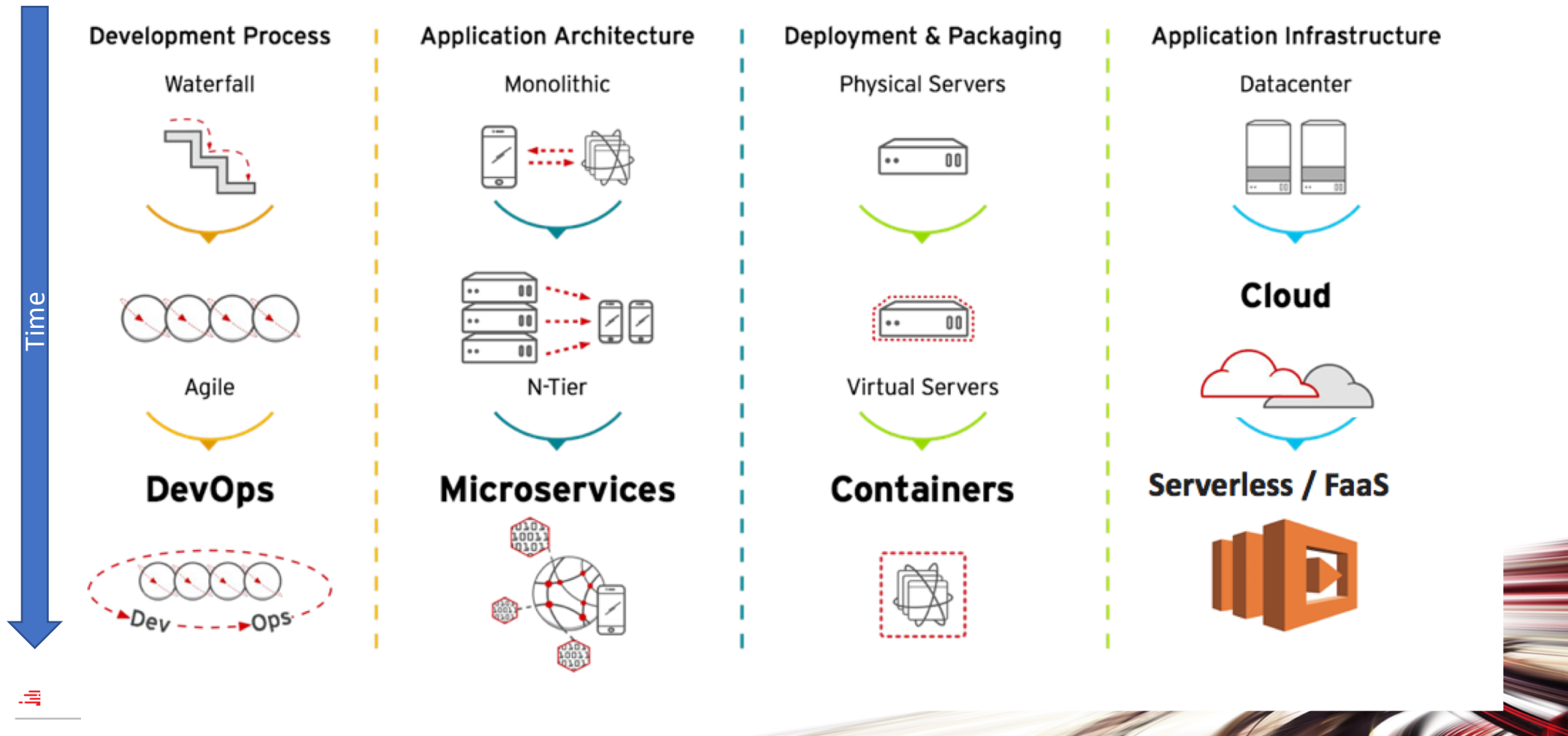
- Scalability, performance and agility
- Marries K8s with flexible network topologies



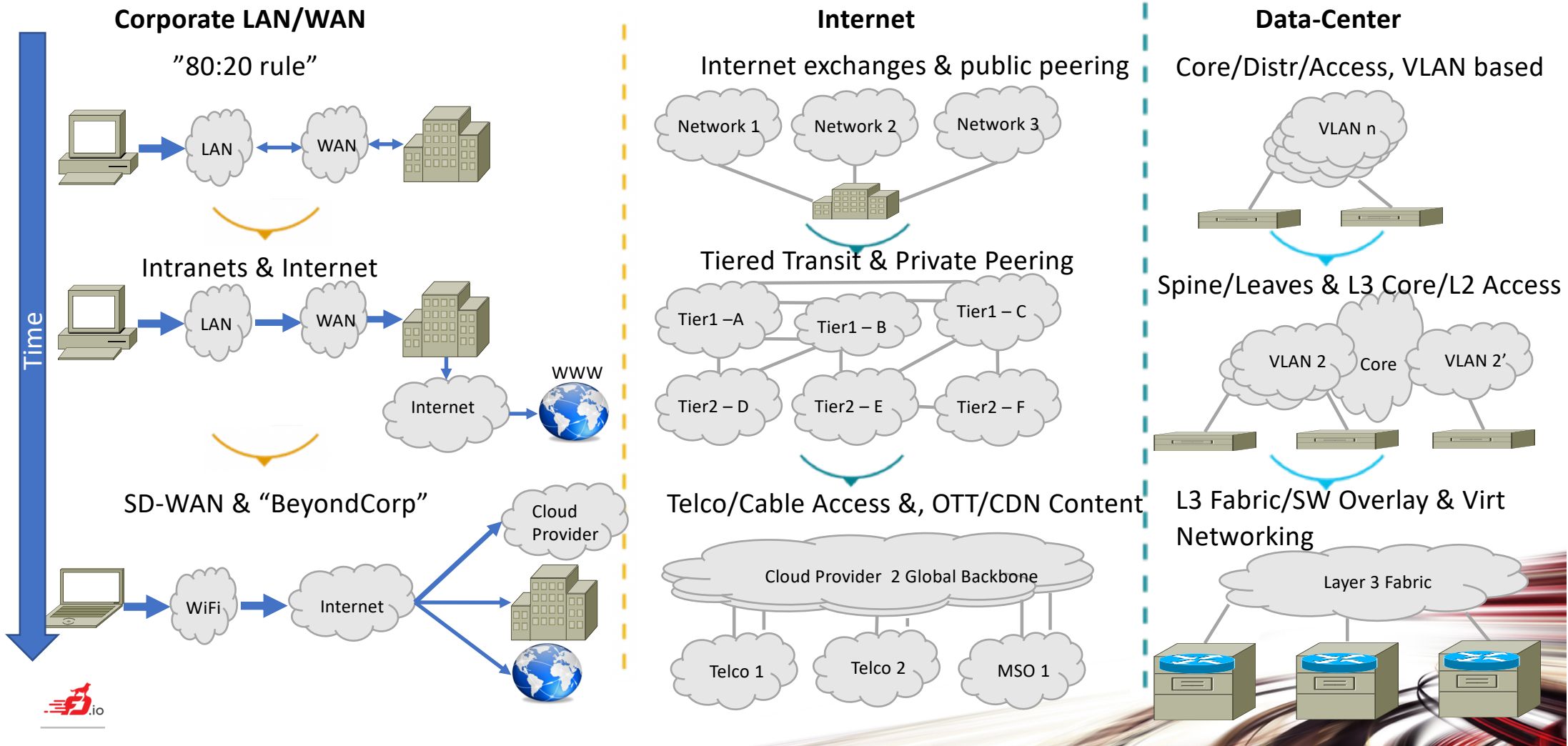
Fast Data Input/Output

- Most efficient on the planet
- Top performance, flexibility and extensibility

The Way Applications Are Developed and Deployed.. Has Changed..

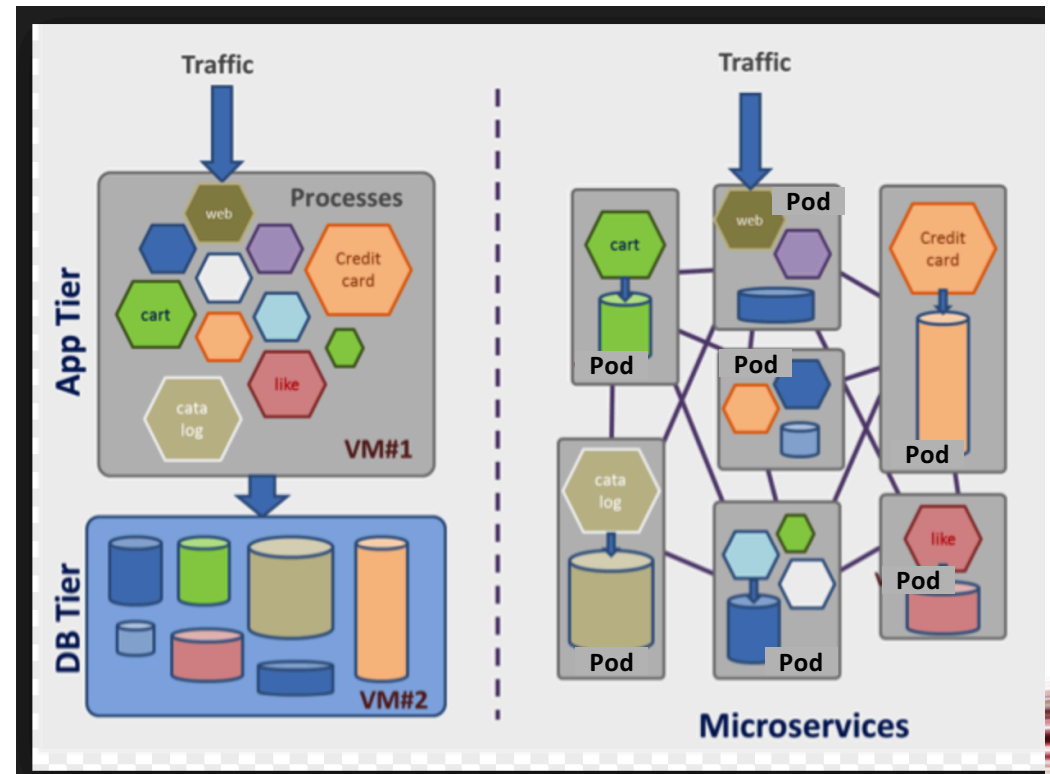


The Way Networks are Deployed and Used... has Changed...



Microservices & Containers have changed many things...

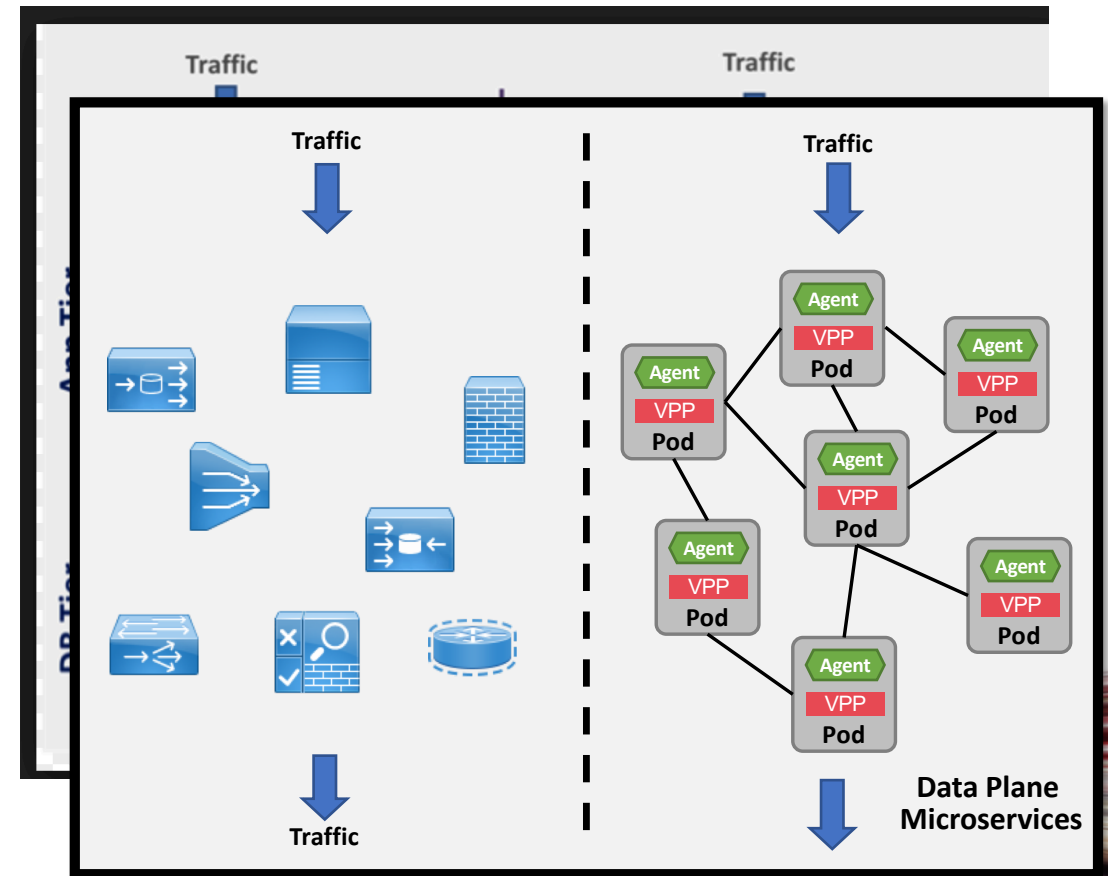
- Applications are being developed and deployed very differently today.
- Microservices split applications into modular pieces with the network stitching the pieces together
- The interconnection of the pieces increases "East / West" traffic
- Network performance is therefore critical to the composite application performance



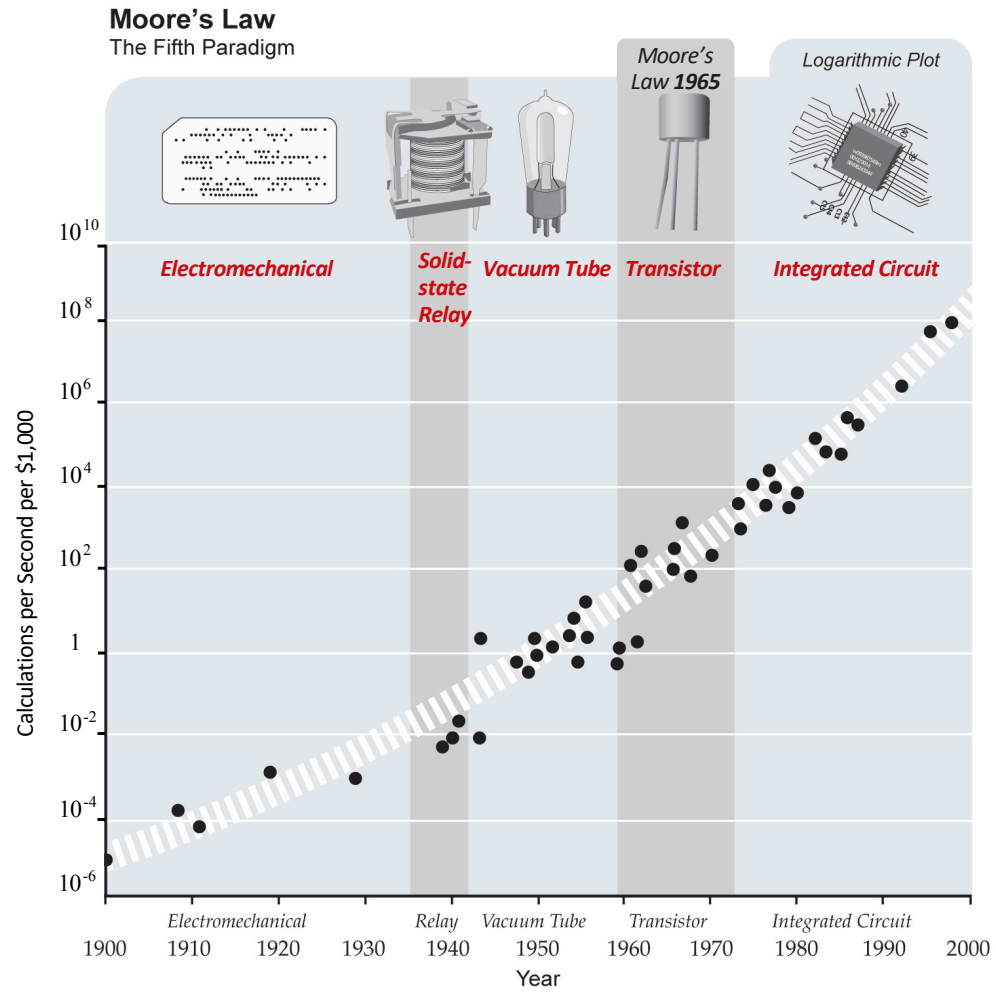
Microservices & Containers have changed many things...

- How do application trends impact SDN / NFV?

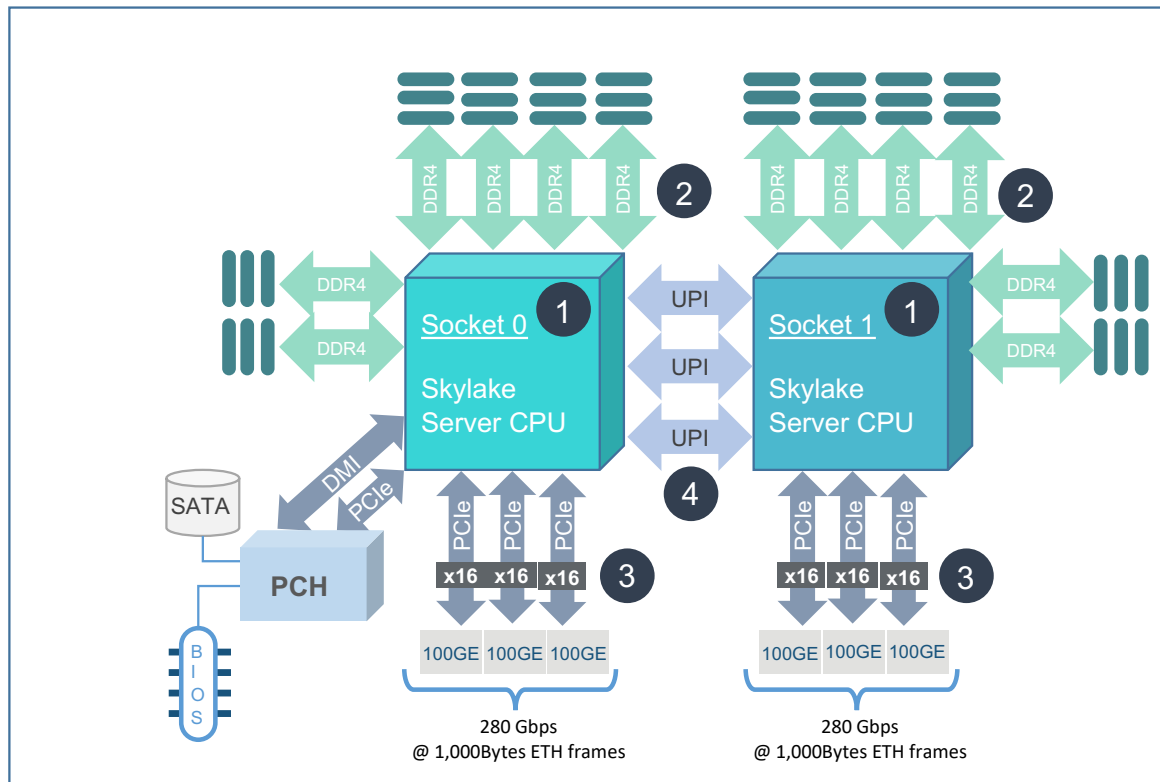
- Network Functions can be implemented using containers
- The microservice model enables decomposition of Network Functions into finer-grained Cloud-native Network Functions (CNFs)
- Since network functions are I/O bound network performance is even more critical than for applications



Remember 1965 "Moore's Law" – ..



Processing Packets: How to Use Compute ..



Resources to Get Performance

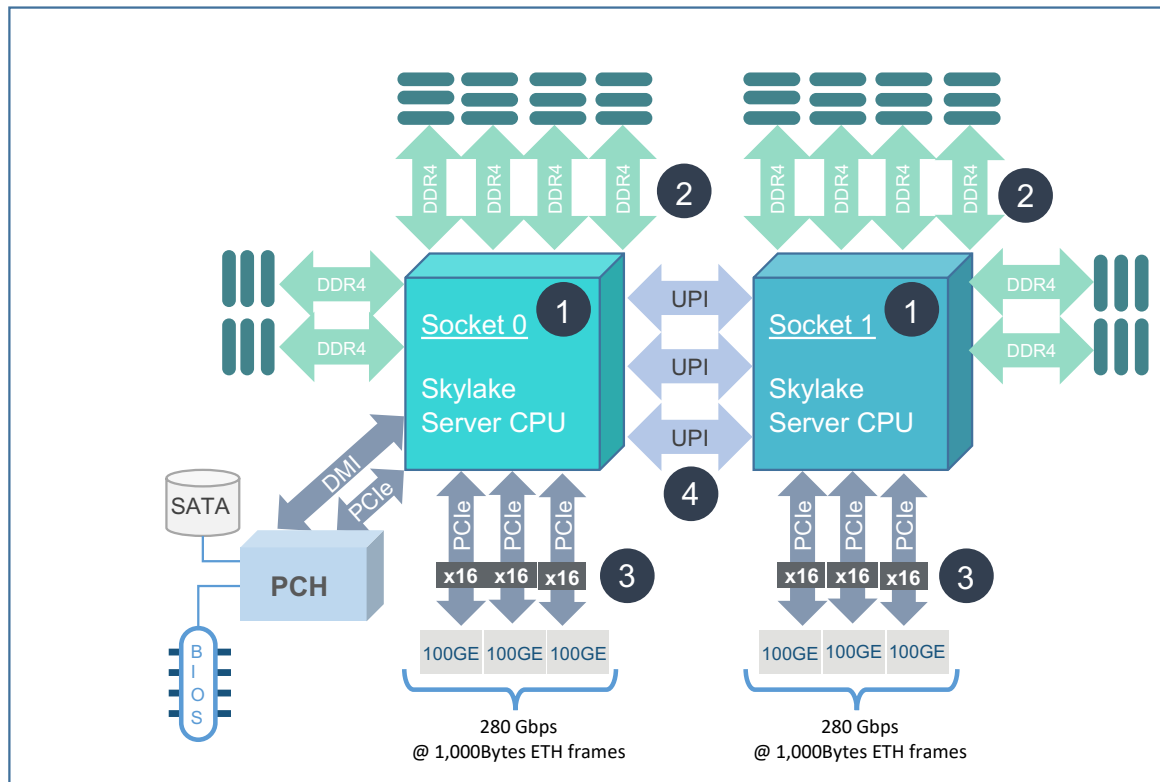
- 1 **Processor and CPU cores**
for performing packet processing operations
- 2 **Memory bandwidth**
for moving data (packets, lookup) and instructions (packet processing)
- 3 **I/O bandwidth**
for moving packets to/from NIC interfaces
- 4 **Inter-socket bandwidth**
for handling inter-socket operations

$$CyclesPerPacket [ClockCycles] = \frac{\#Instructions}{Packet} * \frac{\#Cycles}{Instruction}$$

$$Throughput [pps] = \frac{1}{Packet_Processing_Time[sec]} = \frac{CPU_freq[Hz]}{Cycles_per_Packet}$$

$$Throughput [bps] = Throughput[pps] * Packet_Size[pps]$$

Processing Packets: What Improves in Compute ..



Resources to Get Performance

- 1 **Processor and CPU cores**
FrontEnd: faster instr. decoder (4- to 5-wide)
BackEnd: faster L1 cache, bigger L2 cache, deeper OOO* execution
Uncore: move from ring to X-Y fabric mesh
- 2 **Memory bandwidth**
~50% increase: channels (4 to 6), speed (DDR-2666)
- 3 **I/O bandwidth**
>50% increase: PCIe lanes (40 to 48), re-designed IO blocks
- 4 **Inter-socket bandwidth**
~60% increase: QPI to UPI (2x to 3x), interface speed (9.6 to 10.4 GigTrans/sec)

$$\text{CyclesPerPacket [ClockCycles]} = \frac{\#Instructions}{\text{Packet}} * \frac{\#Cycles}{\text{Instruction}}$$

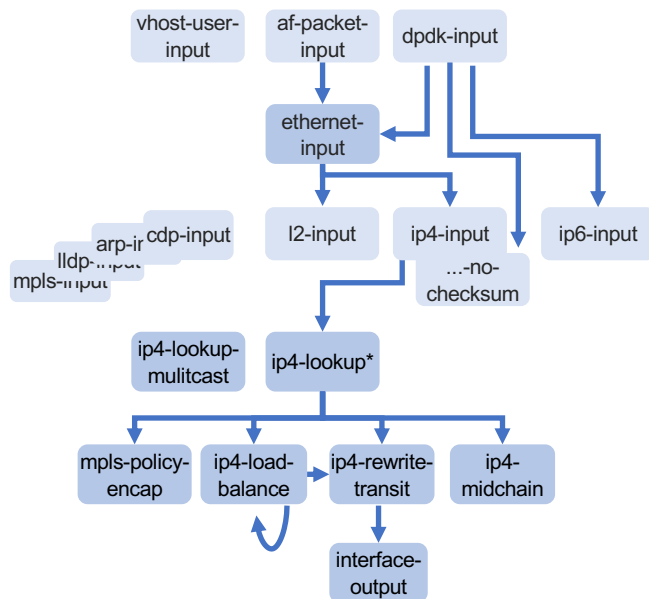
Moore's Law in Action

$$\text{Throughput [bps]} = \text{Throughput[pps]} * \text{Packet_Size[pps]}$$

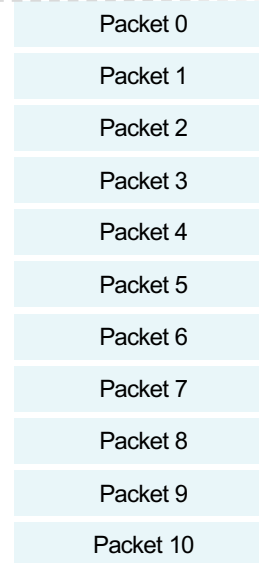
FD.io VPP – The “Magic” of Vectors

Compute Optimized SW Network Platform

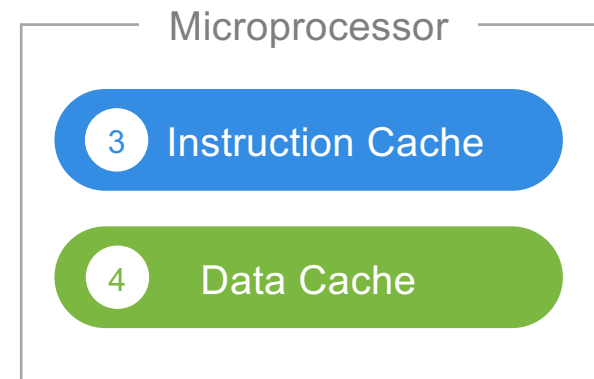
1 Packet processing is decomposed into a directed graph of nodes ...



2 ... packets move through graph nodes in vector ...



3 ... graph nodes are optimized to fit inside the instruction cache ...



4 ... packets are pre-fetched into the data cache.

* Each graph node implements a “micro-NF”, a “micro-NetworkFunction” processing packets.

**Makes use of modern Intel® Xeon® Processor micro-architectures.
Instruction cache & data cache always hot → Minimized memory latency and usage.**



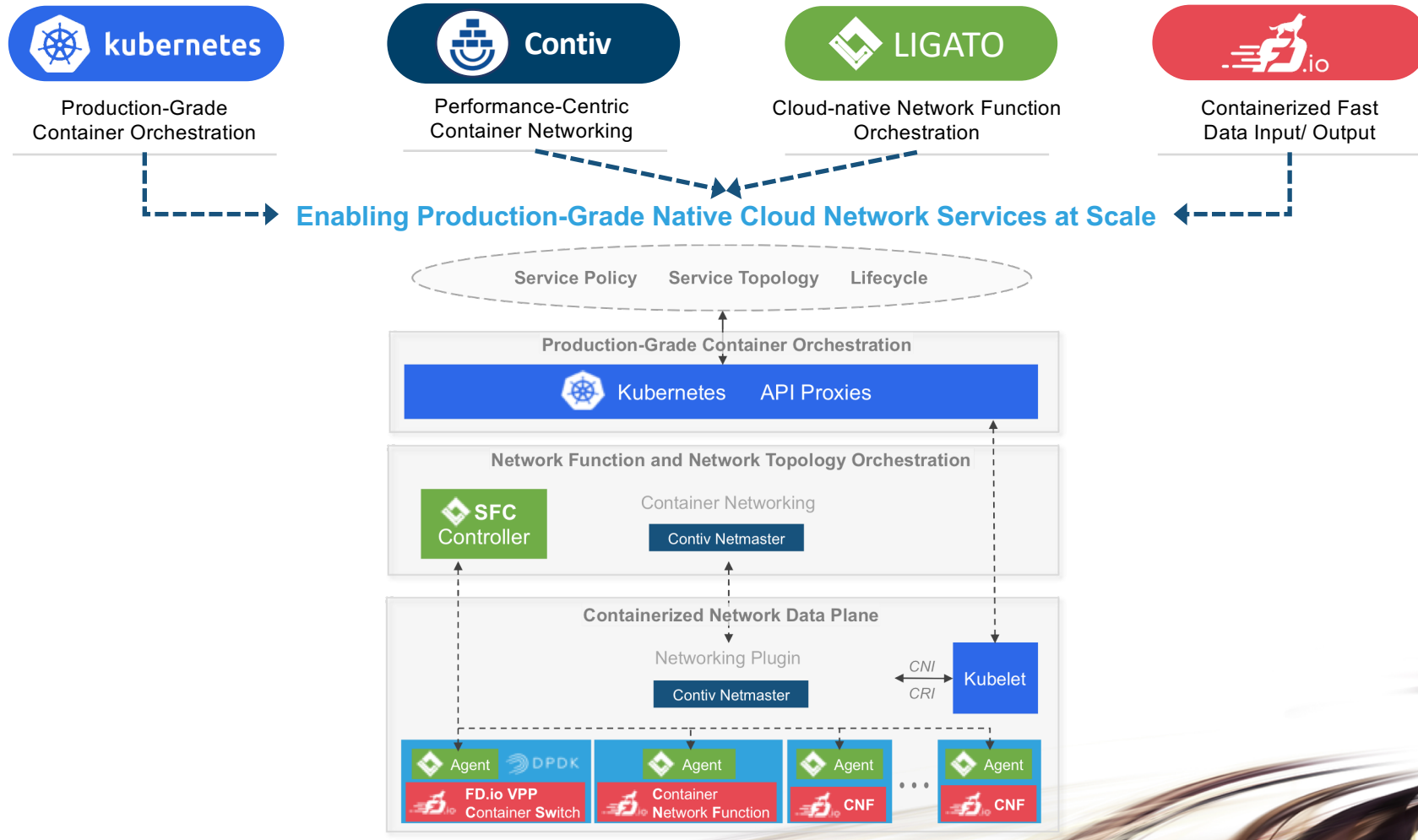
K8s perspective: Orchestration/K8s incl. placement for scale-out

- We have all of this performance, how do we use it
- How do we make it work for us at scale
- How do we intelligently make an efficient use of available resource to deliver intended application and networking services



Cloud-native Network Micro-Services

For Native Cloud Network Services



Kubernetes Overview

- "Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation."
- Terminology:
 - **Node:** a host or VM on which pods can be scheduled
 - **Pod:** one or more co-resident linux containers with a single IP address (typically a /24 of address space is provided per node)
 - **Deployment:** a set of pods implementing the same application
 - **Service:** an abstraction providing a single persistent IP address for a deployment (Kubernetes provides mechanisms to load balance across multiple pods)
- Kubernetes assumes seamless connectivity between pods, wherever it places them
- A networking plugin is needed to abstract the network

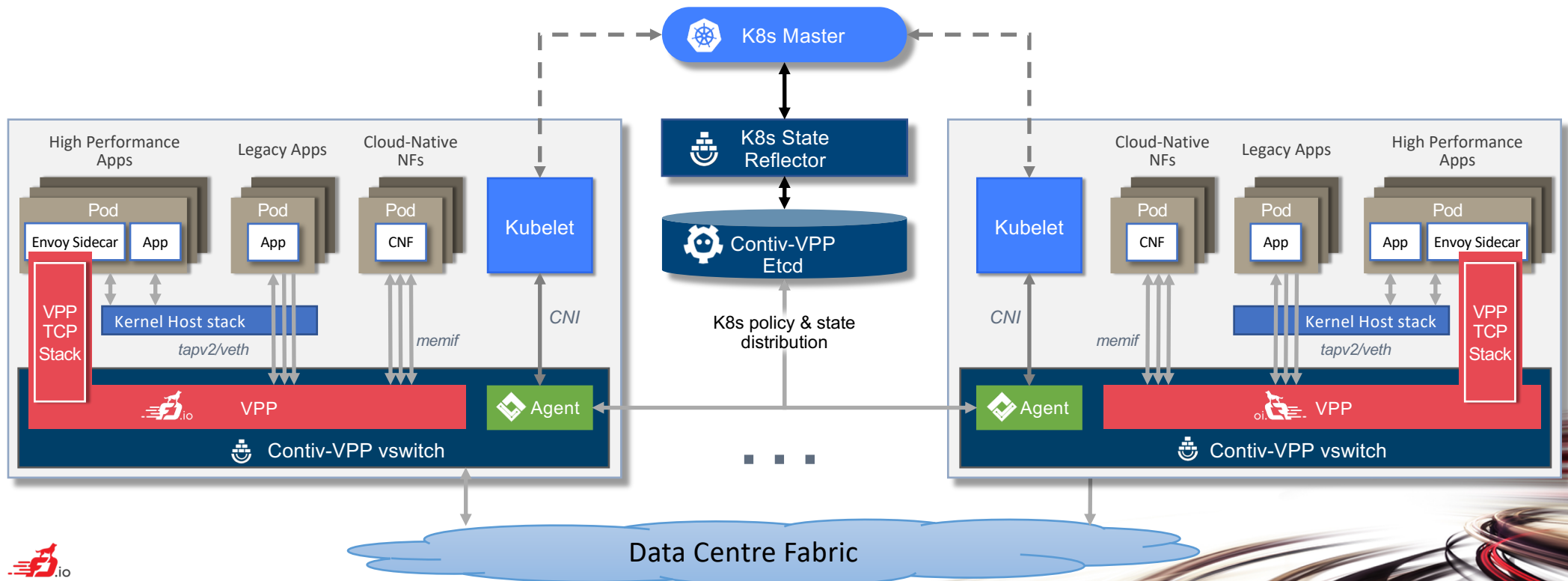
Contiv-VPP Overview

- Contiv-VPP is a networking plugin for Kubernetes that:
 - Allocates IP addresses to Pods (IPAM)
 - Programs the underlying infrastructure it uses (VPP and the Linux TCP/IP stack) to connect the pods to other pods in the cluster and/or to the external world
 - Implements K8s network policies that define which pods can talk to each other
 - Implements K8s services using a stateful load-balanced NAT function
- Contiv-VPP is a user-space, high-performance, high-density networking plugin for Kubernetes - leveraging FD.io/VPP as the industry's highest performance software data plane



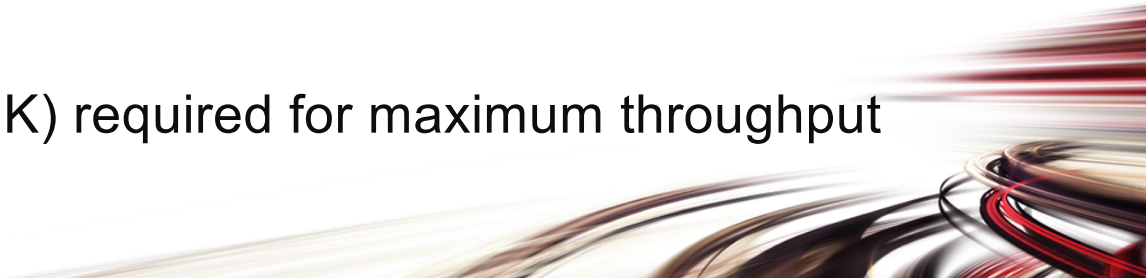
Contiv-VPP Architecture

- Can deliver complete container networking solution entirely from user-space
- Replace all eth/kernel interfaces with memif/user-space interfaces.
- Apps can add VCL library for Higher Performance (bypass Kernel host stack and use VPP TCP stack)
- Legacy apps can still use the kernel host stack in the same architecture

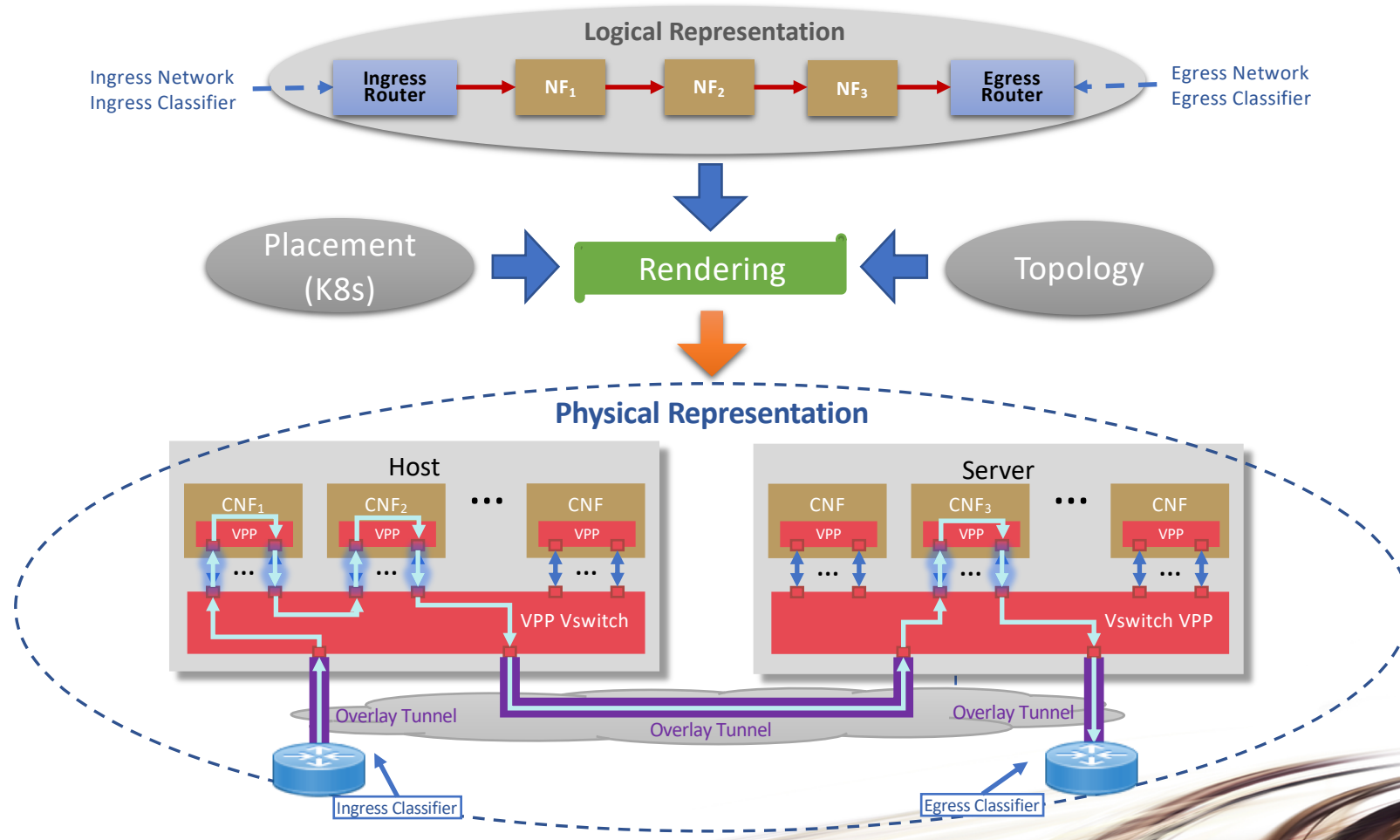


What Container Networking Lacks for NFV Use-Cases

- Networking
 - HTTP or NAT-based load balancing isn't suitable for NFV use-cases
 - No support for multiple interfaces or IP addresses per pod
 - No support for high-speed wiring of NFs
- Policy
 - No support for QoS, network-aware placement etc.
- Isolation
 - Applications run in user-space – kernel networking is unsuited to NFV
- Performance
 - Polling mode drivers (e.g. DPDK) required for maximum throughput

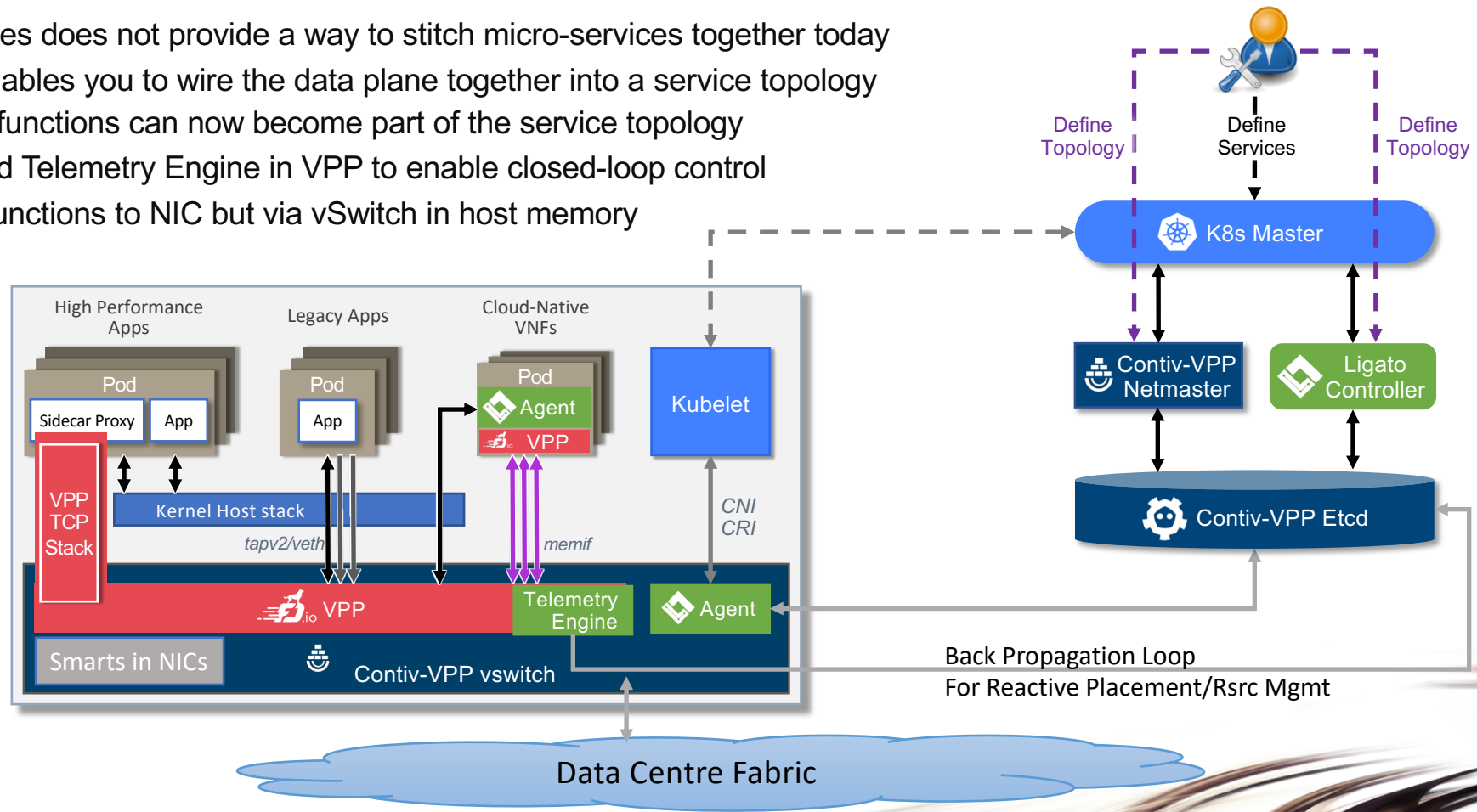


Service Function Chaining with Ligato



Ligato – Cloud-native NFs (CNFs)

- Kubernetes does not provide a way to stitch micro-services together today
- Ligato enables you to wire the data plane together into a service topology
- Network functions can now become part of the service topology
- Dedicated Telemetry Engine in VPP to enable closed-loop control
- Offload functions to NIC but via vSwitch in host memory



Unleashing Innovation in Networking

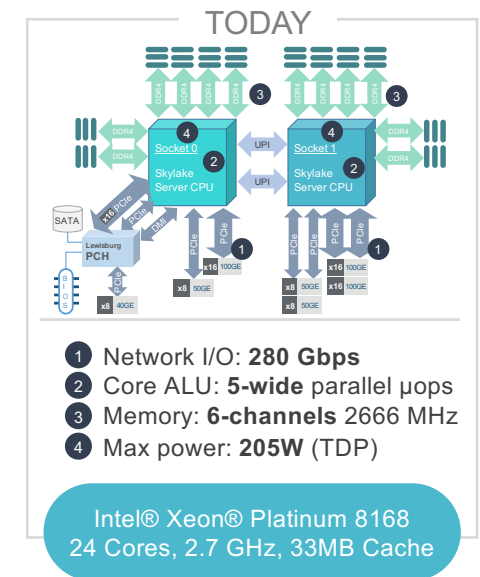
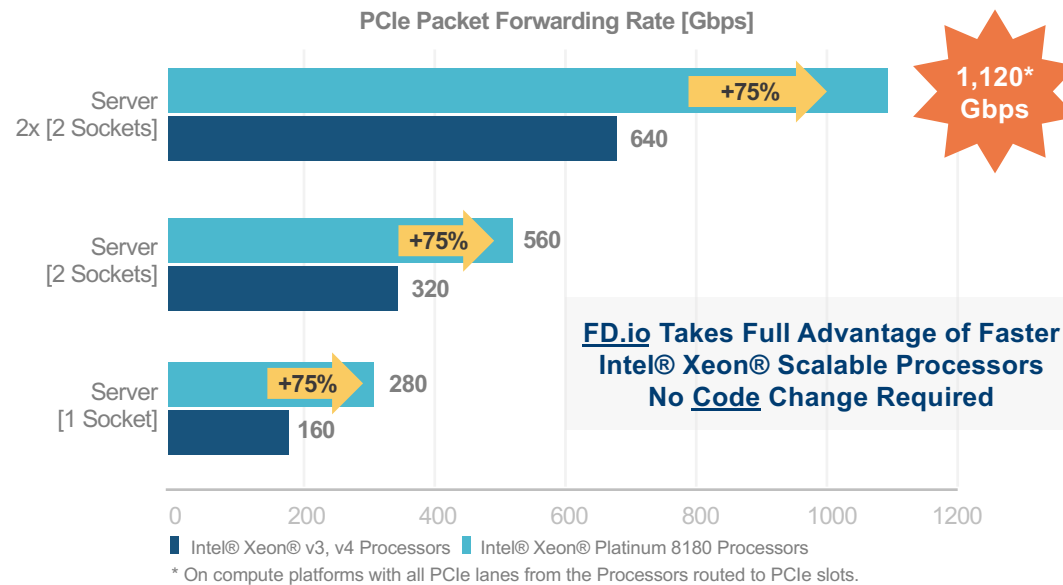
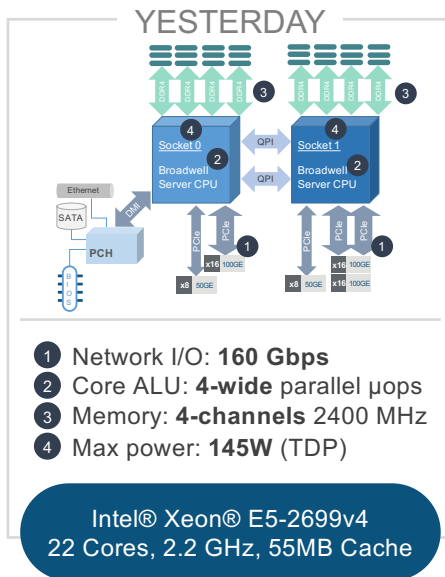
- Container networking requires fast innovation cycles to deploy new models
- Kernel upgrades and ad hoc modules may cause problems in production environments
- Multiple options are being explored in the industry

Technology	Programmability Model	Execution Context
OVS	OpenFlow lets users configure very granular data path behaviour	Primarily kernel based Openflow model
eBPF+XDP	Packets handled by user coded eBPF programs running in a sandbox	Bypass kernel networking stack but still in kernel-mode Bytecode + JIT + kernel helpers
FD.io / VPP	Regular user-mode C program with user loadable plugins	User-mode, no kernel dependencies Native NIC drivers, Linux APIs, DPDK



Baremetal Data Plane Performance Limit

FD.io benefits from increased Processor I/O



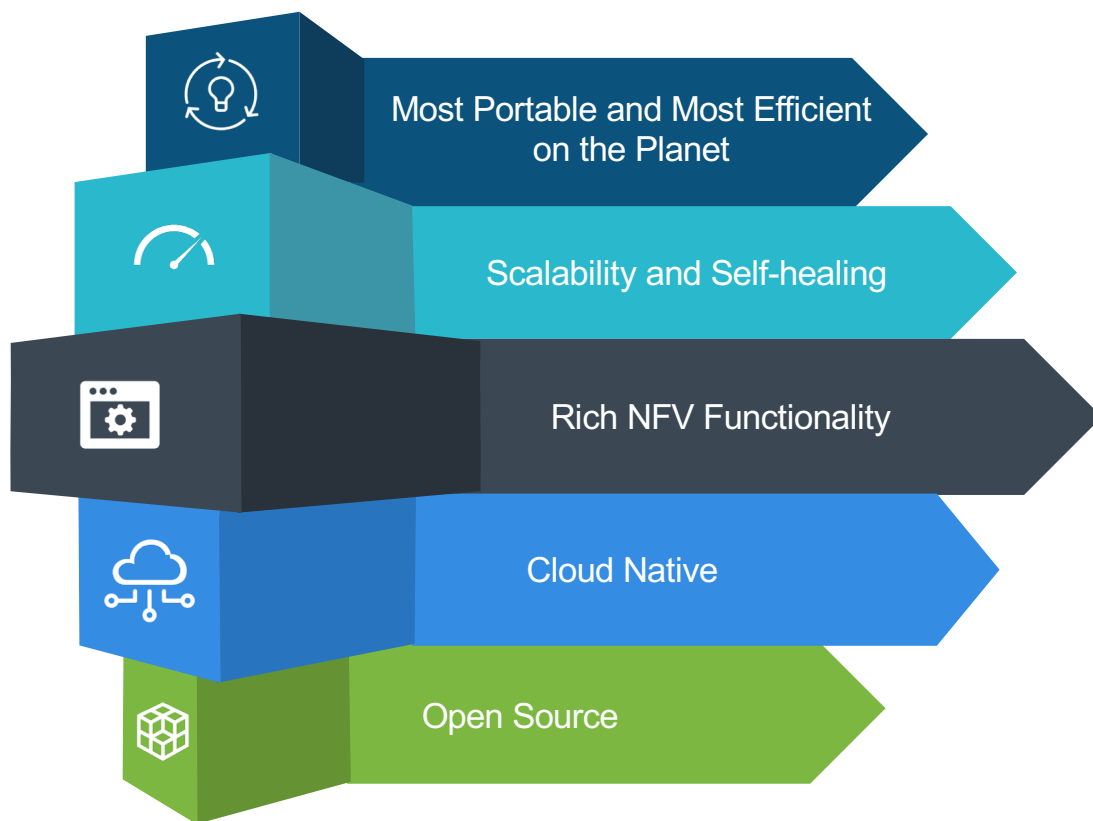
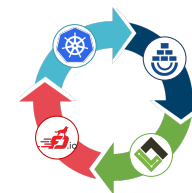
<https://goo.gl/Utbahy>






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

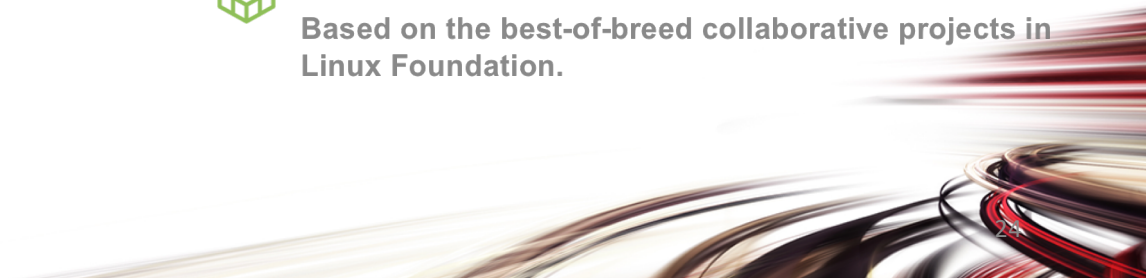


High Performance Cloud-Native Networking

K8s Unleashing FD.io



-  **PORTABILITY AND EFFICIENCY**
Public, private, hybrid, any-cloud. Over 10 times faster Container networking vs. alternatives.
-  **SCALABILITY and SELF-HEALING**
Follows Kubernetes scale and self-healing principles.
-  **SOFTWARE DEFINED NETWORKING**
FD.io VPP with 200 programmable “micro-NFs” (graph nodes), ~20 extension “micro-NF” plugins.
-  **CLOUD NETWORK SERVICES**
Containerized NFs managed as true cloud-native apps, provide and consume dat plane microservices.
-  **LINUX FOUNDATION**
Based on the best-of-breed collaborative projects in Linux Foundation.





ons

EUROPE

OPEN NETWORKING //
Integrate, Automate, Accelerate

September 25 - 27, 2018
Amsterdam, The Netherlands

High Performance Cloud-Native Networking

K8s Unleashing FD.io

THANK YOU !