A Cloud Native Networking Solution Based on Kube-router and VPP-DPDK

Hongjun Ni
Intel
Email: hongjun.ni@intel.com

Acknowledgement:
Ray Kinsella, Steve Liang @Intel
Pierre Pfister, Jerome Tollet @Cisco
Rastislav Szabo @Cisco

.io Foundation

About me

- Focus on FD.io/VPP and Cloud Native Networking
- VPP Maintainer (Load Balancer, VxLAN-GPE, NSH, GPTU, PPPoE)
- Sweetcomb Project Lead
- NSH_SFC Project Lead
- Hc2vpp Committer
- VPPSB Router Maintainer

Agenda

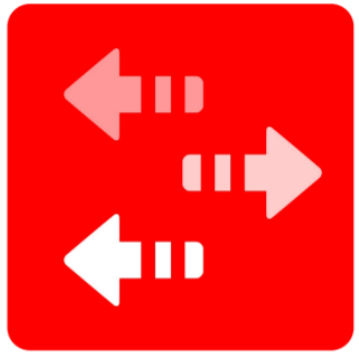
- Cloud Native Networking and Challenges
- Proposed Architecture
- Why Choosing FD.io
- Option 1: Load Balancer and Service Proxy
- Option 2: Pod Networking and DSR
- Option 3: Pod Ingress Firewall
- Key Takeaway
- Promote A New Project: Sweetcomb

What's in Cloud Native Networking?

Control Plane

Control Plane:

- Assigns IPs (from a pool given to each workload)
- Distributes routing information (i.e. how to get to this workload)
- Distributes policy (e.g. who can connect to whom)



Data Plane

Data Plane:

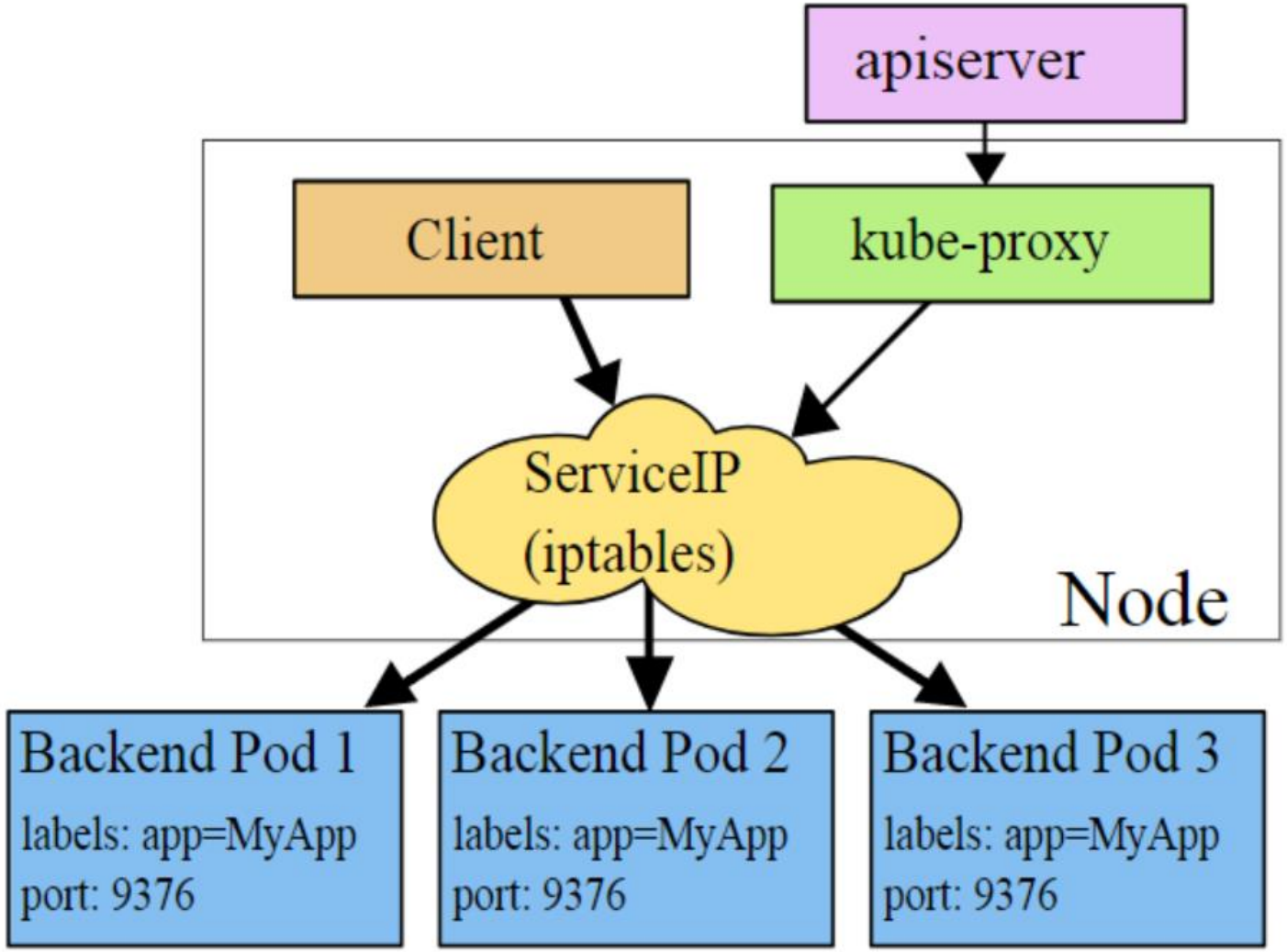
For each packet to/from the workload:

- Enforces policy
- Forwards it to the right destination

Reference: https://www.cncf.io/wp-content/uploads/2017/11/CNCF-Networking-Webinar-final-1-1.pdf

.io Foundation

Challenges on Present Solution



Linux kernel solution:

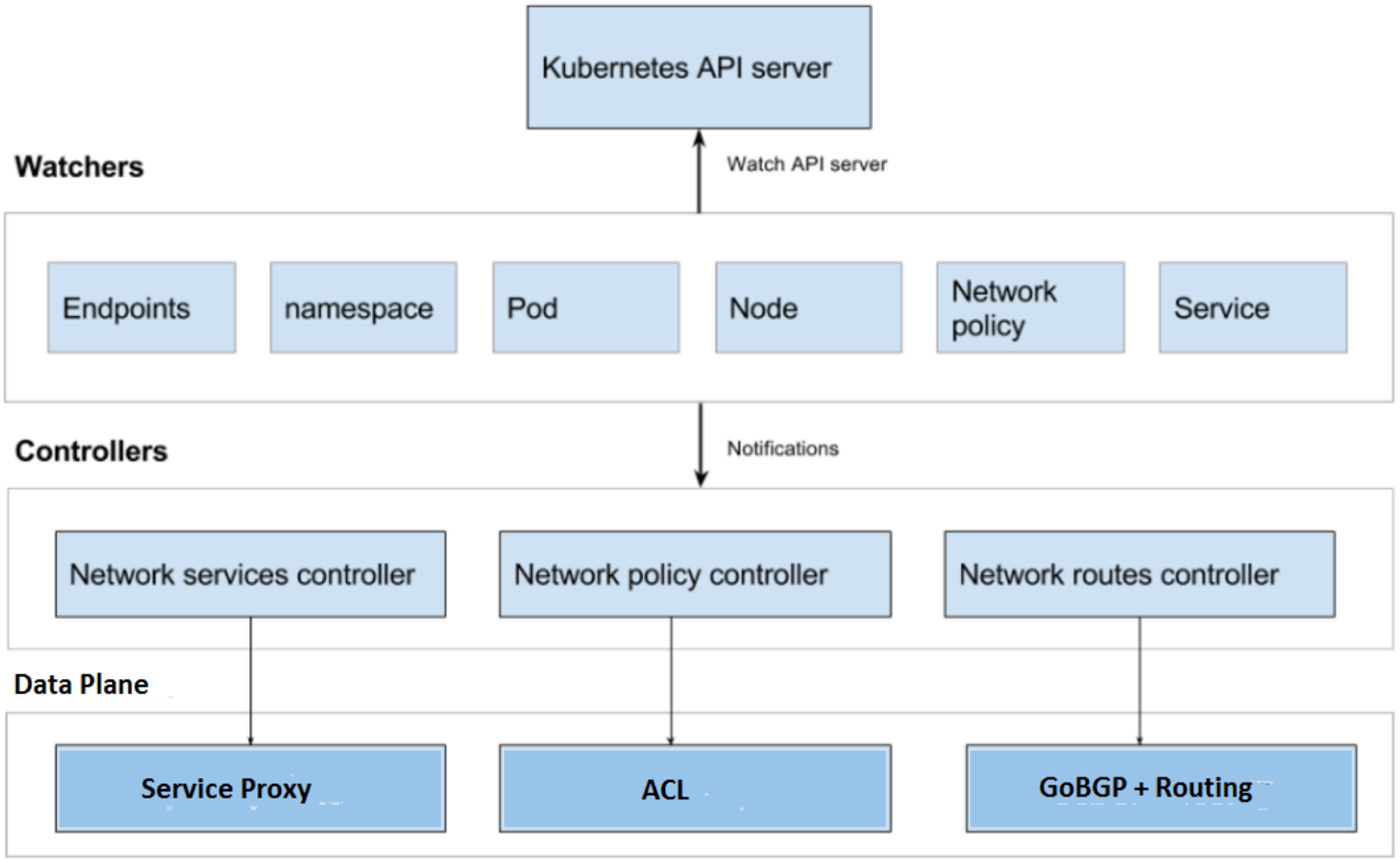
- Watches service and endpoints
- Installs iptables/IPVS rules
- Captures traffic and selects pod
- Redirects traffic to chosen pods

Problems:

- Uses load balancing on iptables/IPVS
- Uses NAT on iptables/IPVS
- Communication via VETH
- Performance degrades when iptables entries increase.

Reference: https://kubernetes.io/docs/concepts/services-networking/service

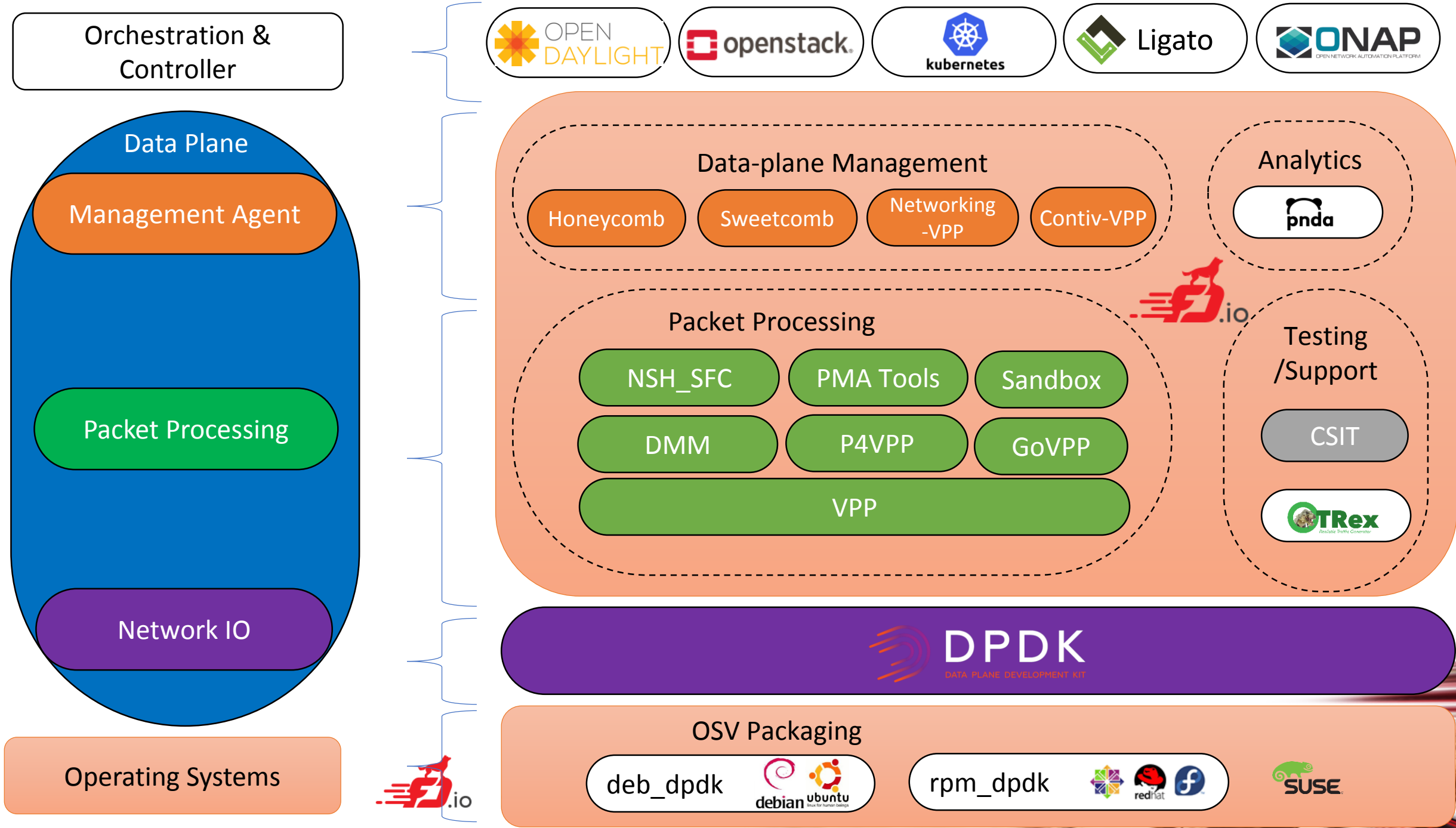
Proposed Architecture



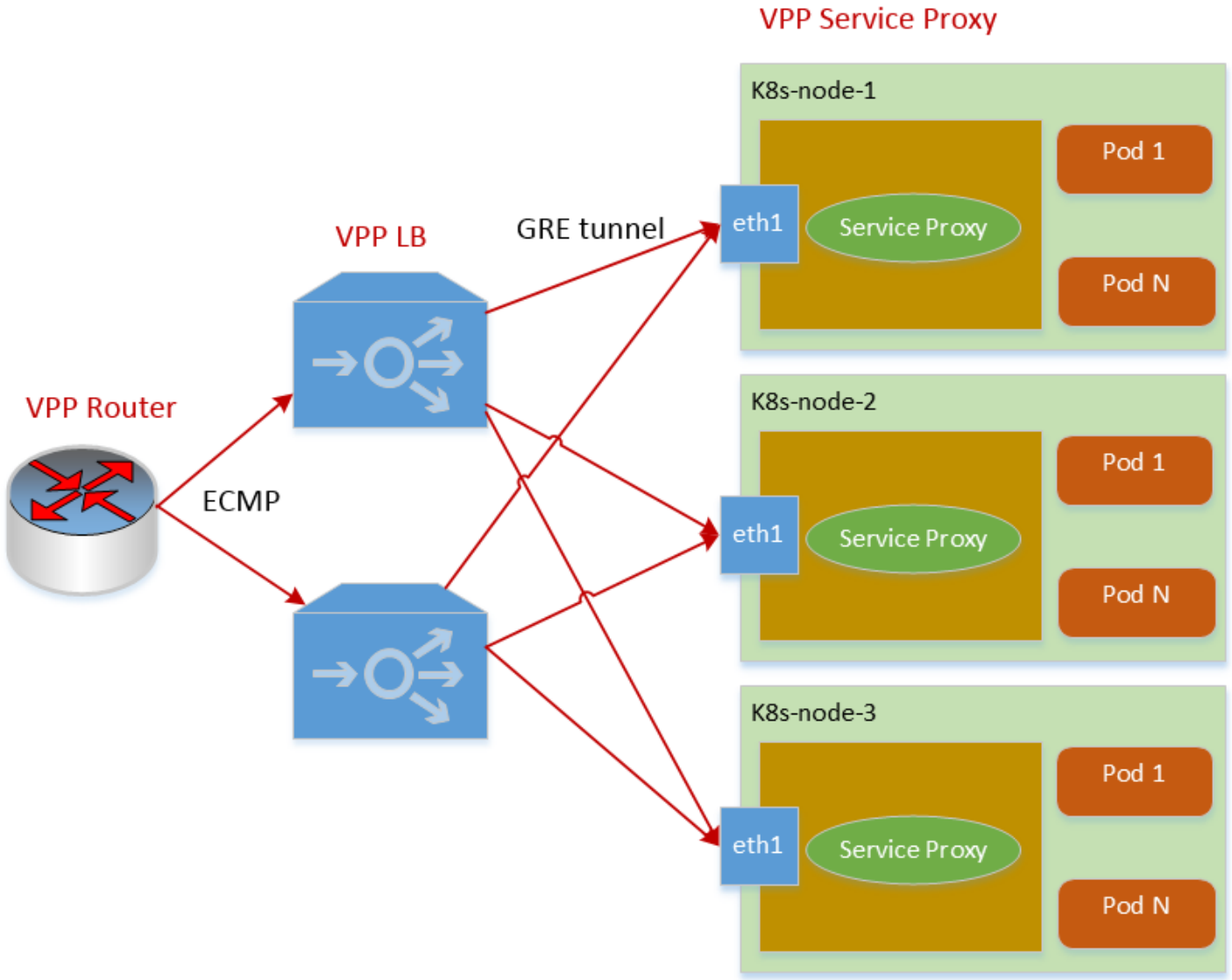
- A turnkey solution on Kube-router, replacing Linux kernel's networking stack.
- Load Balancer and Service Proxy Running on VPP-DPDK
- Policy based on VPP ACL
- Integration with GoBGP
- Routing based on VPP FIB

Existing Solution vs. Proposed Solution

	Existing Solution	Proposed Solution
Solution Stack	Linux Kernel Stack	Kube-router, VPP-DPDK
Policy Enforcement	Iptables + ipset	VPP ACL
Node Load Balancing	Iptables, IPVS	VPP kube-proxy
Connection Tracking	Iptables, IPVS	VPP kube-proxy
DNAT and SNAT	Iptables, IPVS	VPP kube-proxy
Communication between Host and Container	Via VETH	Via vhost-user or memif
External Load Balancer	Via CSP' Load Balancer	Via VPP Load Balancer
Routing	Linux Kernel Networking	GoBGP and VPP vRouter

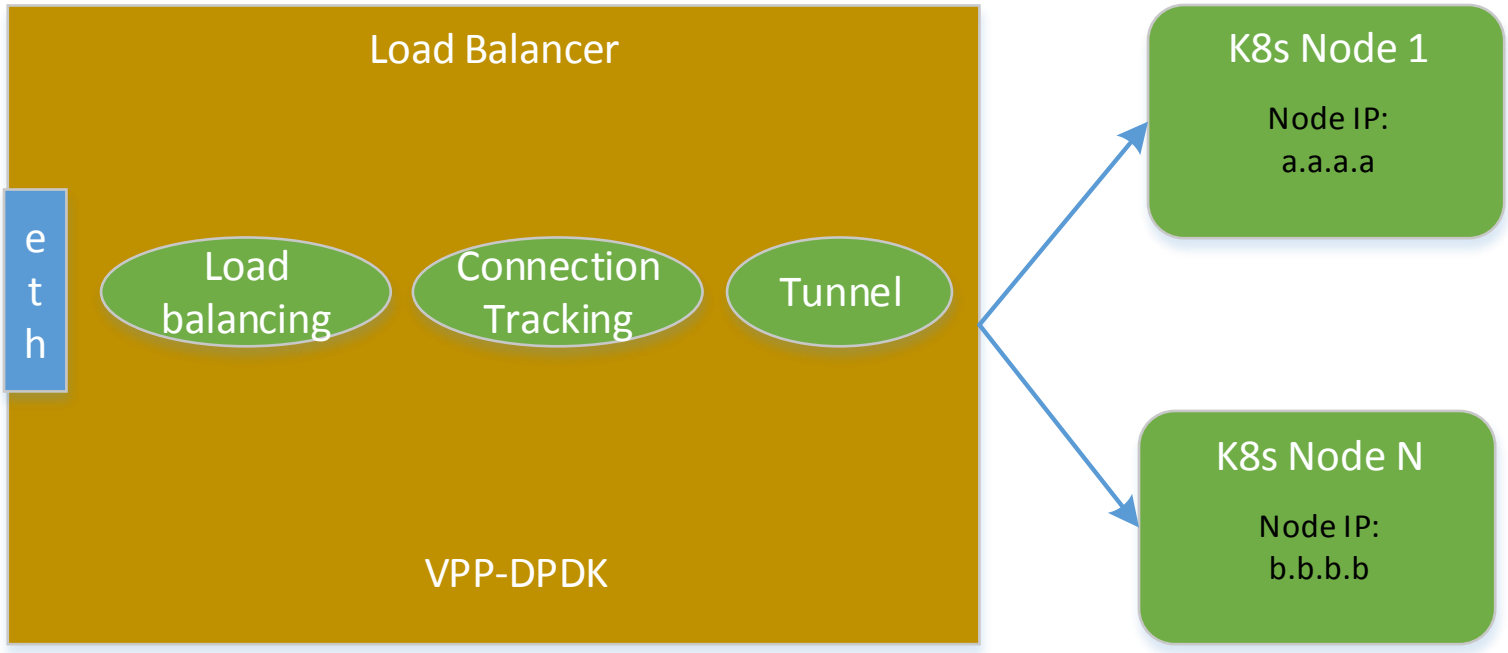

Orchestration & Controller
OPEN DAYLIGHT
openstack.
kubernetes
Ligato
ONAP
Data Plane
Management Agent
Packet Processing
Network IO
Data-plane Management
Honeycomb
Sweetcomb
Networking -VPP
Contiv-VPP
Analytics
pnda
.io
Packet Processing
NSH_SFC
PMA Tools
Sandbox
DMM
P4VPP
GoVPP
VPP
Testing /Support
CSIT
TRex
DPDK
DATA PLANE DEVELOPMENT KIT
Operating Systems
.io
OSV Packaging
deb_dpdk
debian
ubuntu
rpm_dpdk
redhat
SUSE

Option 1: Load Balancer and Service Proxy



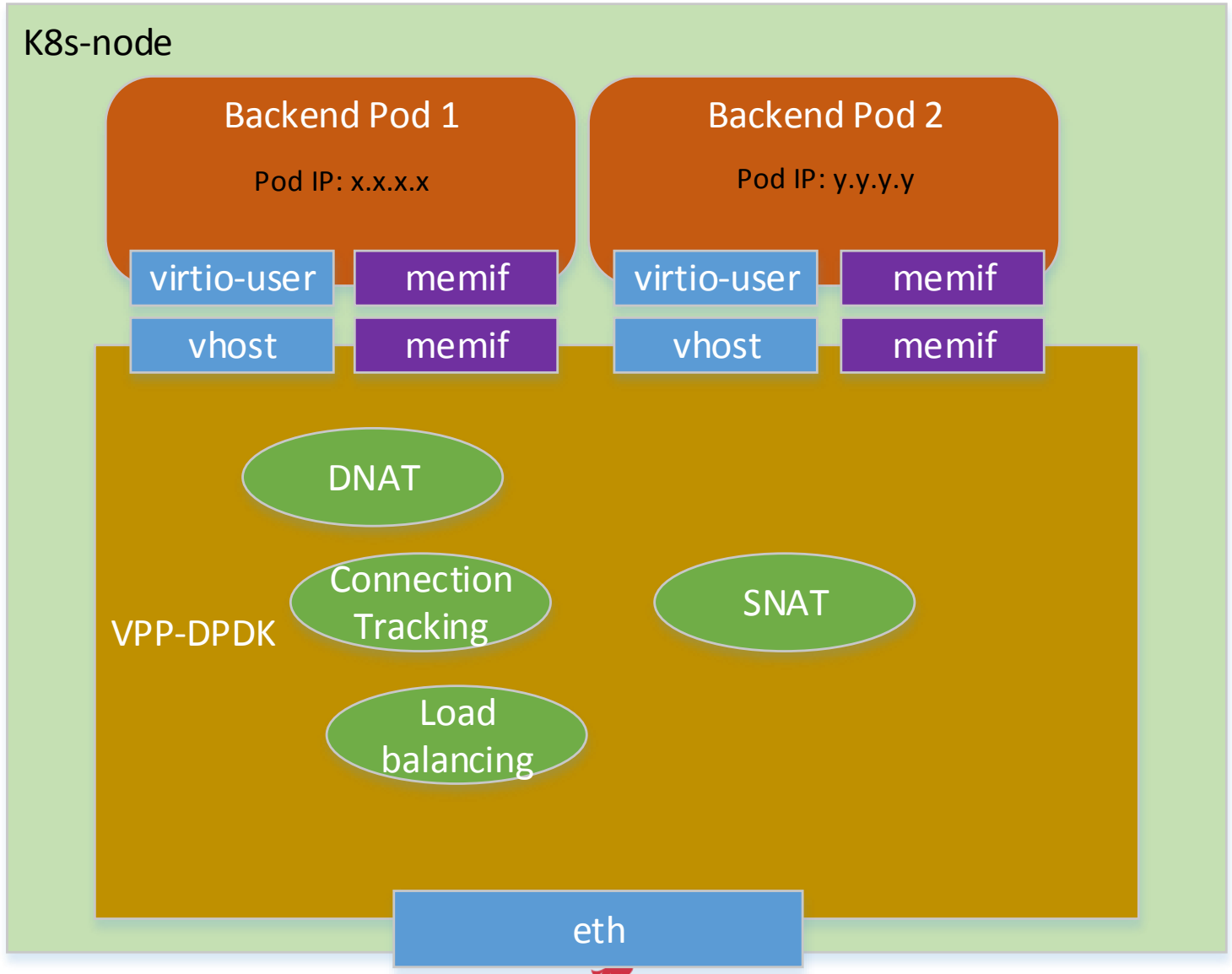
- Kube-router is deployed on each node and run as a service proxy, replacing Linux kernel kube-proxy.
- Router, Load Balancer and Service Proxy could be implemented on VPP.
- They run as typical K8s networking.

Option 1: Load Balancer on VPP-DPDK



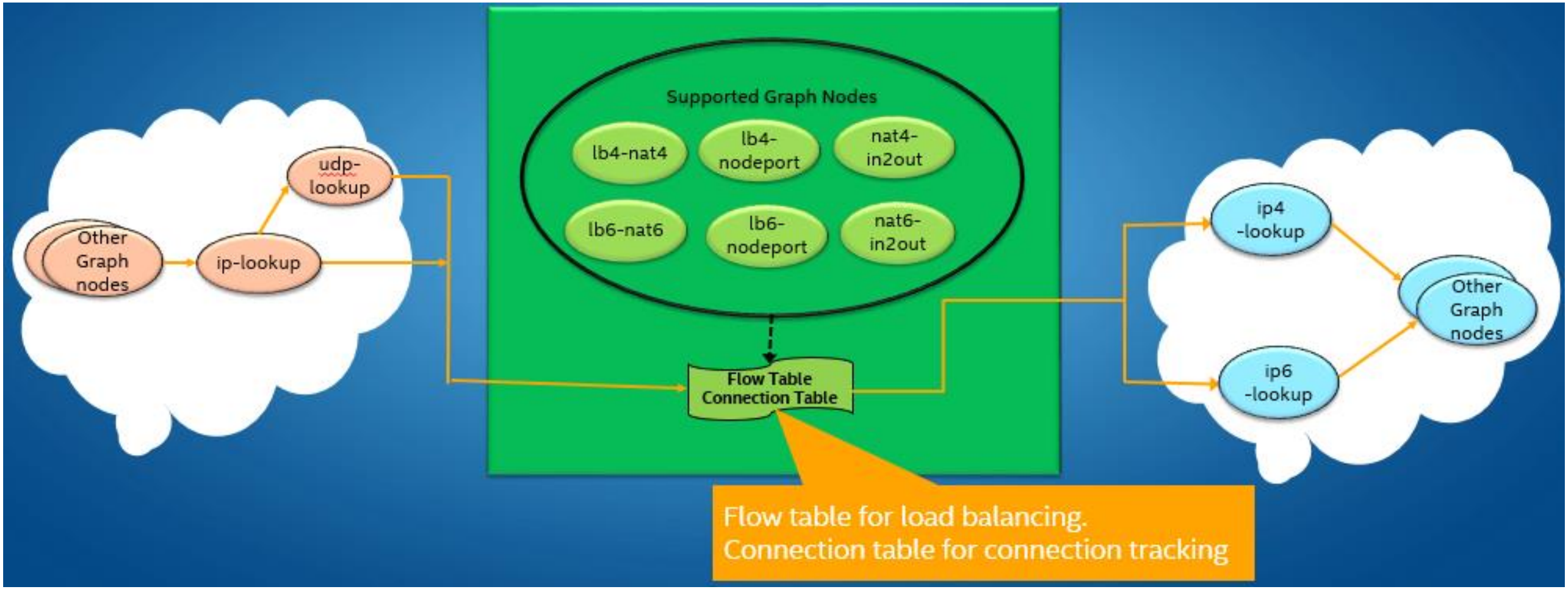
- Distributes traffic among K8s nodes
- Consistent Hashing ensures resilience to K8s node changes.
- Connection Tracking supports connection persistence.
- Supports two encapsulation types
  - GRE tunnel
  - IPIP tunnel

Option 1: Service Proxy on VPP-DPDK



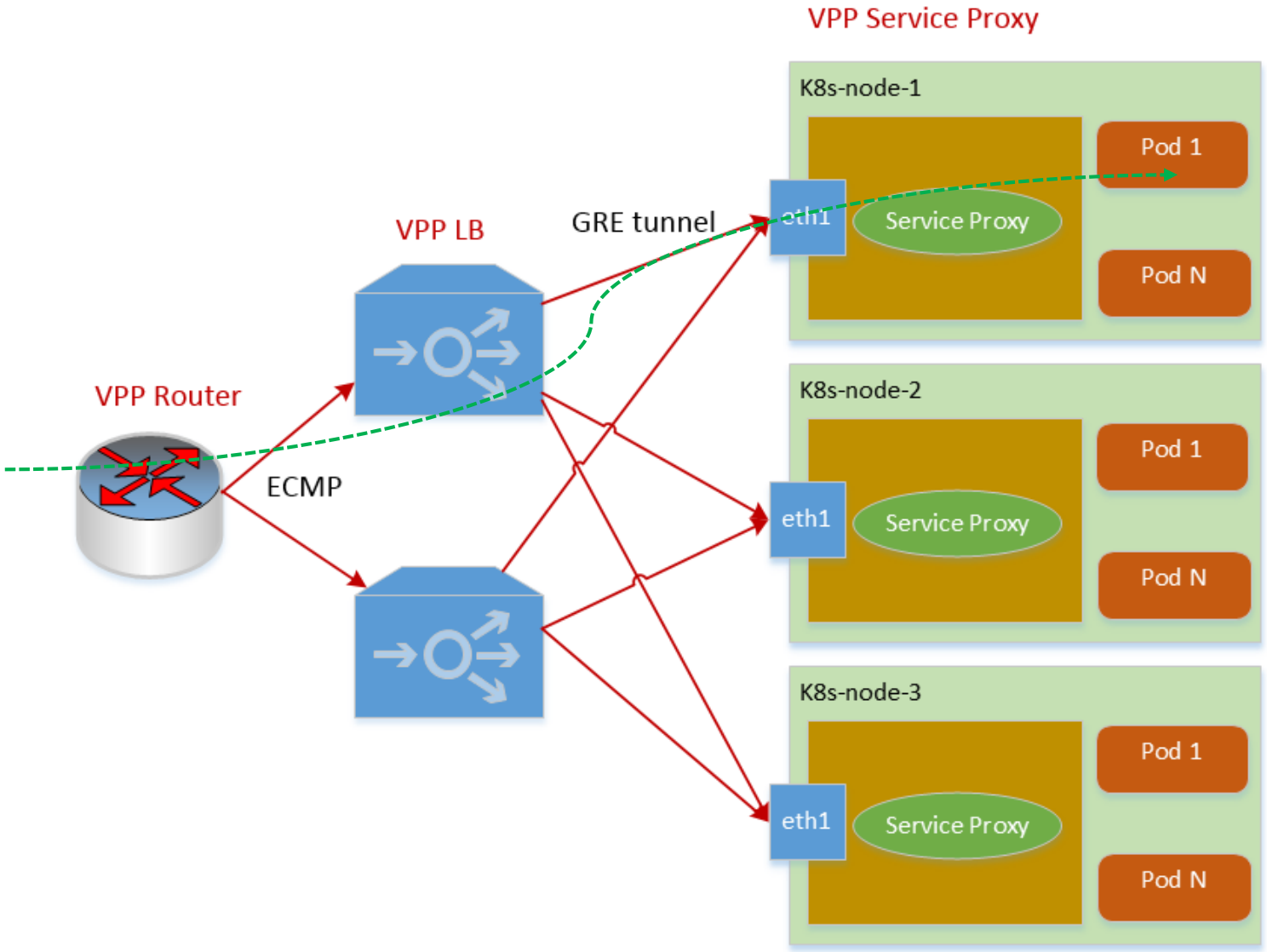
- Distributes traffic among Pods
- Supports two interface types
  - vhost and virtio-user
  - memif
- Supports three service types
  - ClusterIP
  - NodePort
  - External LoadBalancer

Option 1: Service Proxy Data Plane Internals



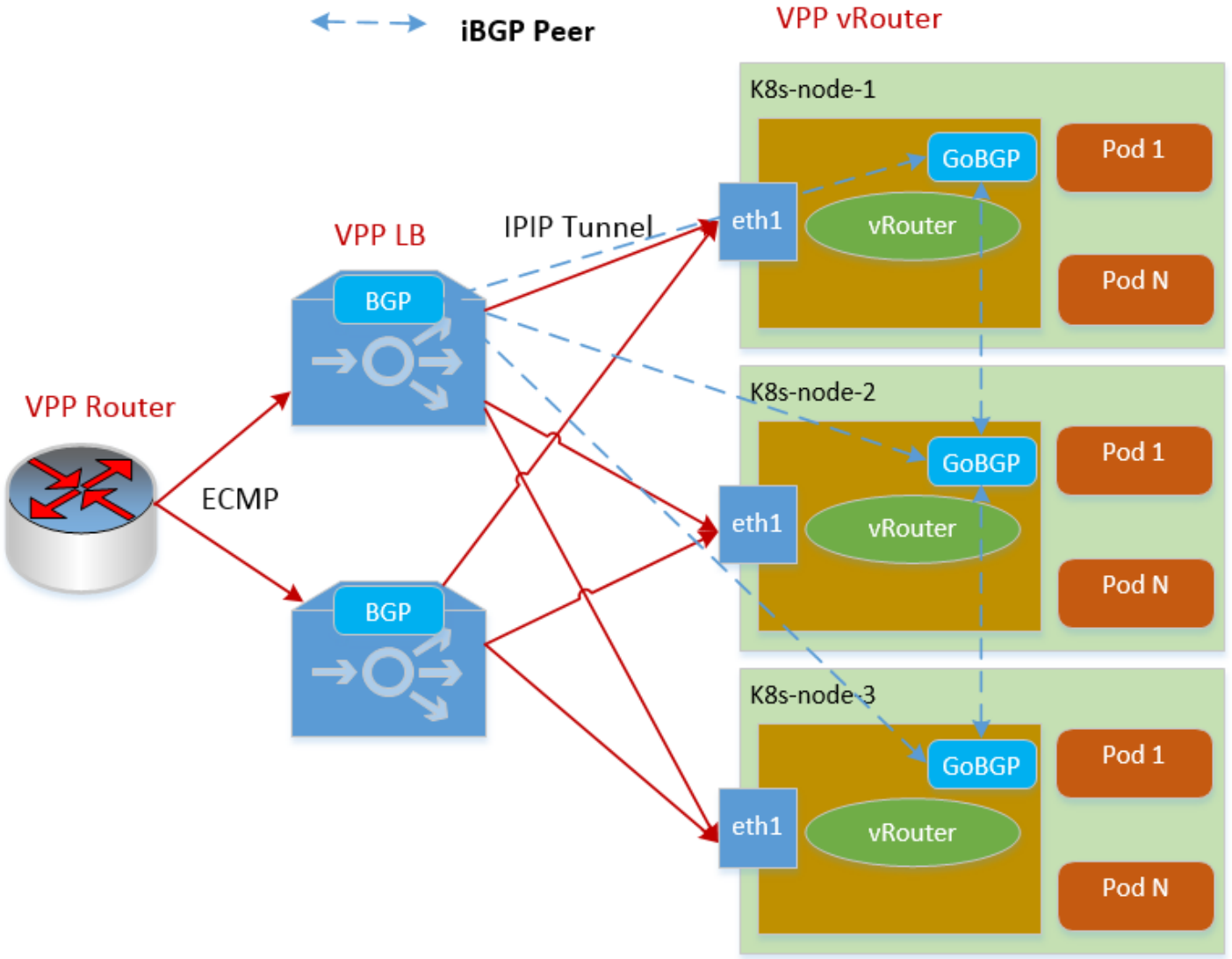
- Ingress IPv4 traffic goes through lb4-nodeport, lb4-nat4 graph nodes.
- Egress IPv4 traffic goes through nat4-in2out graph node.
- IPv6 traffic has similar data path as IPv4.

Option 1: Flow Path



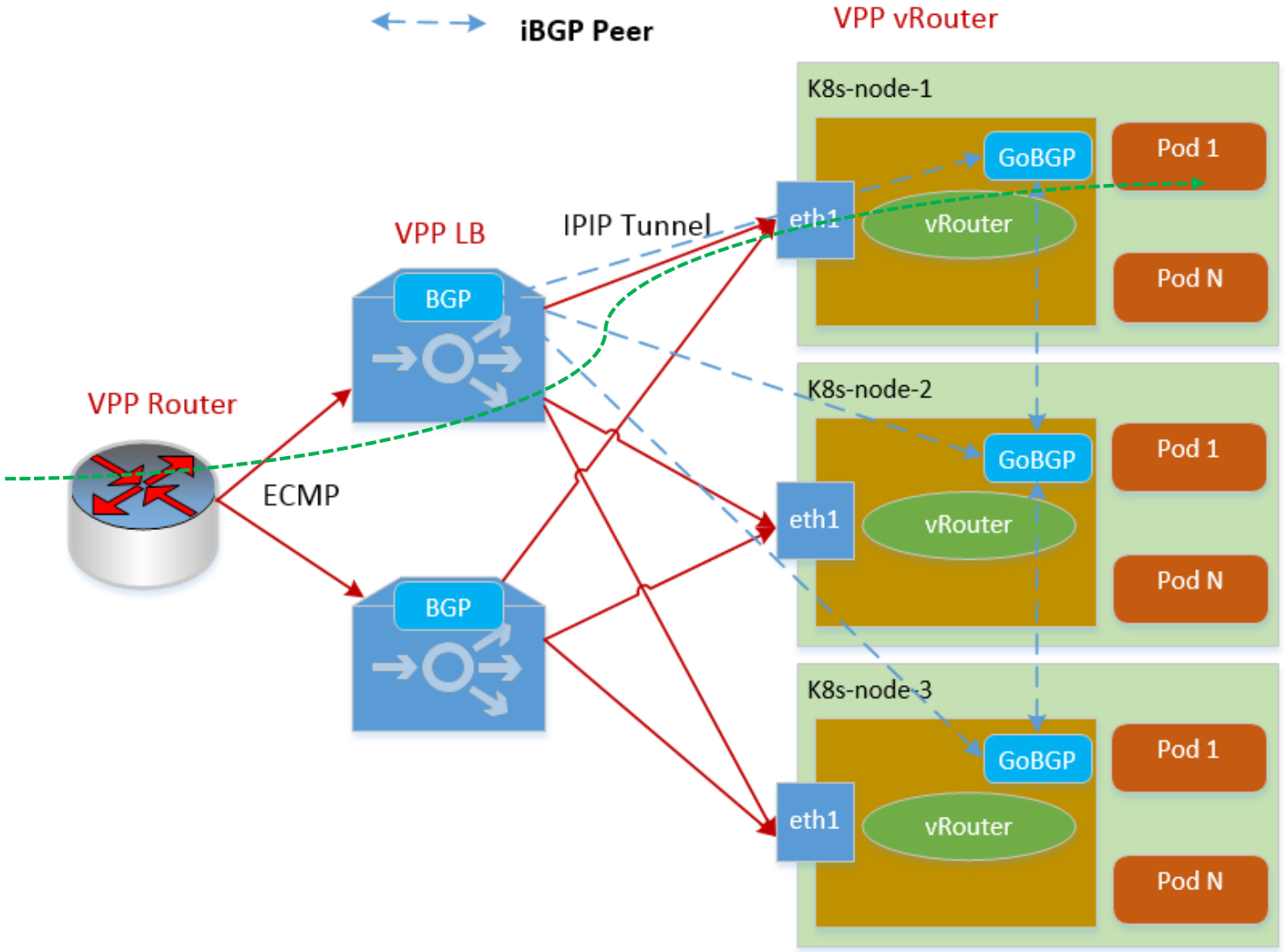
- VPP Router enables ECMP feature.
- VPP Load Balancer distributes traffic and encapsulates packets via GRE tunnels. A specific flow will be sent to the same K8s node.
- On K8s node, it removes GRE tunnel and goes through Service Proxy and performs DNAT, and then distributes traffic to selected pods.
- Return traffic will also pass through Service Proxy performing SNAT.
- Pod IPs are not visible to VPP LB.

Option 2: Pod Networking and DSR



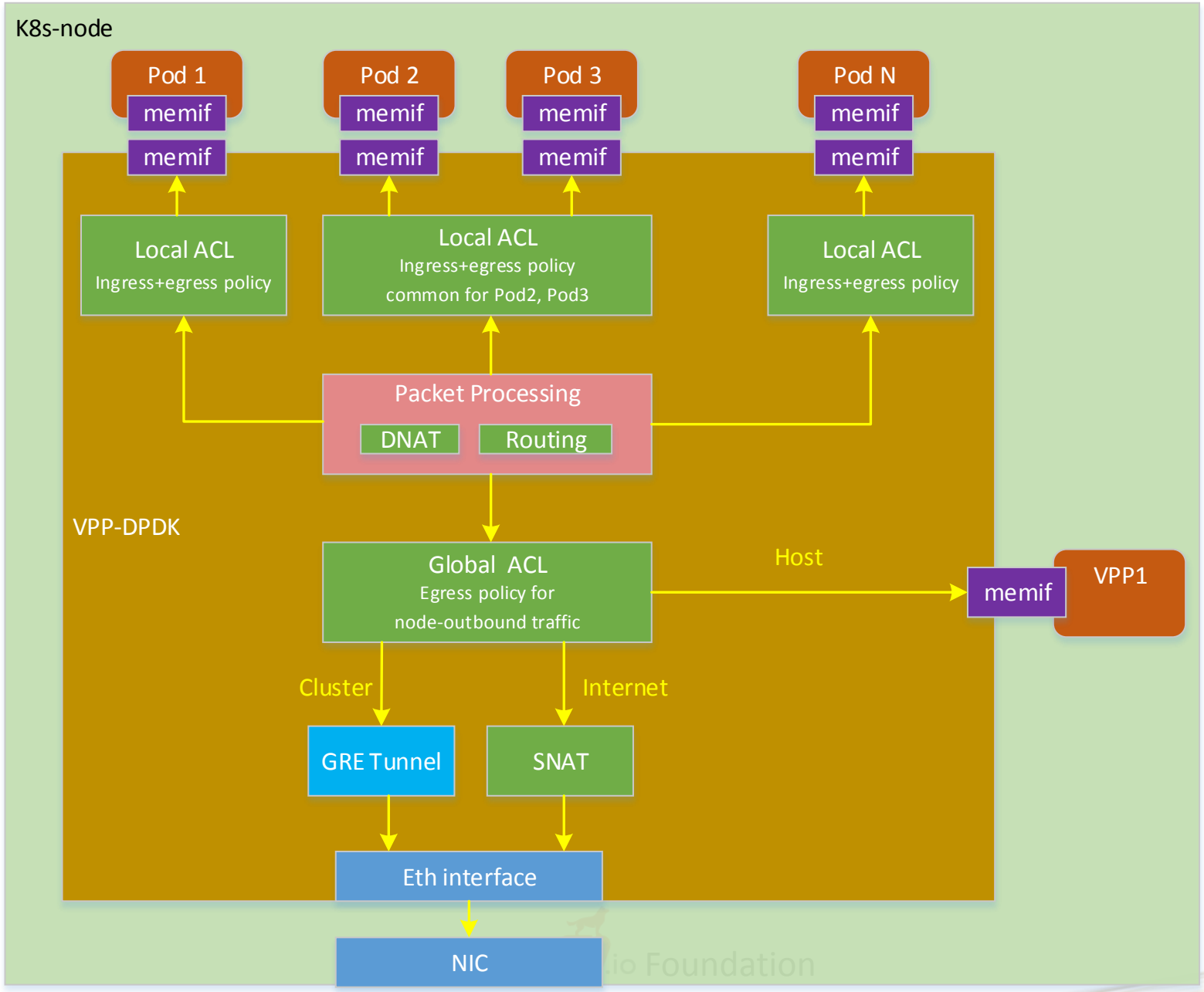
- Kube-router is deployed on each node and run as a vRouter.
- Each K8s node runs iBGP and peers with LB and rest K8s nodes.
- Pod IPs are visible to VPP LB and rest K8s nodes.

Option 2: Flow Path



- VPP Router enables ECMP feature.
- VPP Load Balancer selects pod as per Service IP and encapsulates packets through IPIP tunnels.
- On K8s node, vRouter just routes IPIP traffic to selected Pod.
- On each Pod, it removes IPIP tunnel, manipulates packets, swaps source IP and destination IP, and then sends packets directly to clients (Direct Server Return).

Option 3: Pod Ingress Firewall



- Leverage VPP ACL feature.
- Convert ingress and egress rules into per-pod ingress and egress ACLs (local tables), each assigned to a memif interface connecting to a pod.
- A single egress ACL (global table) assigned to interfaces connecting rest cluster node, the tunnel and the memif interface connecting other VPP instance.
- Pods with the same policy configuration share the same ACL.

Key Takeaway

- A solution to enable high performance K8s networking.
- Kube-router provides operational simplicity and high performance.
- VPP-DPDK implements Load Balancer and Service Proxy.
- Kube-router handles Pod networking with direct routing on VPP-DPDK.
- Converts Network Polices to ingress and egress rules on local and global ACL.

Promote a New Project: Sweetcomb

Sweetcomb project scope:

- Northbound interfaces:
  - Netconf northbound interface
  - Restconf northbound interface
  - SSL northbound interface
- Yang models for VPP management
  - Configuration data
  - Operational data
- Translation layer between VPP management and Yang based data structures
  - Must support all features of VPP exposed in its APIs in an extensible manner
- Expose APIs to integrate with other open source projects
  - Base implementation of all generic southbound interfaces leverage VPP-VAPI
  - expose APIs to integrate with SD-WAN control plane, such as SDN Controller.
  - expose APIs to integrate with Routing Daemon, such as FRR.
  - expose APIs to integrate with IKE protocol, such as strongswan.
  - expose APIs to integrate with DPI control plane, such as nDPI.
  - expose APIs to integrate with BRAS control plane, such as OpenBRAS.
  - To be added.

Sweetcomb Facts

Project Lead: Hongjun Ni, @ Intel

Committers:

- Drenfong Wang, @ Intel,
- Chuanguo Wang, @ HuachenTel,
- Hui Chen, @ HuachenTel,
- Zhuoqun Li, @ China Mobile,
- Hongfeng Li, @ China Unicom,
- Jinglong Zhi, @ China Telecom,
- Changlin Li, @ NXP,
- Tianyi Wang, @ Tieto,
- Feng Gao, @ Tencent,
- Jianyong Chen, @ Alibaba,
- Jian Gu, @ ZTE,
- Jerome Tollet, @ Cisco,
- Maciek Konstantynowicz, @ Cisco,
- Rastislav Szabo, @ Cisco,
- Hongjun Ni, @ Intel,

Repository: git clone https://gerrit.fd.io/r/sweetcomb

Mailing List: sweetcomb-dev@lists.fd.io

Jenkins: jenkins silo

Gerrit Patches: code patches/reviews

Bugs: sweetcomb bugs

Thank you !

Q & A

Email : hongjun.ni@intel.com