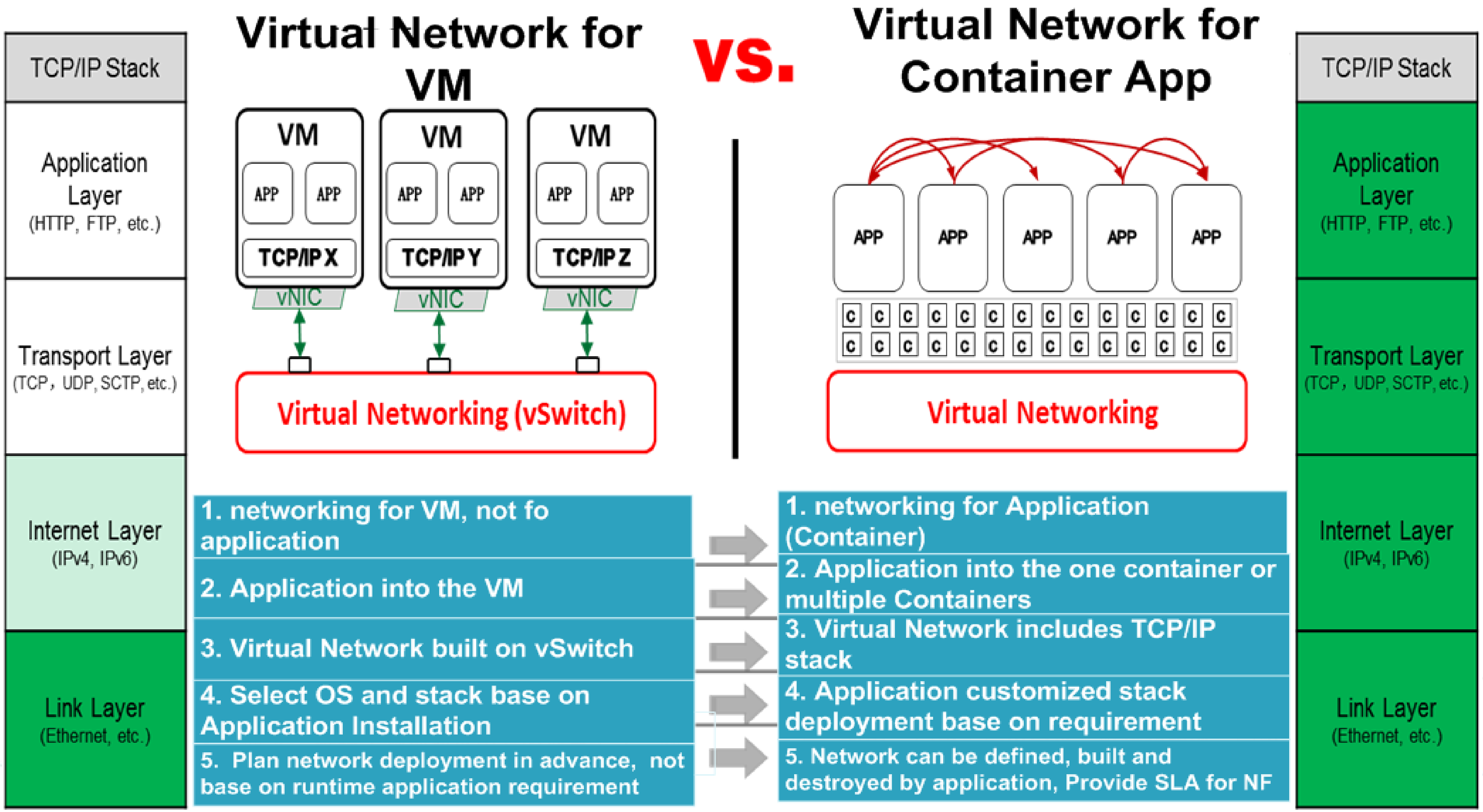
Simplify Container Networking With iCAN

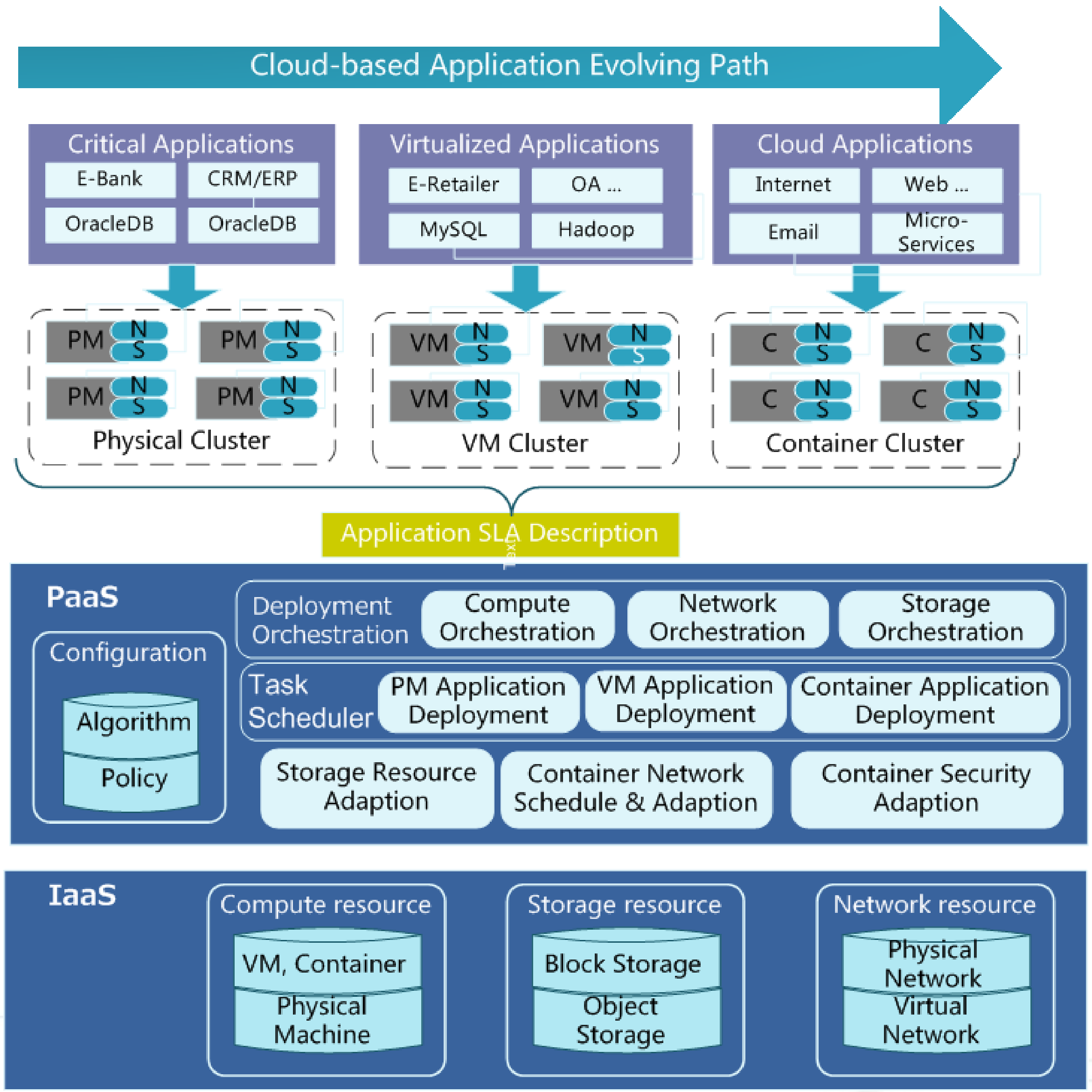
Huawei Cloud Network Lab

HUAWEI

Container Network Defined By Application



What we face today



•Automation Deployment and Orchestration:

✓Automate deploy resource for application based on Application SLA (bandwidth / delay / security)

✓Compatible with SDN controller

✓Need to deal with High Density Scale (10 x than VM)

✓More diverse and heterogeneous container network solutions, but every solution only target to solve a single problem

E-to-E SLA Assurance of the Container Network:.

✓Hope to provide applications with controllable network quality based on container platforms and systems

✓The flexibility of the virtual network make the control of network quality very difficult because of computing and I/O resources sharing between virtual network components and applications

✓No single SLA model applicable for all scenarios

"Application to Application" Monitoring :

✓With the development of container technologies, the virtual network becomes more complex

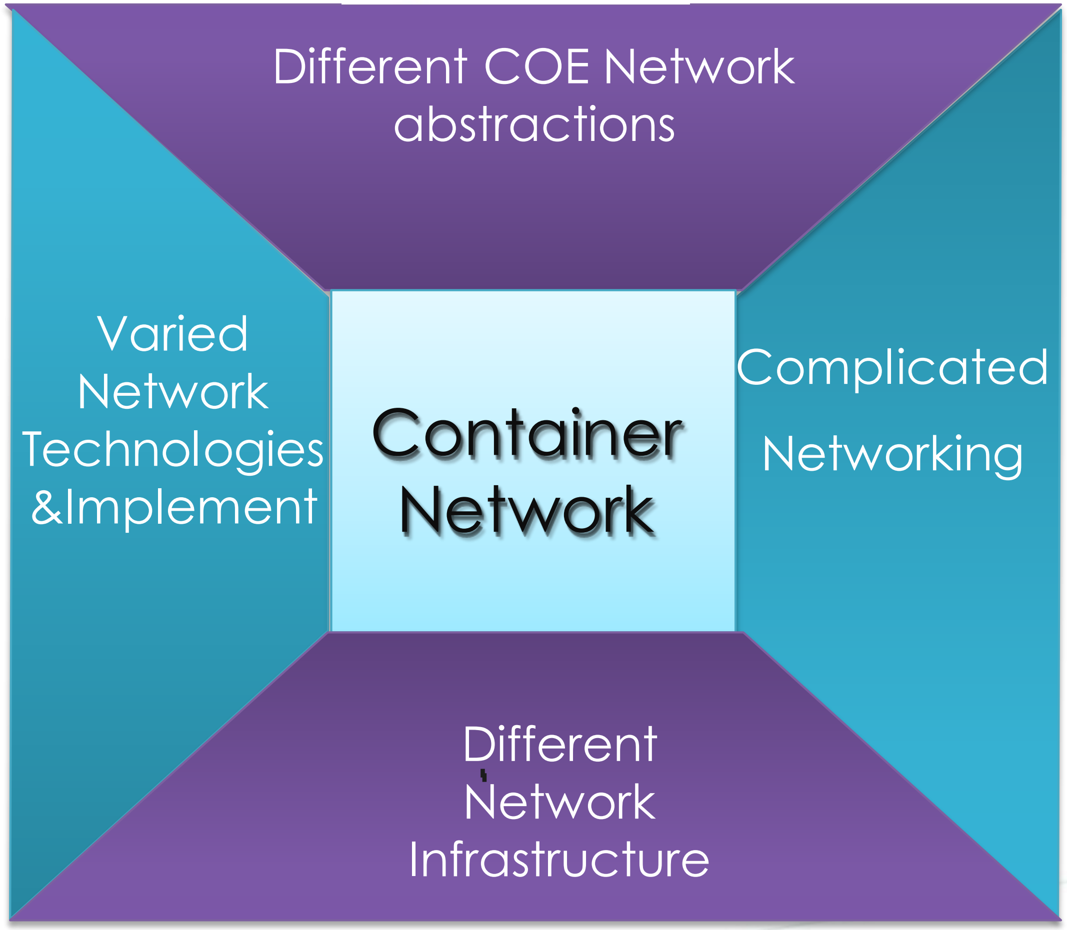
✓Lack of E-to-E monitoring causes no assurance of network quality and difficulties of troubleshooting

✓Virtual network technologies based on software make flexible and customizable monitoring possible

What we face today

kubernetes MESOS swarm

- L2 / L3
- Overlay
- NAT
- VLAN
- BGP



- Multiple tenants
- Multiple Plane
- Performance Isolation
- Security Isolation
- Network Policies

- VM s
- Physical Host

Existing Container Network Solutions

Solution Comparison	Weave	Flannel (CoreOS)	Contiv on ACI (Cisco)	Kuryr@Neutron (Midokura)	Calico (Metaswitch)	iCAN
Basic Networking	L3 Overlay	L2+L3 Overlay	L3 :software Overlay L2: ACI	L2 via vSwitch	L3(BGP)	Flexible L2 or L3
Optimized stack for Container App	Private UDP Tunnel	VXLAN+ Private Tunnel	No	No	Linux IP +BGP	1. Provide high performance tunnel and stack 2. Supported acceleration via customized protocl
Isolation & Security	Multi-tents，APP isolation, crypto	No	Tent isolation and security policies via ACI；support firewall	Rely on Neutron	Rely Linux Capabilities	1. Multi-tents； 2. Support isolation via network and app, basic security； 3. Support firewall
Monitoring	No	No	Just monitor in the physical network, no monitor in the application network	No	No	Provide monitoring capability from end to end
Network SLA	No	No	ACI can provide QoS via EPG; no SLA for App	No	No	support (Proactive)SLA base application demanding and (Reactive SLA)

What is iCAN

- **iCAN(intelligent ContAiner Network)** is an open source project which provides an extensible framework to manage hybrid container network for Cloud Orchestration, define an operational mechanism to address SLA between application and infrastructure.
- Provide flexible framework to work with multiple network components , support on-demanding network plane for multi-tents and micro-services via rich modeling mechanisms.
- Implement multi dimension SLA management and monitoring including bandwidth, latency, drop rate, provide rich network polices for Orchestration with performance isolation and scheduling algorithm.
- Support both CNI and CNM interfaces for most Container Orchestration Platforms like Kubernetes, MESOS.

iCAN Key Features

Agile Framework

✓Support multiple Orchestration Platforms, Kubernetes, Rancher, Mesos
✓Easily Network deployment via templates
✓Selectable components with profiles to support different scenarios and performance

Powerful Monitoring

✓Implement “monitoring on-demand ”and “E-to-E monitoring” based on the topology
✓Facilitate on-demand DSL based troubleshooting
✓Cooperate with the SLA subsystem to assess the SLA quality

Rich Network Support

✓Powerful network component modeling : SNC and Modeling via Yang
✓Rich network schemes, support L2, Overlay, NAT, VLAN, L3, BGP, VPC
✓Accelerated Network Stack

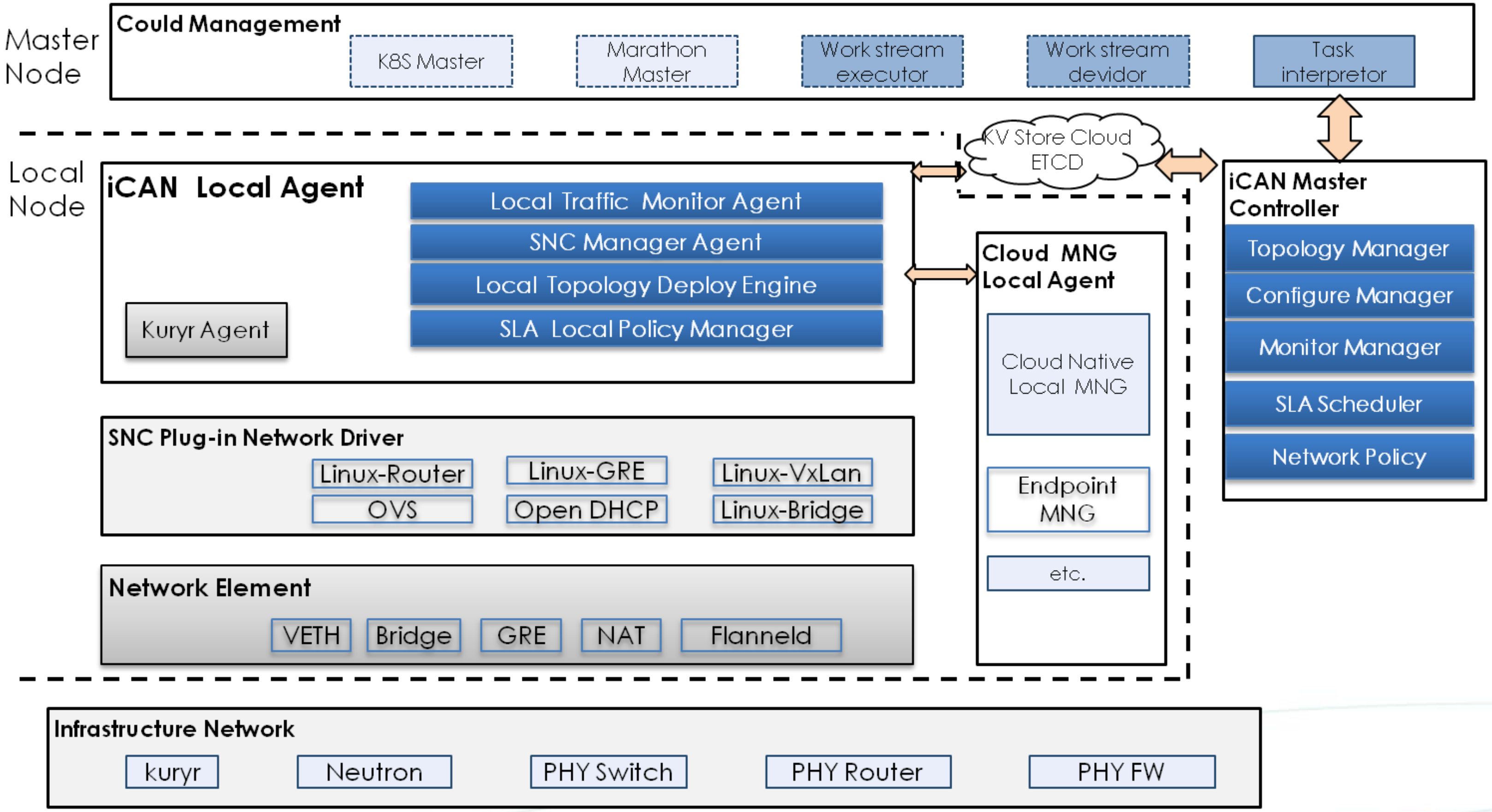
Multi-dimension SLA& Security

✓Performance Isolation with bandwidth, latency, drop rate(Proactive Network SLA and Reactive Network SLA)
✓Security Isolation: VLAN/VXLAN, ACL

iCAN Overall Architecture

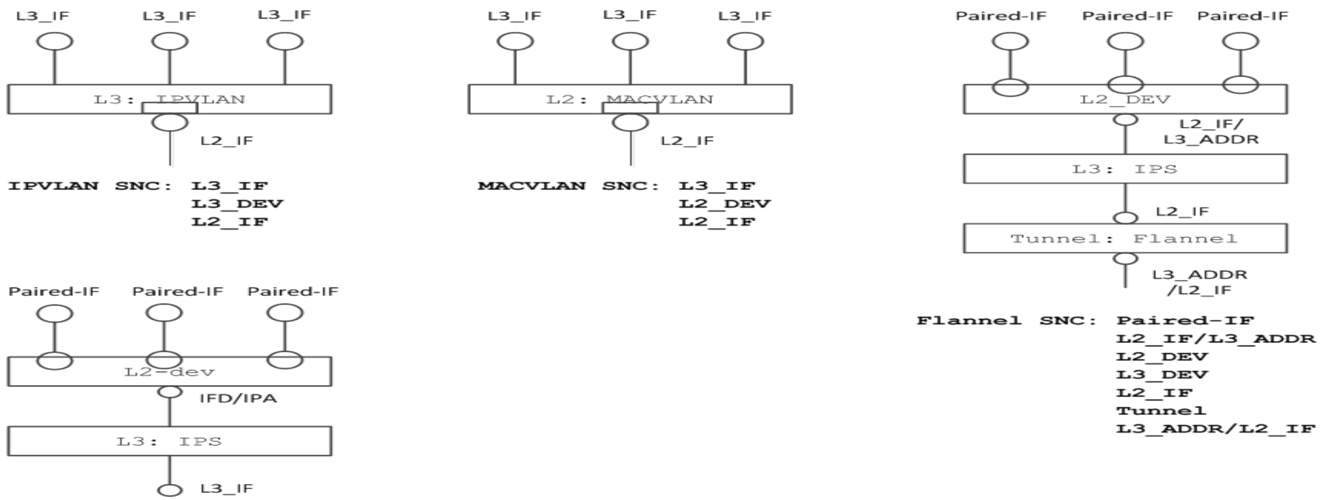
iCAN is composed of Controller Node and Local agent node. Controller node will responsible for communication with orchestration, local node will manage local network and plicies.

Main components include:
iCAN Master Controller :
-Communicate with COE
-Convert network requirement to topologies , policies and configurations through templates
- define network policies , distribute them to each node.
- analyze and trace network failure
-Provide End-to-End network SLA for applications
iCAN Local Agent :
-Configure local network element
-Deploy policies
-Create network with isolation polices
SNC Plug-in Network Driver:
- Support abstract network topology definition to generate container networking data path.



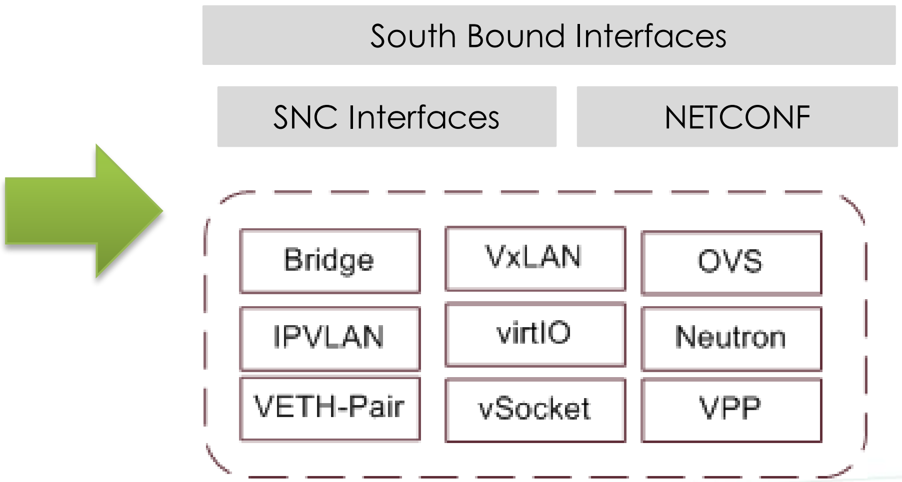
Modeling for Container Network-SNC

- SNC upward links virtual network configuration of deployment template (flexible to make virtual network topo), downward provide united interface of plugin components
- SNC Modeling can simplify network management :
- ✓ Enhance network performance through replacing legacy components with high performance ones ;
- ✓ provide network solution suitable for application according users requirements with profiles ；
- ✓ Customize highly flexible network solution for users;
- ✓ implement global network control and monitoring through the specifications of SNC interfaces ，implement network SLA and optimization.



SNC modeling for Container Network Solutions

Substitute Standard Component freely

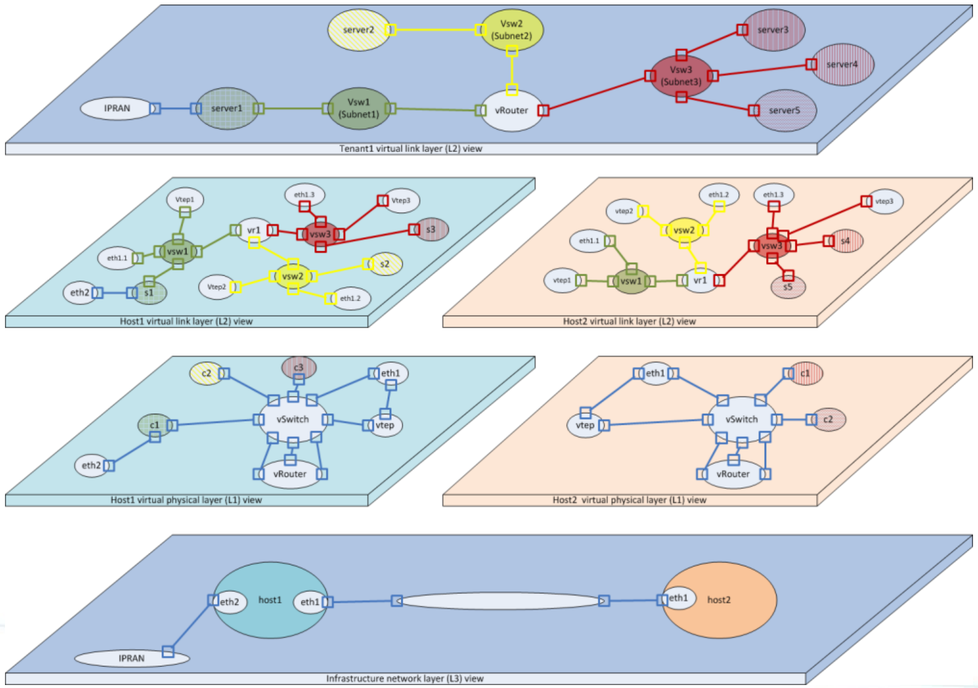


SNC Components List

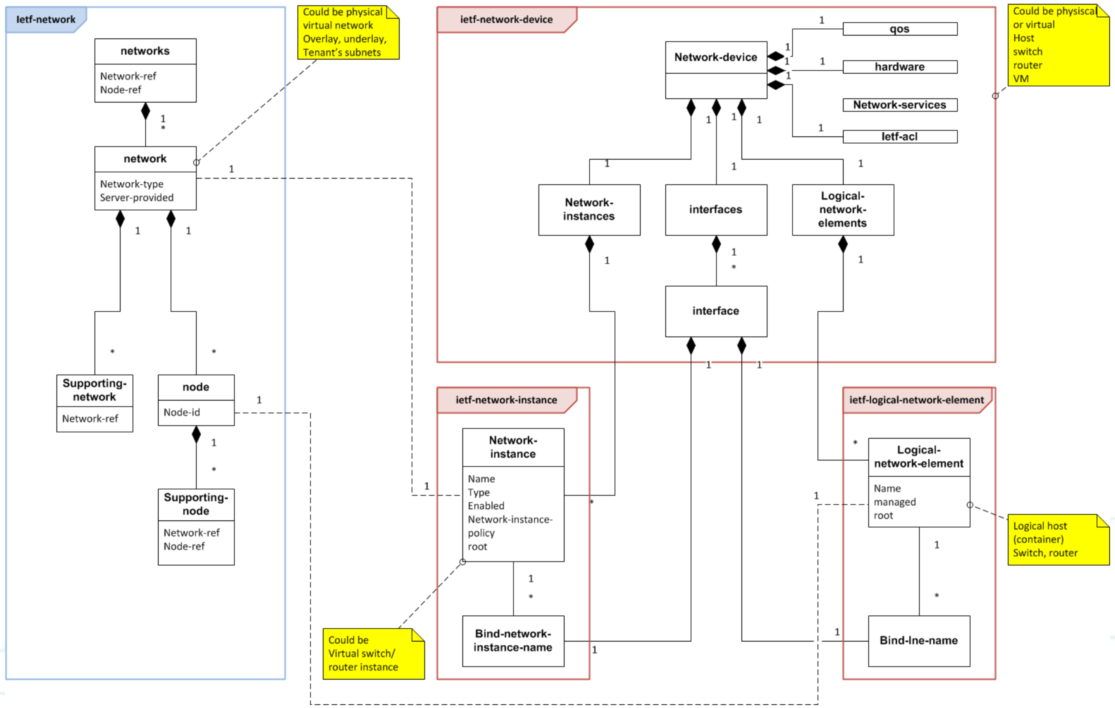
Class	SNC	name	Implementation	Relative SNC	Capability	Operation
Interface	L2_IF	MAC	Eth0, Tap	Port(1:1); L2_DEV(1:n); L3_DEV(1:n)	Explicit; Implicit	Statistics()
	L3_ADDR	IPA	IPv4, IPv6 Addresses	L2_IF(1:n); L3_DEV(1:n)		
	PAIRED_IF	DM_IF	Veth-pair; CETH-Pair	Port(1:1) or Port(2:1)		
Port	Port	Port	vPort	L2_IF(1:1)	Explicit; Implicit;	
Device	L2_DEV	L2_DEV	br; macvlan; ovs;	Port(n:1); L2_IF(n:1);	ACL, QoS, monitor	Filter(port, flow) Ratelimit(port, flow, bw) Shaping(port, flow, bw) GuaranteeBW(port, flow, bw) Prioritize(port, flow, prio) Monitor(port, flow, mon_obj)
	L3_DEV	L3_DEV	IP_Stack; vRouter; IPVLAN	Port(n:1) L3_ADDR(n:1)	ACL, QoS, monitor	
	OpenFlow	OFD	OVS	Port(n:1) L2_IF(n:1) L3_ADDR(n:1)	ACL, QoS, monitor	
	Tunnel	TUN	VXLAN; Flannel; GRE; IPsec	L2_IF(1:1) or L2_IF(2:1) L3_ADDR(1:1) or L3_ADDR(2:1)	Encap, Decap	get_peer_tunnel()
Service	Firewall	FW	Firefly;	Port(n:1) L2_IF(n:1) L3_ADDR(n:1)	NAT	Get_nat_rule(old_flow, &new_flow)
	LB	LB	BigIP, ELB;		LB	Get_lb_rule(old_flow, &new_flow)
Socket	Socket	SK	vSocket			

Modeling for Container Network- YANG

- Node of a network specifies inventories
  - Can be augmented with hardware/acceleration capability and statistical information for resource scheduling
- Links and termination points define network or service topologies
  - Can be augmented with QoS, like level stats
- One network can have one or more supporting networks
- Vertical layering relationships between networks define mapping between layers

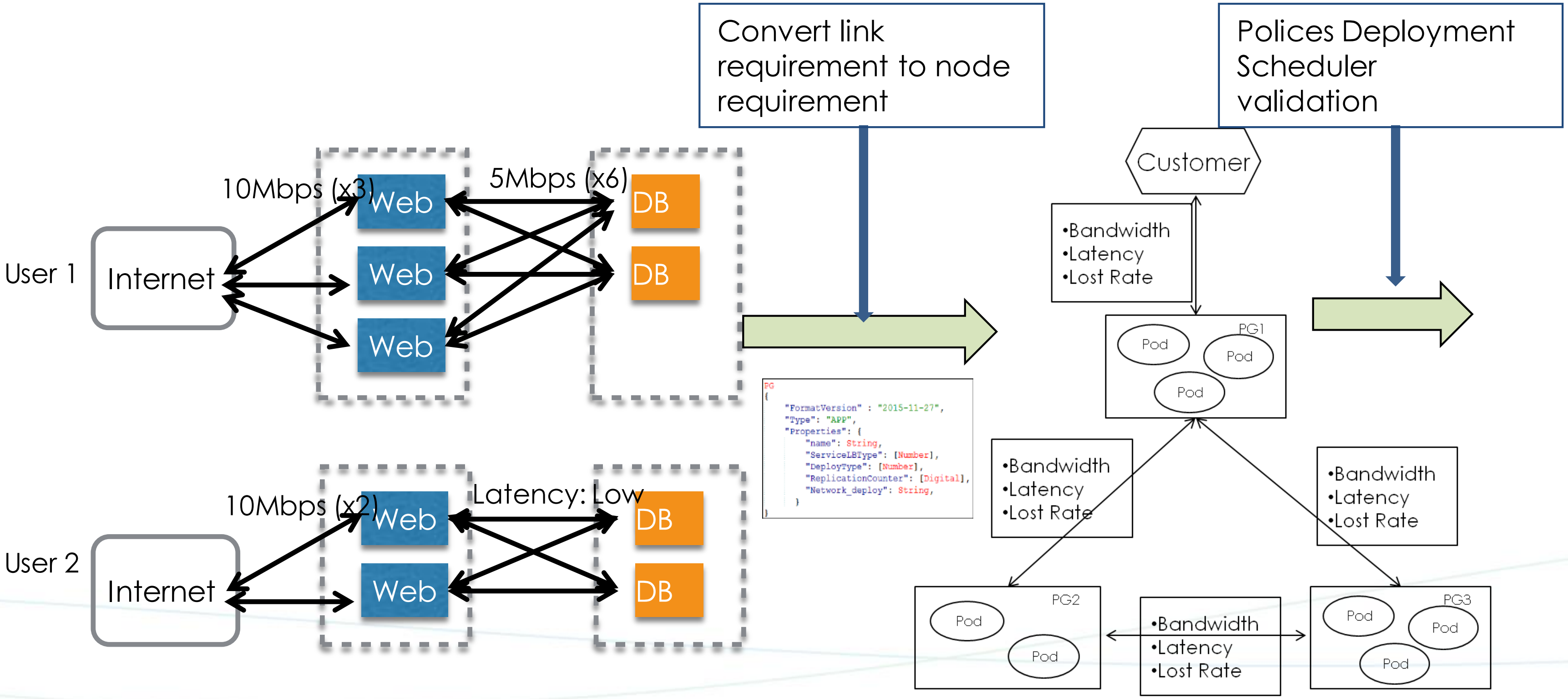


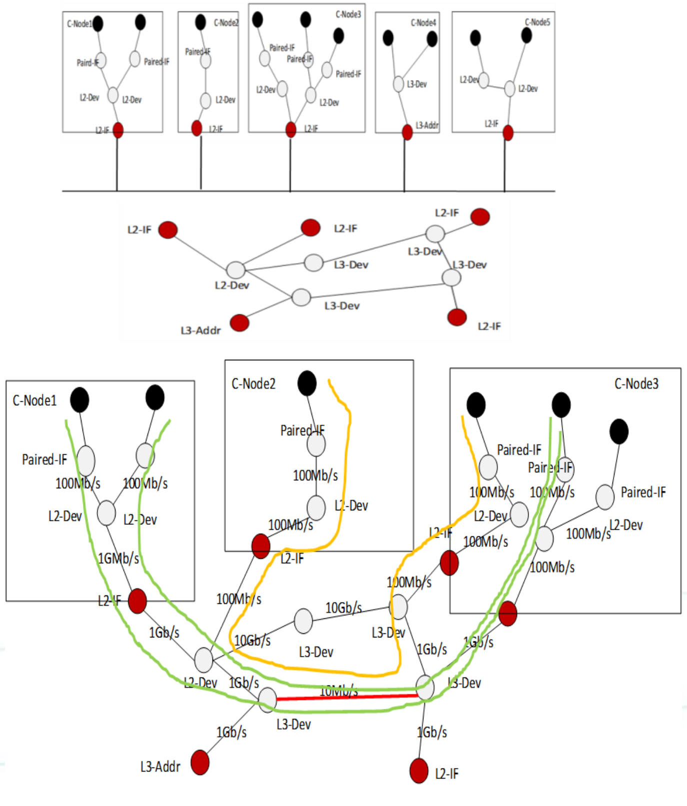
Reference YANG Models for Network Node



Network SLA modeling

- iCAN provides north bound interfaces for orchestration and applications to define their requirements through PG(Pod Group: a group of pods with the same functions), Linking (network requirement between PG) , SLA Service types and Service LB Type.
- Given topology and link bandwidth, evaluate the offers when deploying pods. Essentially a evaluation for pod placement, and validate the deployment.
- 2-Tiers Network topology management Underlay Network (Stable and Predictable) and Overlay Network (Customizable and Dynamic)
- Support: bandwidth, latency and drop rate
  - Bandwidth <5%
  - Latency <10%, more non-deterministic, affected by many factors such as queuing in software switch and hardware, application response, server IO, etc





Monitoring Bases Modeling Network Node

Monitoring Usage:

SLA Monitoring	Network Performance View	Network Topology View

End to End Monitoring in Master Node:

Pod to Pod	Pod to vNic	vNic to vNic	vNic to pNic	pNic to pNic	Tunnel

- E2E Latency
- E2E Bandwidth
- E2E PKT Loss Rate
- Traffic Analysis

Point Monitoring in Agent Node:

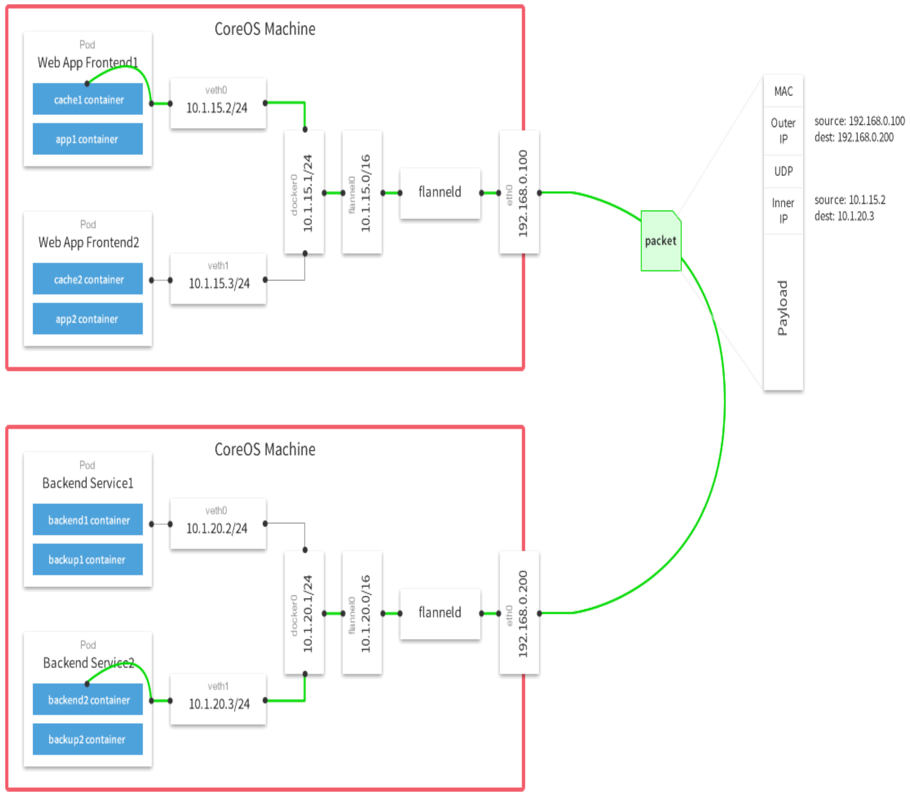
Virtual Interfaces	Virtual Ports	Virtual Network Device	Physical NIC	Physical Network Device

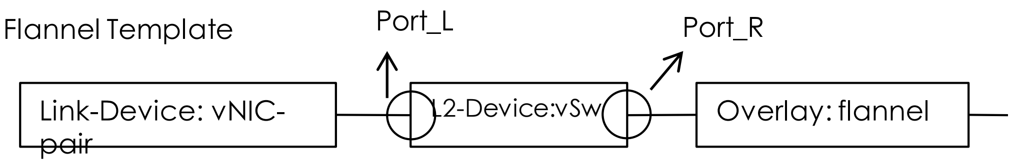
- Bandwidth Capacity
- Current Bandwidth
- Runtime Status
- Traffic Analysis

E2E Monitoring	Monitoring Data Source
E2E Latency	Provide UDP,TCP,ICMP based one way and two ways detection
E2E Bandwidth	Average single point data in central
E2E PKT Loss Rate	Compare single point data in central
Traffic Analysis	IP stack statistic program for local Pods Multiple steps efforts for cross hosts

Point Monitor Item	Monitoring Data Source
Bandwidth Capacity	•Between vNIC and pNIC, maximum is pNic Speed •Between vNic, no fixed upper limitation. Can calculate in static mode
Current Bandwidth	Single point interface RX/TX packets , bytes
Runtime Status	Single point interface RX/TX errors, dropped, overrun
Traffic Analysis	Traffic filter (collecting through enable all vPorts)

Case Study: Support with Flannel via SNC






```
== High-level topology:
 +---+      +--------------------+     +----------+      +----------------------+
 | C <------| Link:VNIC-pair |-----> L2:SW <------| Overlay:Flannel  |------>
 +---+      +--------------------+     +----------+      +----------------------+
```

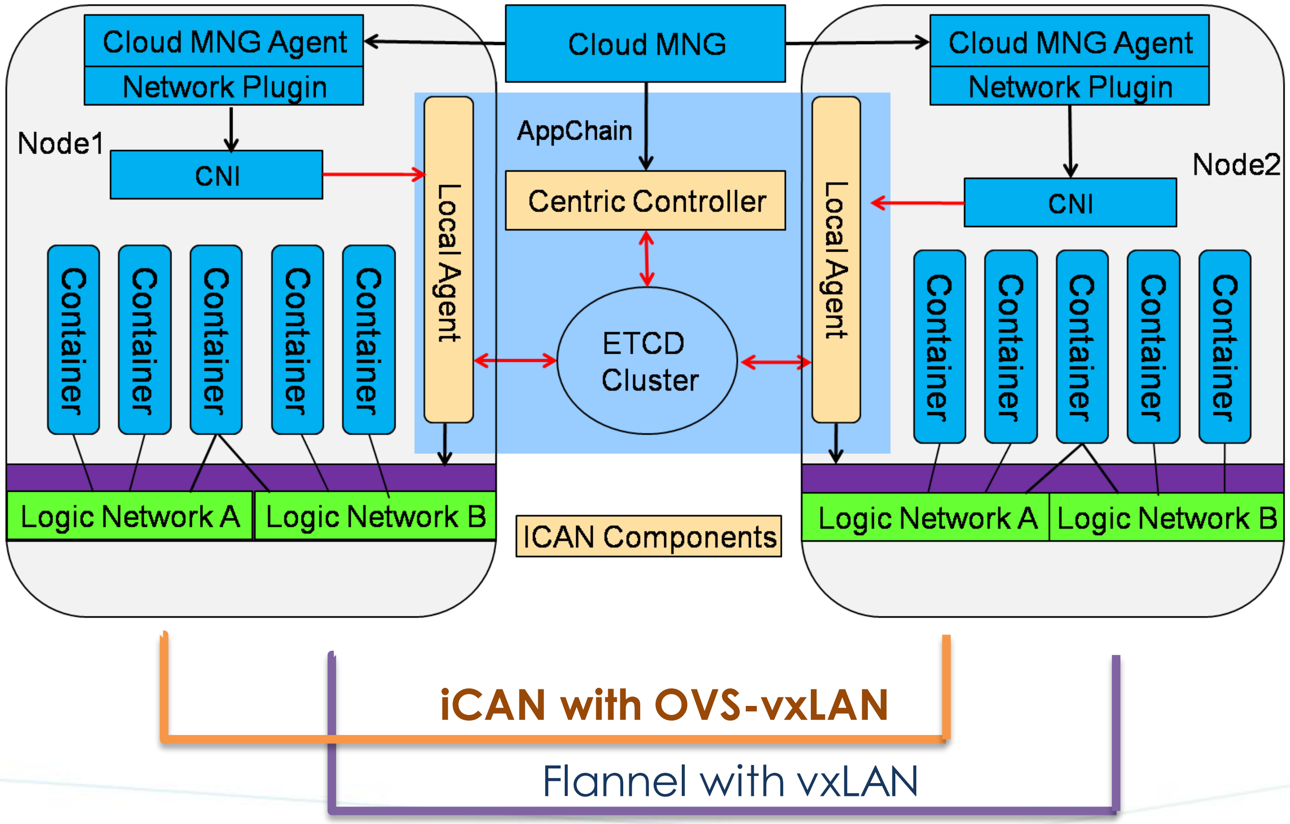
> interface
< port

== Operating abstraction:
- CreateSubnet() -- get subnet information via etcd API
- L2:SW.CreateDevice() => "l2_sw_dev"
- L2:SW.CreatePort(port_L)
- L2:SW.CreatePort(port_R)
- Overlay:Flannel.CreateDevice() => "flannel_dev"
- Overlay:Flannel.Connect(flannel_dev.inf_L, l2_sw_dev.port_R)
- Overlay:Flannel.Connect(flannel_dev.inf_R, eth0)
- Link:vNIC-pair.CreateDevice() => "link_dev"
- Link:vNIC-pair.Connect(link_dev.inf_R, l2_sw_dev.port_L)

```
SNC interfaces:
/* L2:SW device definition */
{
   /* members */
   string port[];

   /* methods */
   CreateDevice(); // creat L2:SW device
   CreatePort(string port_name);
}

/* Overlay:Flannel device definition */
{
   /* members */
   string inf_L;
   string inf_R;

   /* methods */
   CreateDevice();
   Connect(string inf, string port);
}

/* Link:VNIC-pair device definition */
{
   /* members */
   string inf_L;
   string inf_R;

   /* methods */
   CreateDevice();
   Connect(string inf, string port)
}
```

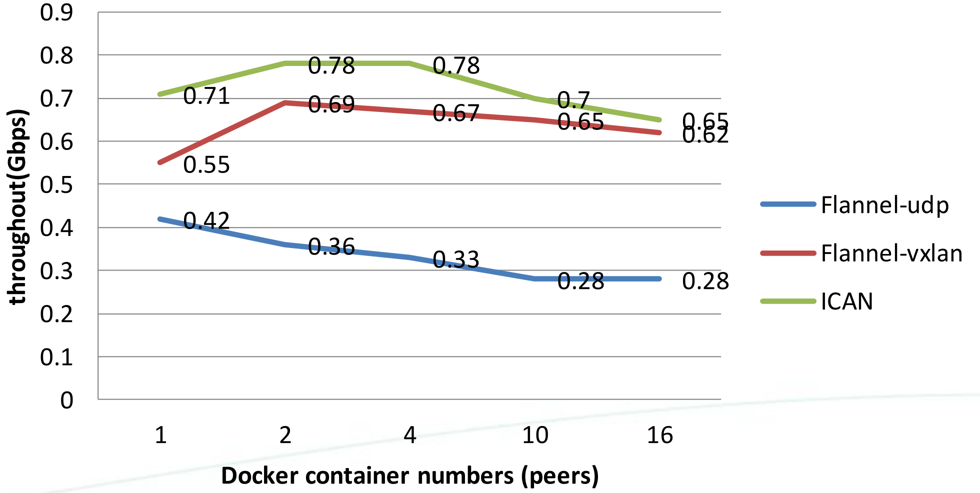
Case Study: Deploy Cluster with iCAN

- SNC based, each node deploys different network components via iCAN framework
- High Performance: 10% higher throughput than flannel without optimization

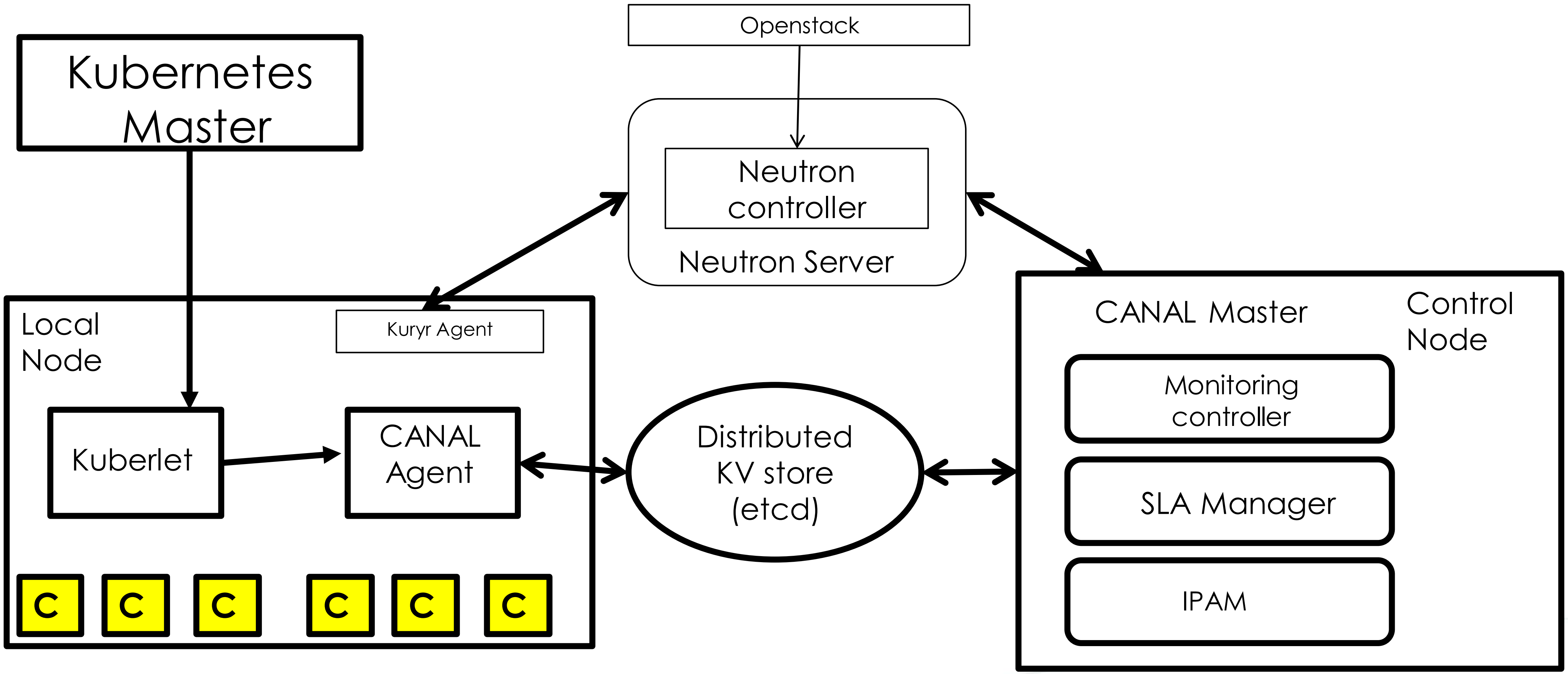


- *Remark: iCAN use OVS-VxLAN, while Flannel employ udp-private and kernel VxLAN Tunnel;*

512-bytes packet throughput
base on cross vm in same host



iCAN Control Plane Integrated with Openstack



Installation and Deployment

- Download:
  - git clone https://github.com/Huawei/iCan

THANK YOU