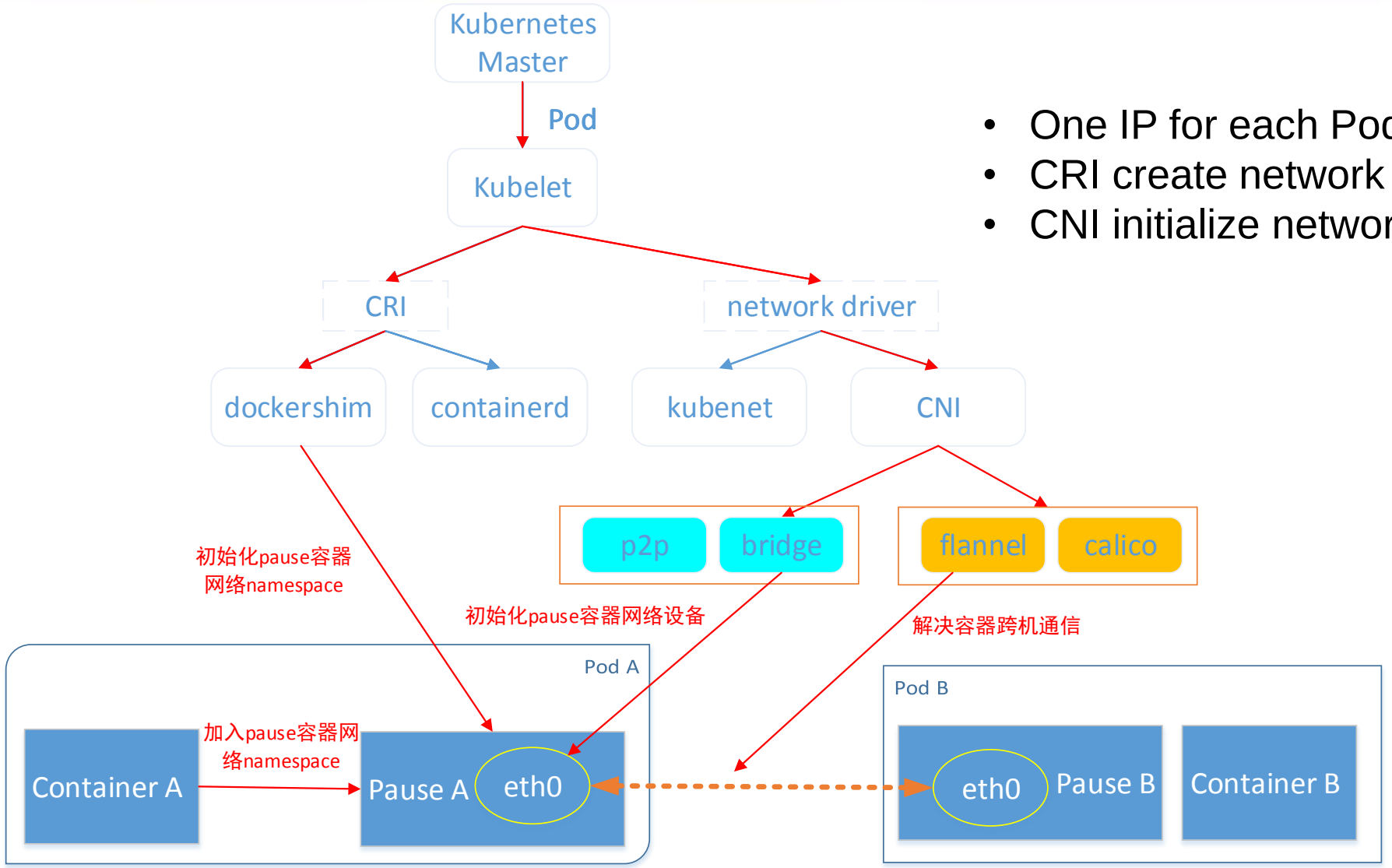
KubeCon | CloudNativeCon

China 2018

Network QoS Support for Kubernetes Applications

Jun Du Huawei Cloud

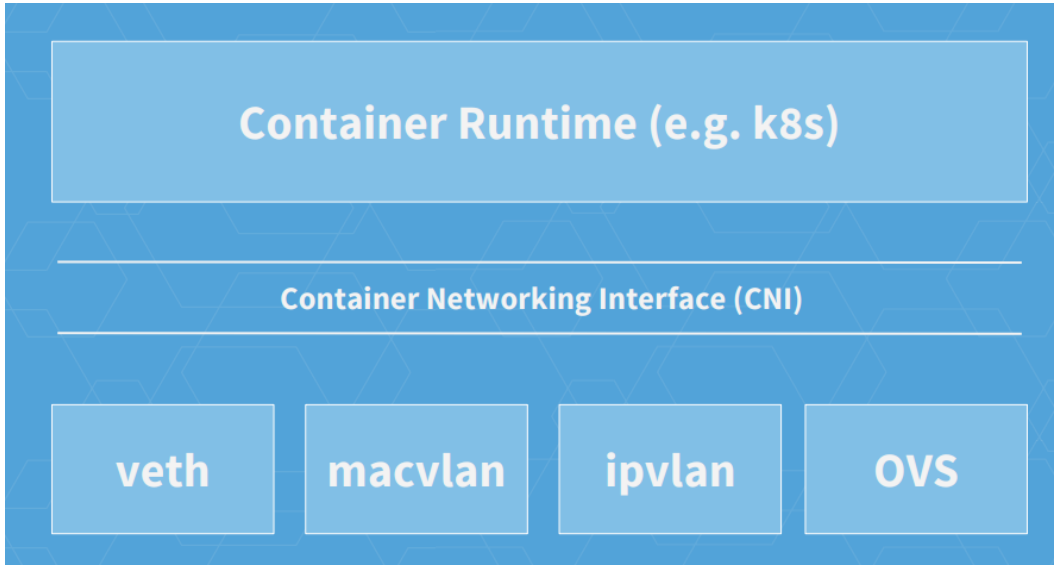
Kubernetes Networking Overview



- One IP for each Pod
- CRI create network namespace
- CNI initialize network device

CNI: Container Network Interface

- Standard of linux container network
- Configure container interfaces using JSON
- Two kinds of interfaces：
  - configure network – invoked when create container
    AddNetwork(net NetworkConfig, rt RuntimeConf) (types.Result, error)
  - clean up network – invoked when delete container
    DelNetwork(net NetworkConfig, rt RuntimeConf) error



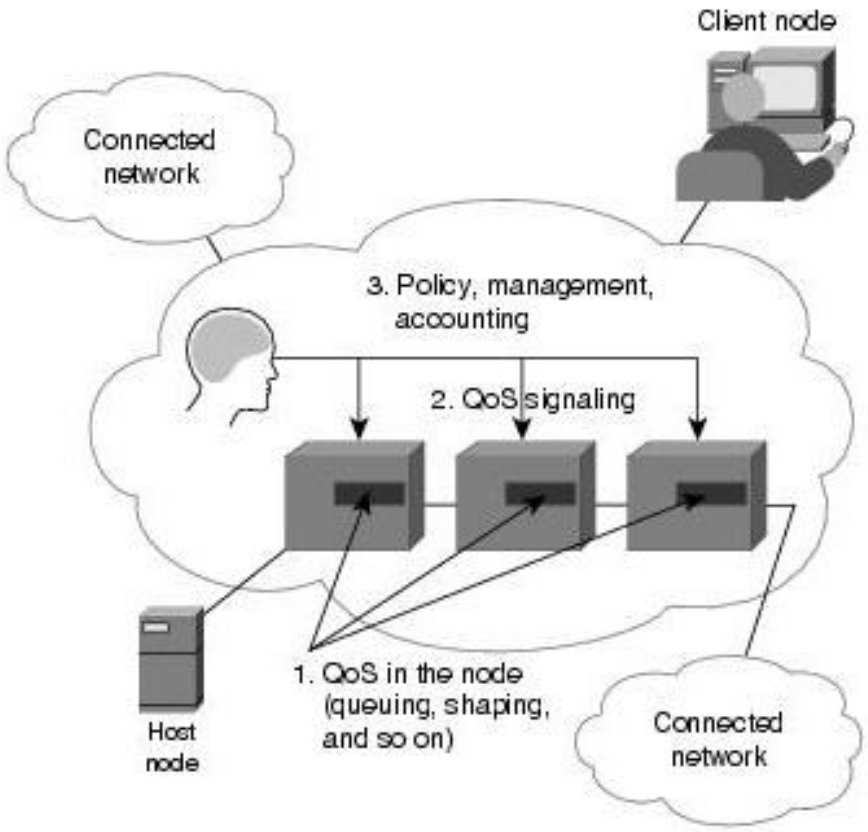
- Bridge
- PTP
- IPVLAN
- MACVLAN
- VLAN
- PORTMAP

flannel
PROJECT CALICO
weaveworks
Google Cloud Platform
nuagenetworks
ROMANA
aws
Contiv
CNI-Genie
OVS Open vSwitch
OPENCONTRAIL
cilium
KUBE ROUTER

Why Need Network QoS Support

- **For Users:**
  - Applications should have the ~SAME performance in Cloud
  - Do NOT want to live with the *noisy* neighborhood
- **For Cloud Providers:**
  - Need a way to isolate different tenants or applications
  - Need a way to handle network flood
  - Charge with different QoS level
- **For Kubernetes:**
  - Better support for voice/video streams, IoT scenarios etc.
  - Handle well even if *scheduling* result is not pretty good
  - Part of multi-tenancy
  - Deprecate *Kubenet*

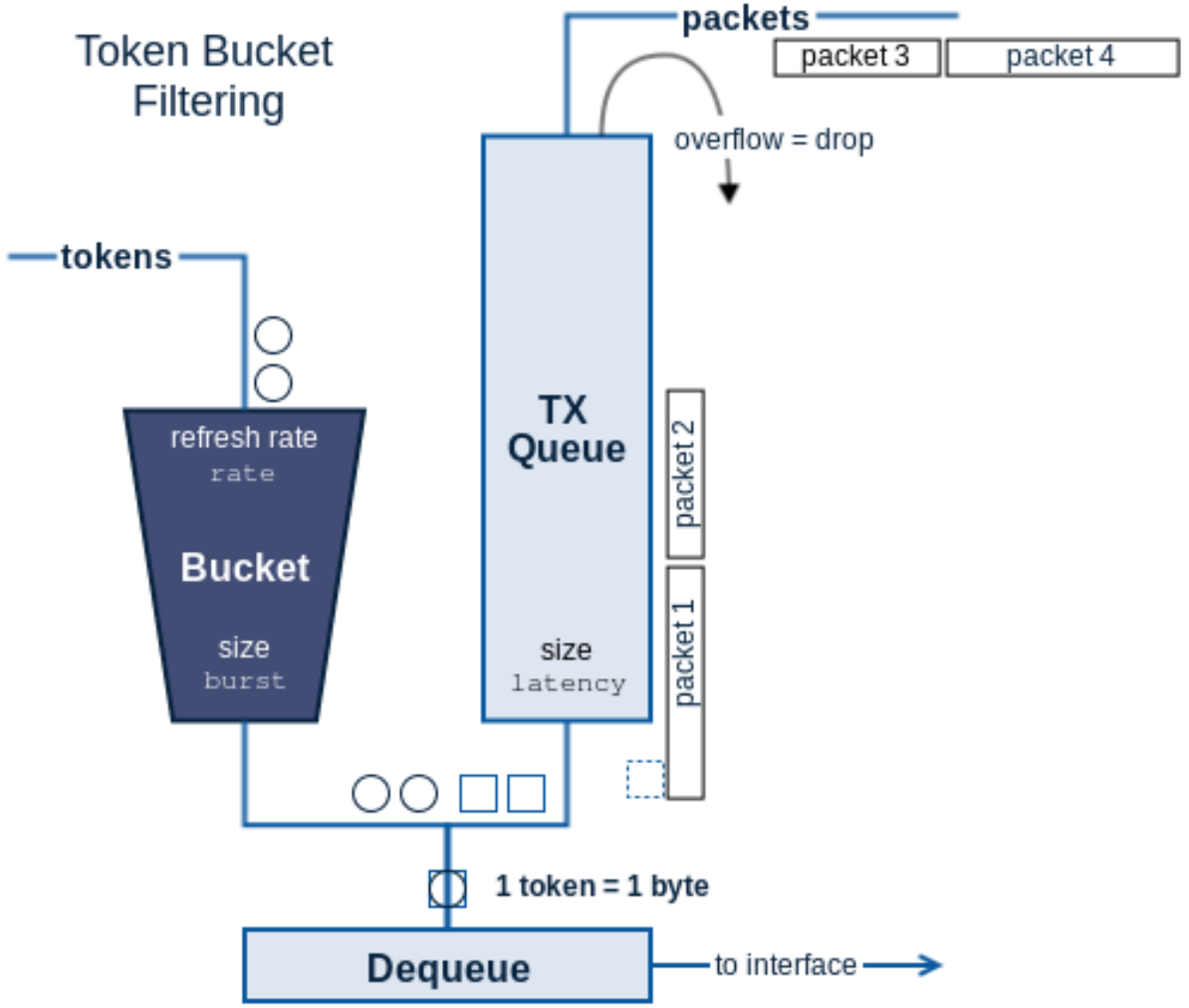
Basic Network QoS Implementations



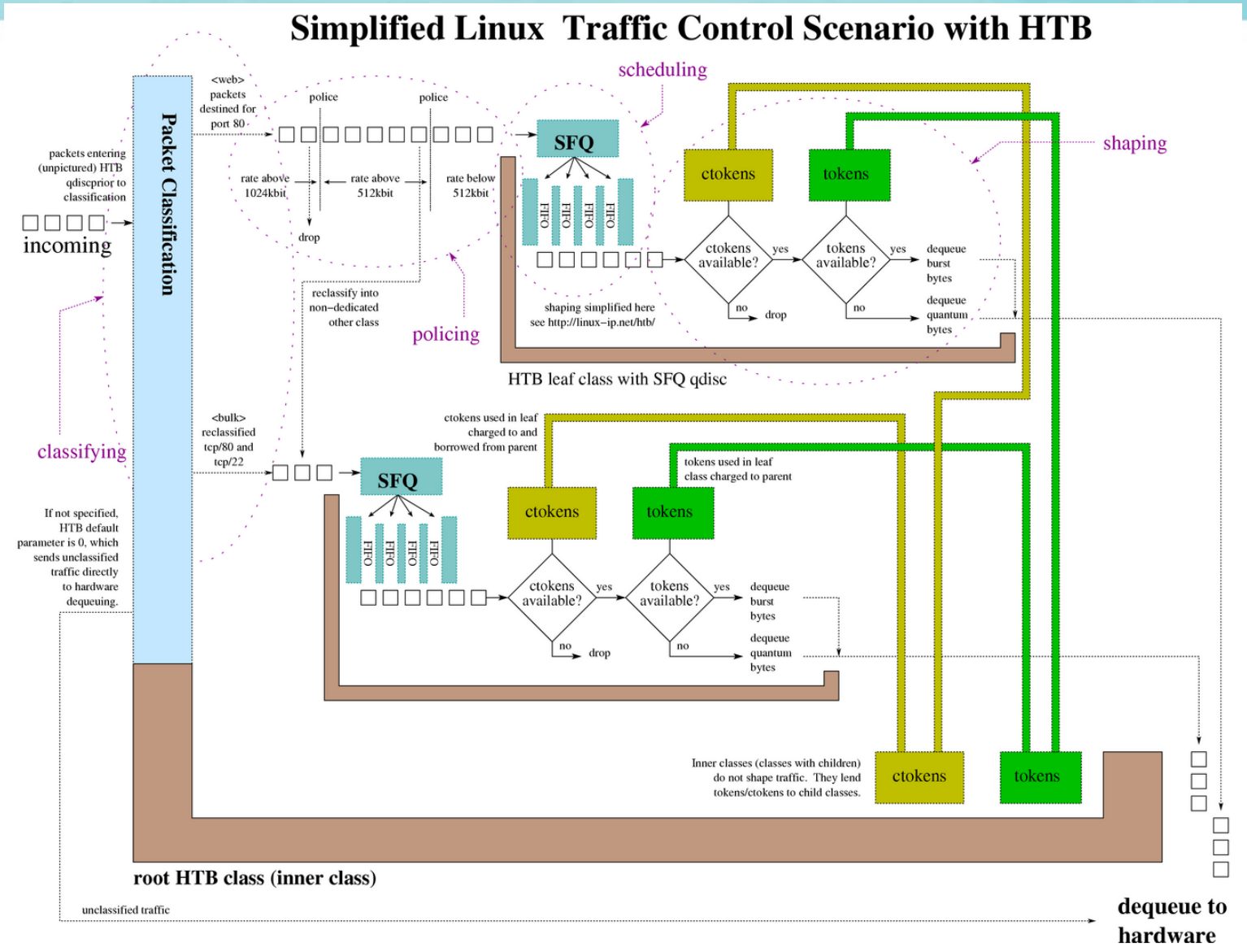
- QoS policy, management, and accounting functions to control and administer end-to-end traffic across a network
- QoS identification and marking techniques for coordinating QoS from end to end between network elements
- QoS within a single network element (for example, queuing, scheduling, and traffic-shaping tools)

QoS within a Network Element

- Congestion control
- Queue management
- Link efficiency
- ***Traffic shaping and policing***



QoS in Linux with TC and Filters



Glossary:

SFQ: Stochastic Fairness Queueing

HTB: Hierarchical Token Bucket

Linux TC Sample Commands


```
export POD_IP=172.17.0.4
export DLD_CLASS=1
tc qdisc add dev cni0 root handle 1: htb default 30
tc class add dev cni0 parent 1: classid 1:${DLD_CLASS} htb rate 10Mbit
tc filter add dev cni0 protocol ip parent 1:0 prio 1 u32 match ip dst ${POD_IP}/32 flowid 1:${DLD_CLASS}
```

cni0 1:

1:1 # 10Mb/s 1:30默认

CNI Bandwidth Plugin

- Configure Linux's Traffic control (tc) subsystem
- Configures a token bucket filter (tbf) queuing discipline (qdisc) on both ingress and egress traffic
- Creates an Intermediate Functional Block device (ifb) to redirect packets from the host interface

- Applies traffic shaping to interfaces created by previously applied plugins

Sample Config of Bandwidth Plugin


```
{
  "cniVersion": "0.3.1",
  "name": "mynet",
  "plugins": [
    {
      "type": “bridge", // can be ptp as well
      "ipam": {
          "type": "host-local",
          "subnet": "10.0.0.0/24"
      },
    },
    {
      "name": "slowdown",
      "type": "bandwidth",
      "ingressRate": 123,
      "ingressBurst": 456,
      "egressRate": 123,
      "egressBurst": 456
    }
  ]
}
```

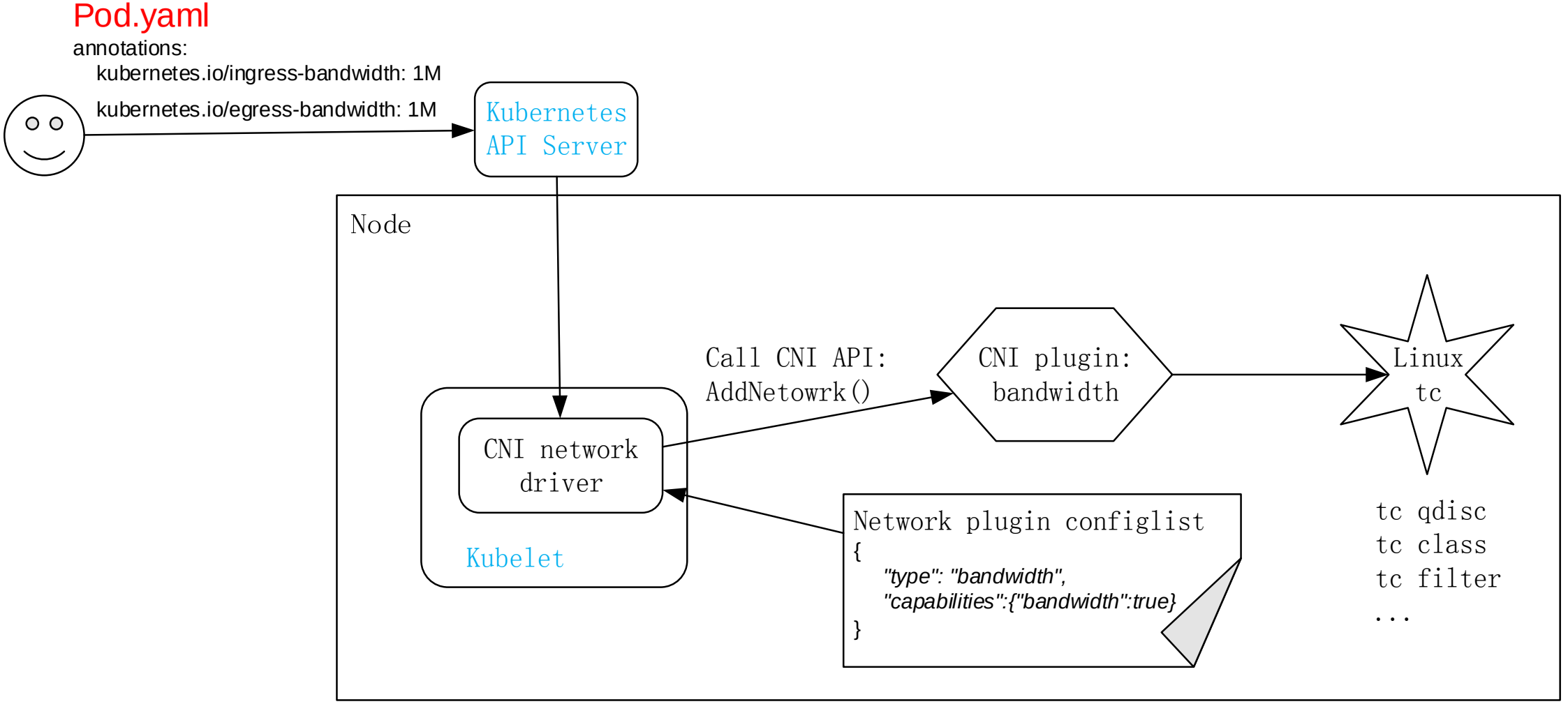
Integrating With Kubernetes

- Kubelet runs with any network driver: cni or kubenet
- Configure Pod's annotations to limit ingress/egress bandwidth rate
- For CNI:
  - Make sure bandwidth plugin binary exists in CNI plugins directory (/opt/cni/bin)
  - Configure enabling traffic shaping in network plugin config file (/etc/cni/net.d/10-caclico.confilist)

```
apiVersion: v1
kind: Pod
metadata:
  name: iperf
  annotations:
    kubernetes.io/ingress-bandwidth: 1M
    kubernetes.io/egress-bandwidth: 1M
spec:
  containers:
  - name: iperf
    image: moutten/iperf
```

```
{
  "name": "k8s-pod-network",
  "cniVersion": "0.3.0",
  "plugins": [
    {
      "type": "calico",
      "datastore_type": "kubernetes",
      .....
    },
    {
      "type": "bandwidth",
      "capabilities": {"bandwidth": true}
    }
  ]
}
```

Workflow of Limit Pod's Bandwidth



K8S Support Traffic Shaping in v1.12

Huawei team will keep contributing…

kubernetes

Documentation Blog Partners Community Case Studies English v1.12

Support traffic shaping

The CNI networking plugin also supports pod ingress and egress traffic shaping. You can use the official bandwidth plugin offered by the CNI plugin team or use your own plugin with bandwidth control functionality.

If you want to enable traffic shaping support, you must add a `bandwidth` plugin to your CNI configuration file (default `/etc/cni/net.d`).

```
{
  "name": "k8s-pod-network",
  "cniVersion": "0.3.0",
  "plugins": [
    {
      "type": "calico",
      "log_level": "info",
      "datastore_type": "kubernetes",
      "nodename": "127.0.0.1",
      "ipam": {
        "type": "host-local",
        "subnet": "usePodCidr"
      },
      "policy": {
        "type": "k8s"
      },
      "kubernetes": {
        "kubeconfig": "/etc/cni/net.d/calico-kubeconfig"
      }
    },
    {
      "type": "bandwidth",
      "capabilities": {"bandwidth": true}
    }
```

Demo

https://asciinema.org/a/L60lAcHknt9BhdsMDNAr9oJ2q

Future Work

- ***Burst*** rate support in Kubernetes API
  - Given the burst rate support in CNI bandwidth plugin side
- Support traffic shaping in CNI third party plugins(calico, weave…)
  - weave already got a plan
- Explore a way to re-configure when a Pod is running
- If ALL network plugins supports traffic shaping, should we move it out of annotations?
- More Flexible policies instead of static configuration
- Windows Container traffic shaping?

KubeCon | CloudNativeCon

China 2018