容器 SDN 技术与微服务架构实践

大纲

- SDN
  - 容器与网络
- 开源容器网络方案
  - Flannel
  - Calico
  - Weave
- 七牛实践与案例分析

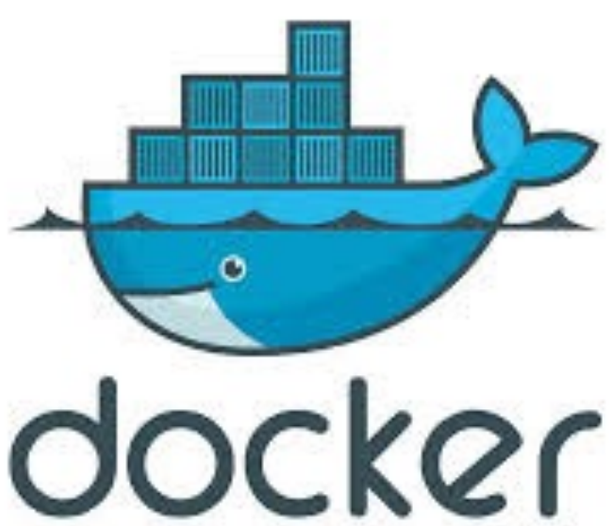
SDN

软件定义**网络**

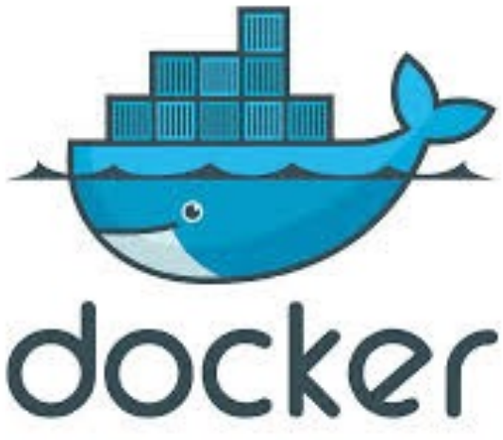
网络结构的**灵活性**

容器与网络

- 相比传统虚机
- 每个容器的职能更少
- 容器之间的关系更加复杂
- 网络端点数量上升
- 容器的生命周期更短



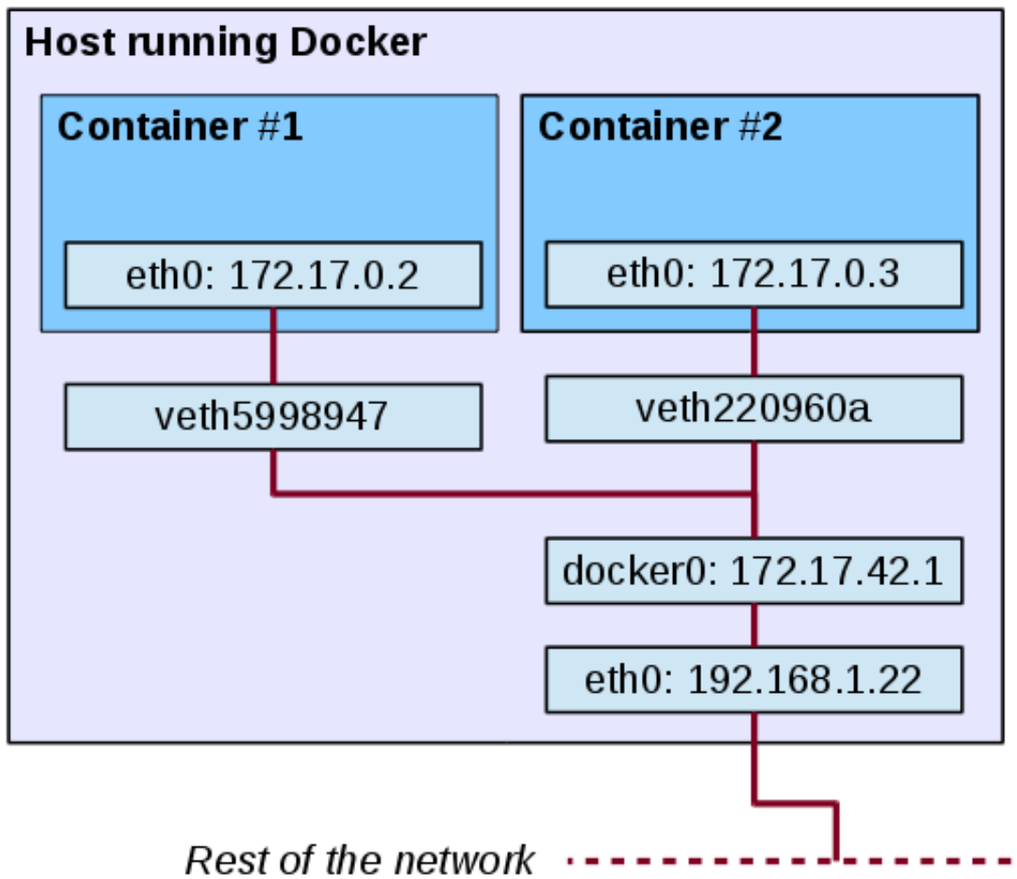
Pipework



PROJECT CALICO

Docker Bridge

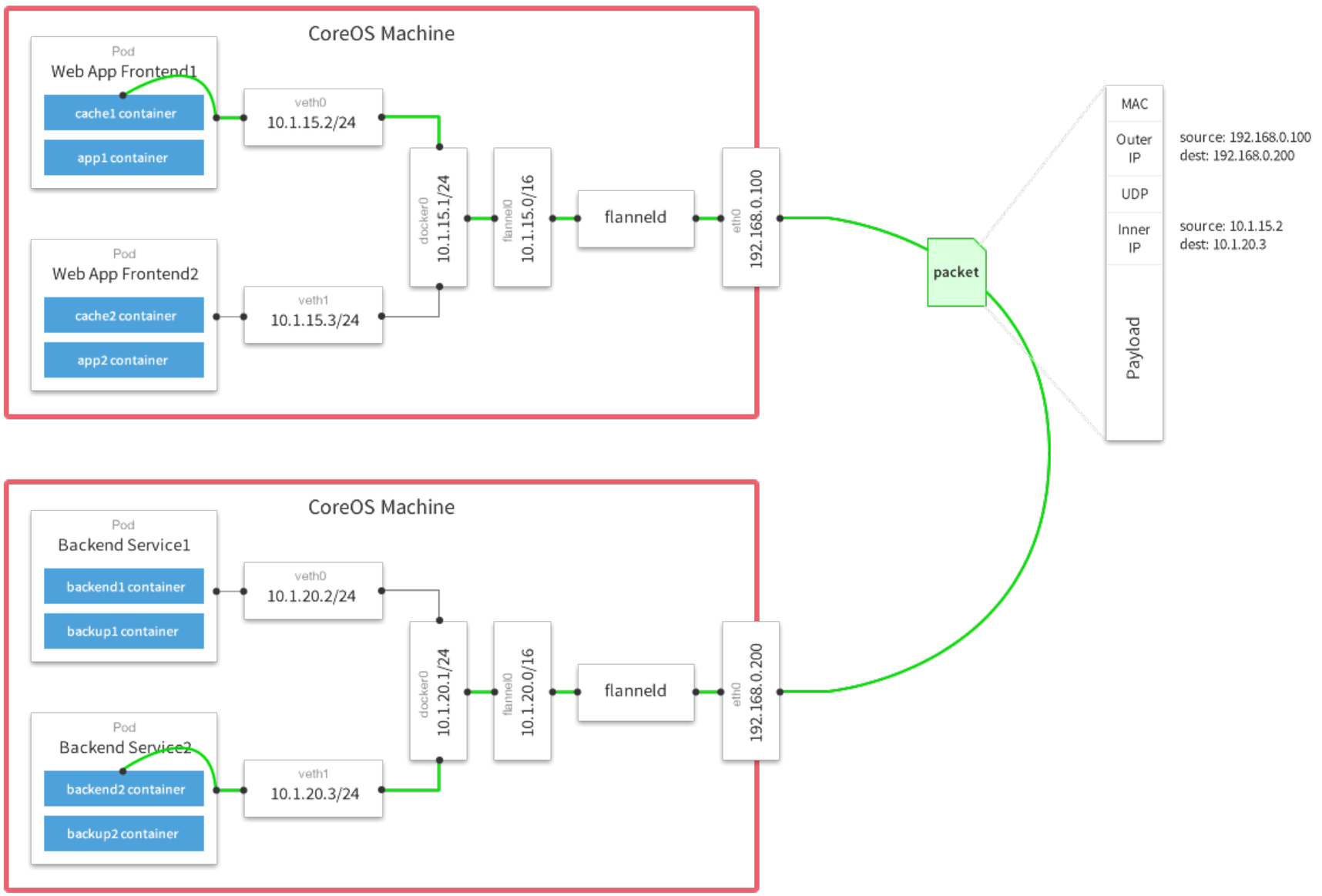
- 原理：
  - Linux bridge 二层包交换
  - iptables NAT 地址转换
  - ip_forward 路由转发



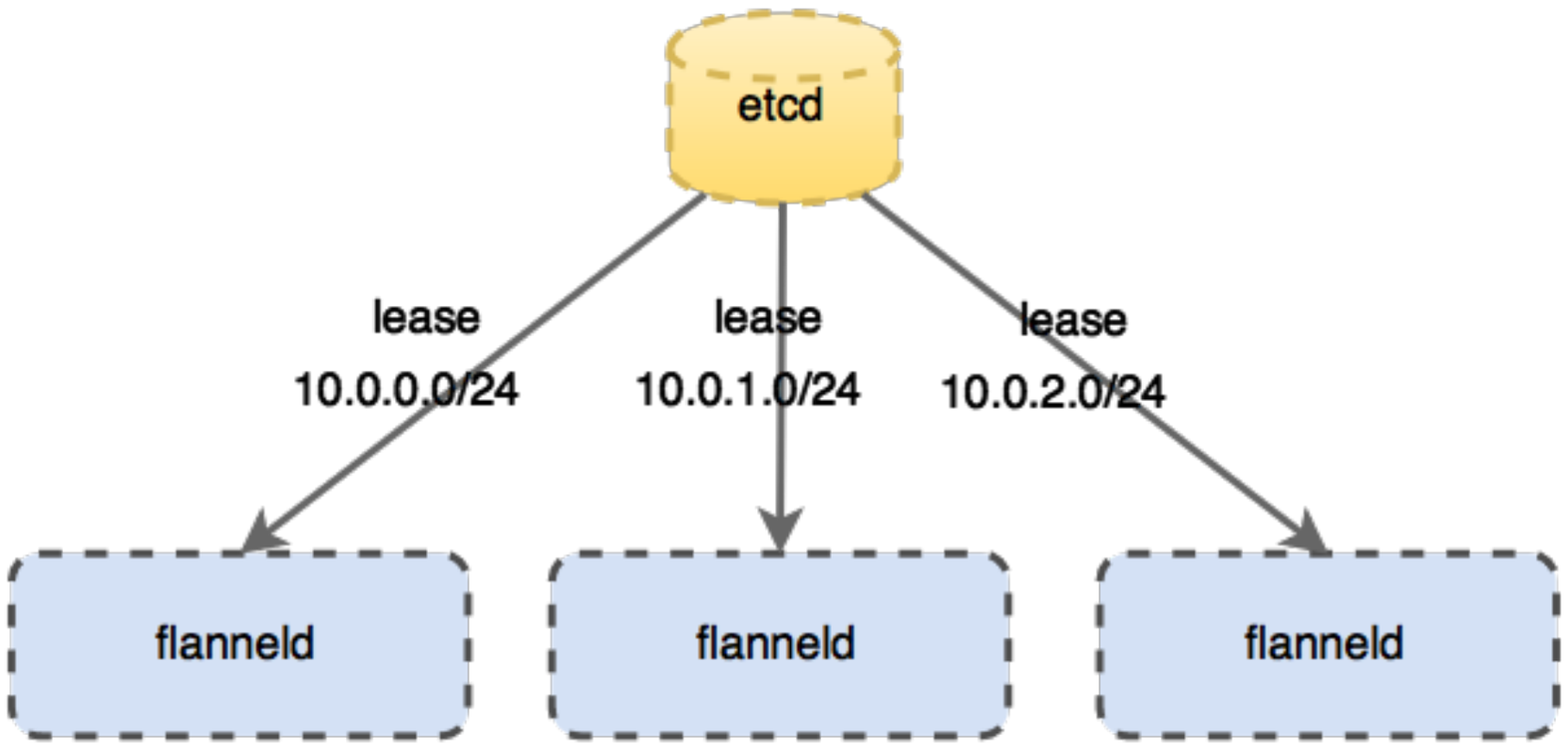
Docker Bridge

- 重启后 IP 发生变化
- IP 无法跨主机灵活迁移
- NAT 模式下，跨主机间通信会隐藏地址信息
- NAT 性能损耗
- Bridge 内流量难以精细隔离
- 需额外管理端口冲突
- blahblah…

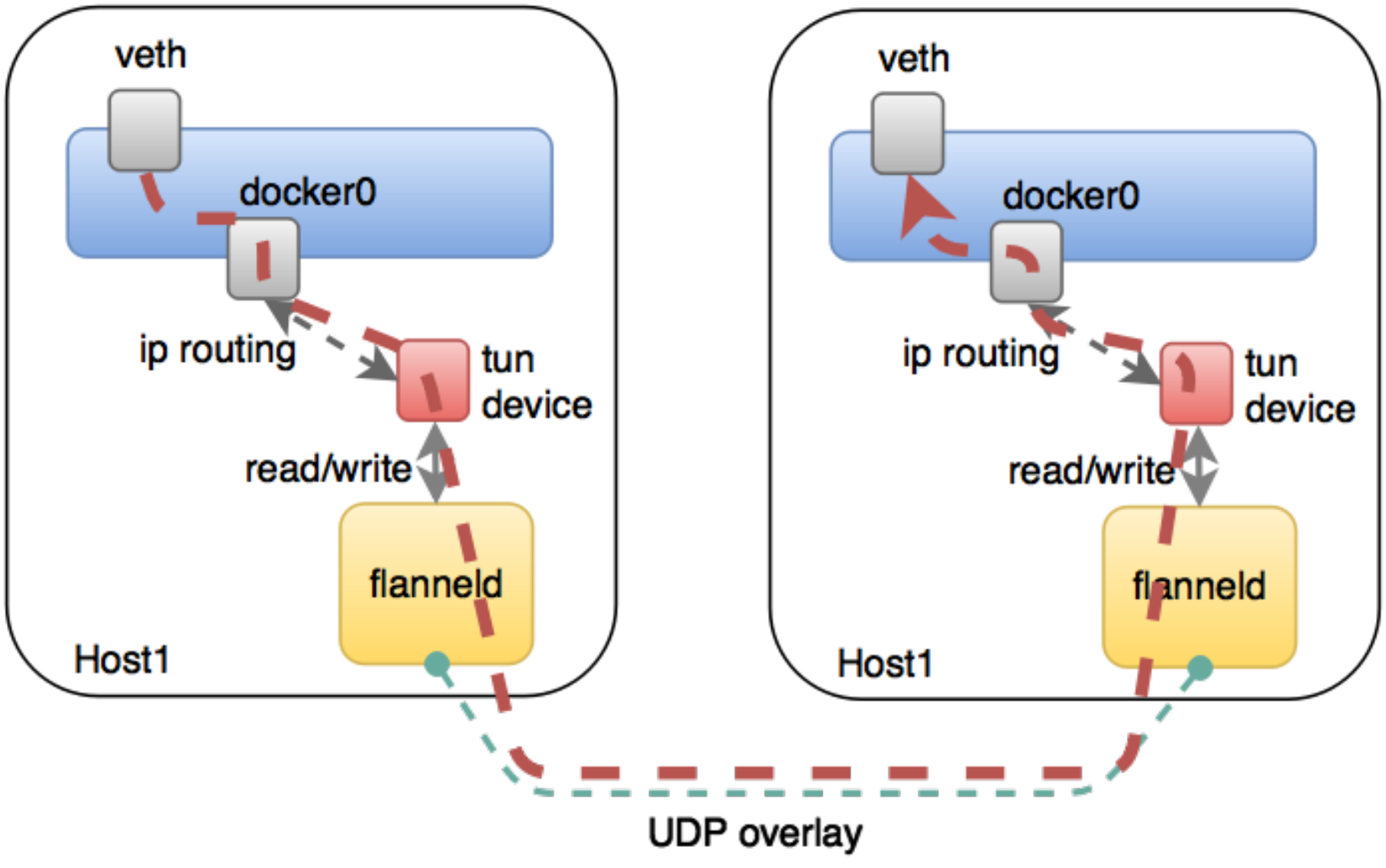
Flannel


CoreOS Machine
Pod
Web App Frontend1
cache1 container
app1 container
veth0
10.1.15.2/24
Pod
Web App Frontend2
cache2 container
app2 container
veth1
10.1.15.3/24
docker0
10.1.15.1/24
flannel0
10.1.15.0/16
flanneld
eth0
192.168.0.100
packet
MAC
Outer IP
UDP
Inner IP
Payload
source: 192.168.0.100
dest: 192.168.0.200
source: 10.1.15.2
dest: 10.1.20.3
CoreOS Machine
Pod
Backend Service1
backend1 container
backup1 container
veth0
10.1.20.2/24
Pod
Backend Service2
backend2 container
backup2 container
veth1
10.1.20.3/24
docker0
10.1.20.1/24
flannel0
10.1.20.0/16
flanneld
eth0
192.168.0.200

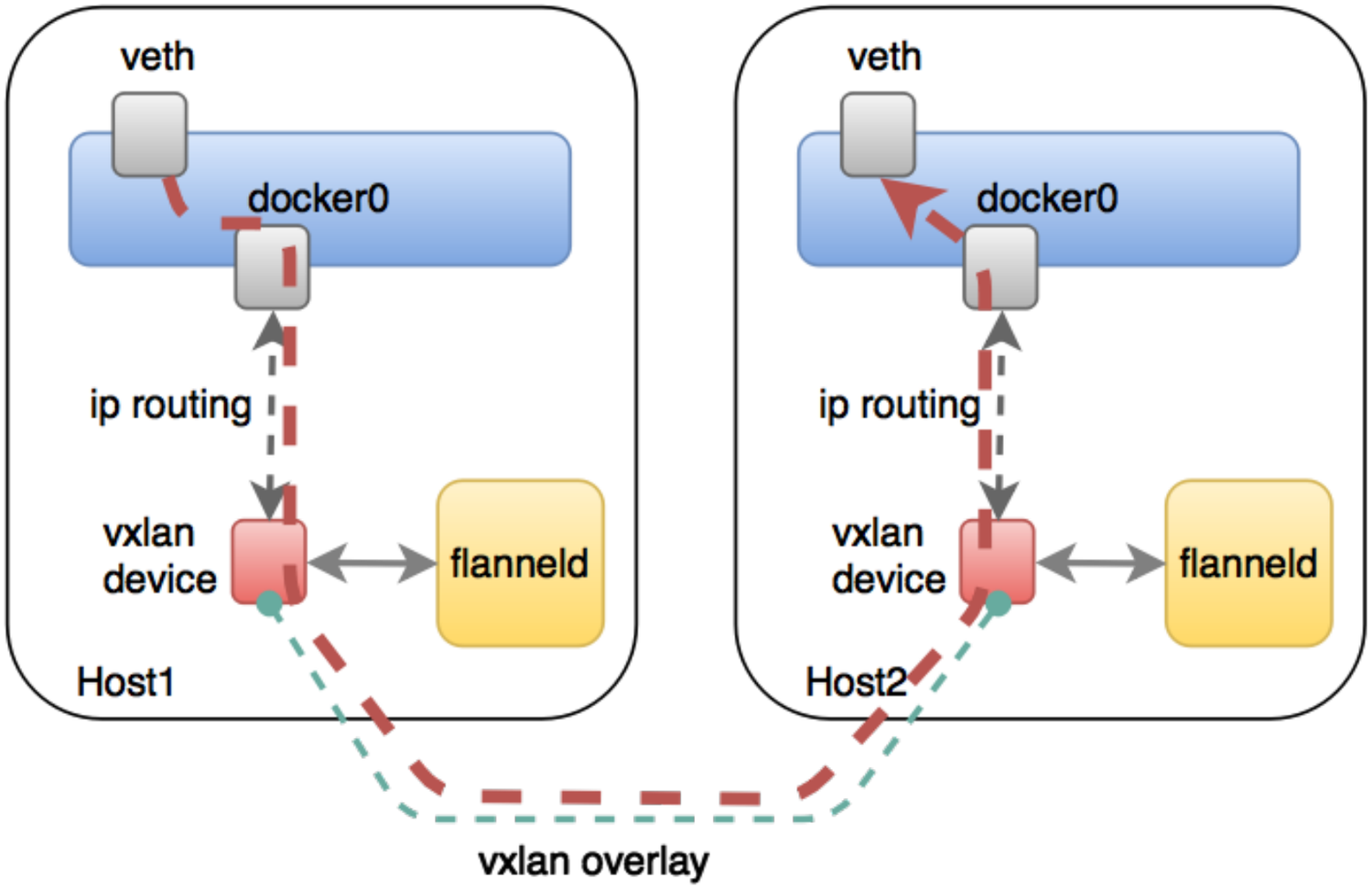
控制平面


etcd
lease
10.0.0.0/24
lease
10.0.1.0/24
lease
10.0.2.0/24
flanneld
flanneld
flanneld

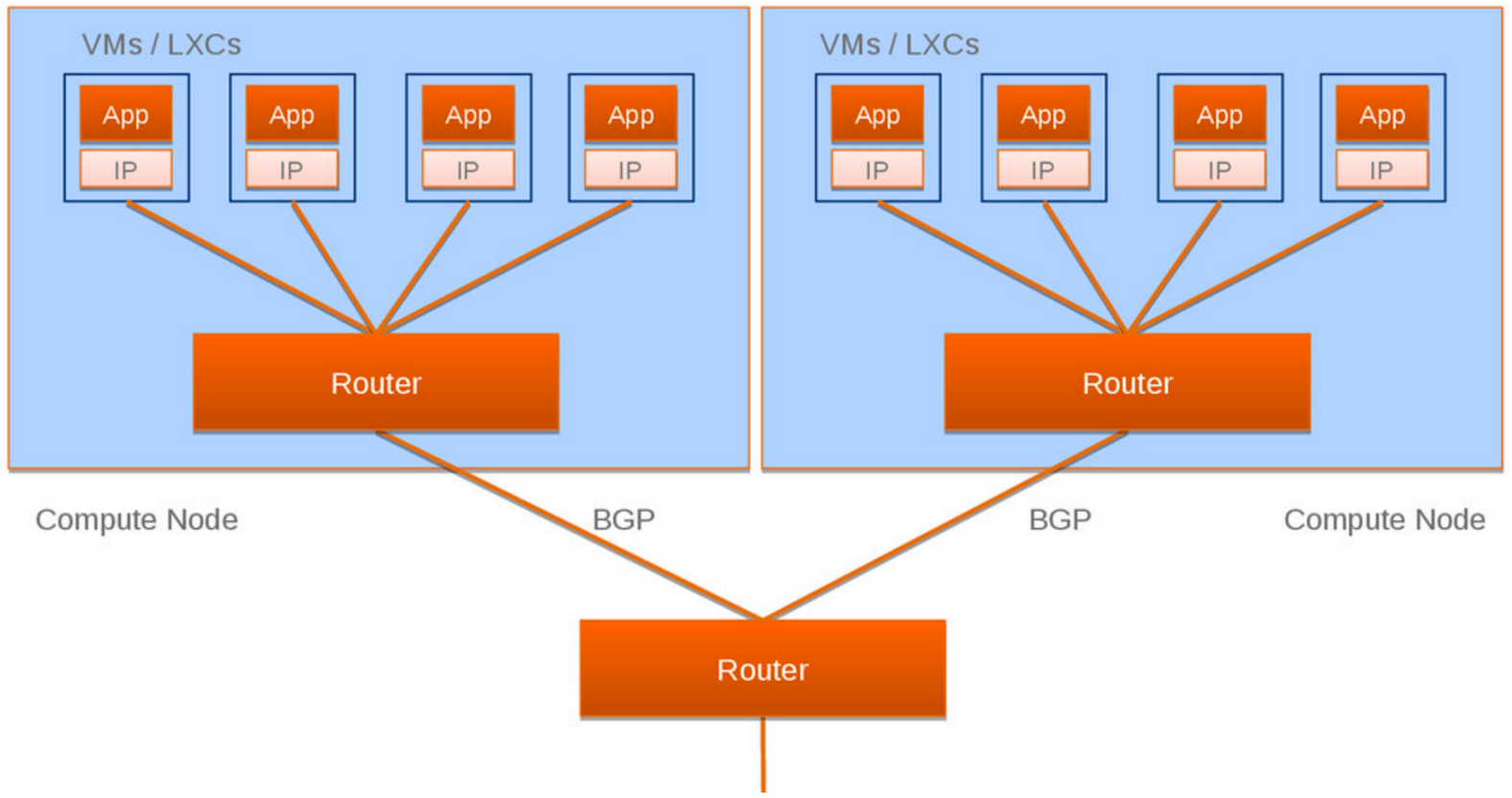
转发平面-udp


veth
docker0
ip routing
tun
device
read/write
flanneld
Host1
veth
docker0
ip routing
tun
device
read/write
flanneld
Host1
UDP overlay

转发平面-vxlan

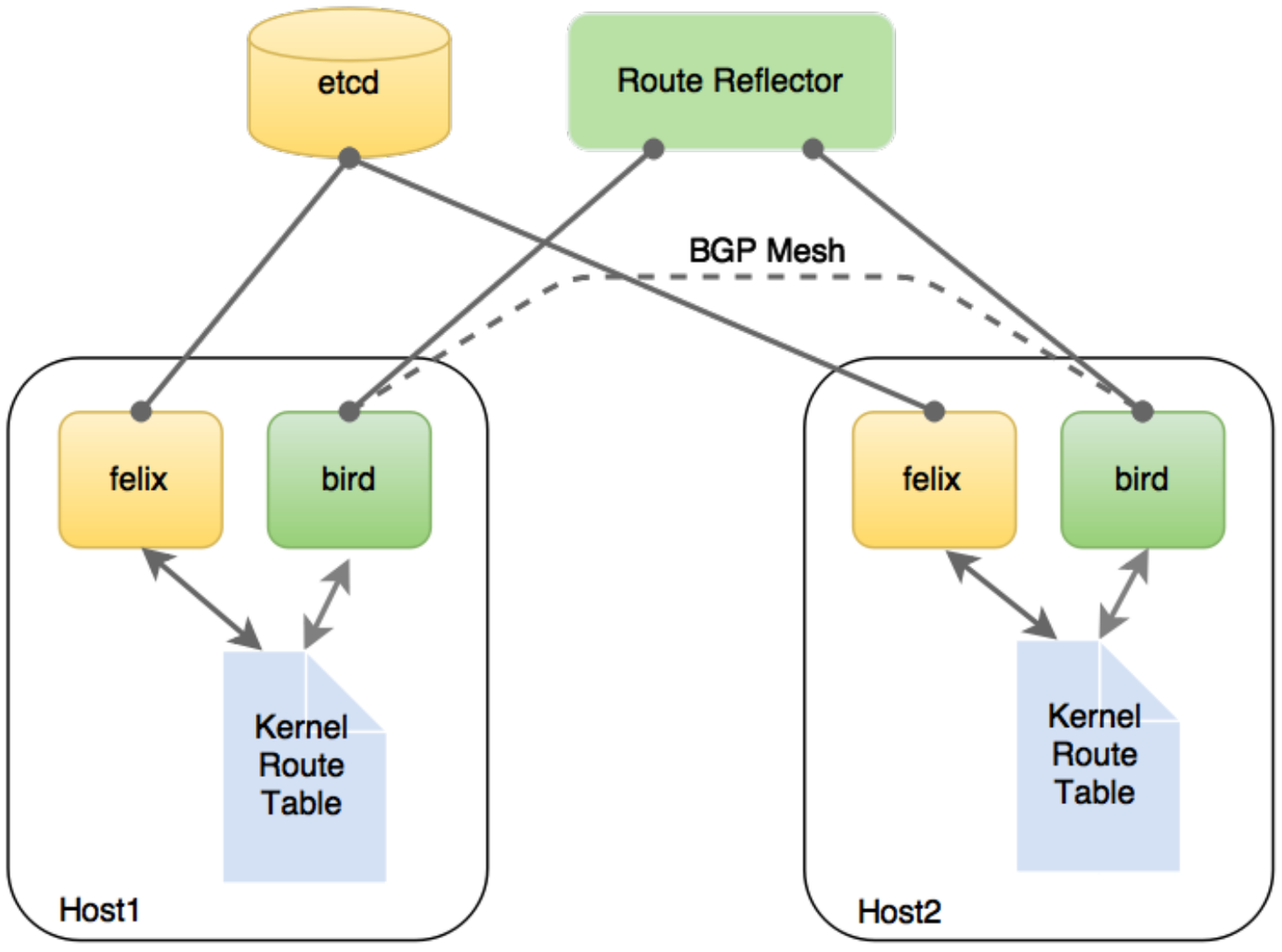

veth
docker0
ip routing
vxlan
device
flanneld
Host1
veth
docker0
ip routing
vxlan
device
flanneld
Host2
vxlan overlay

Calico

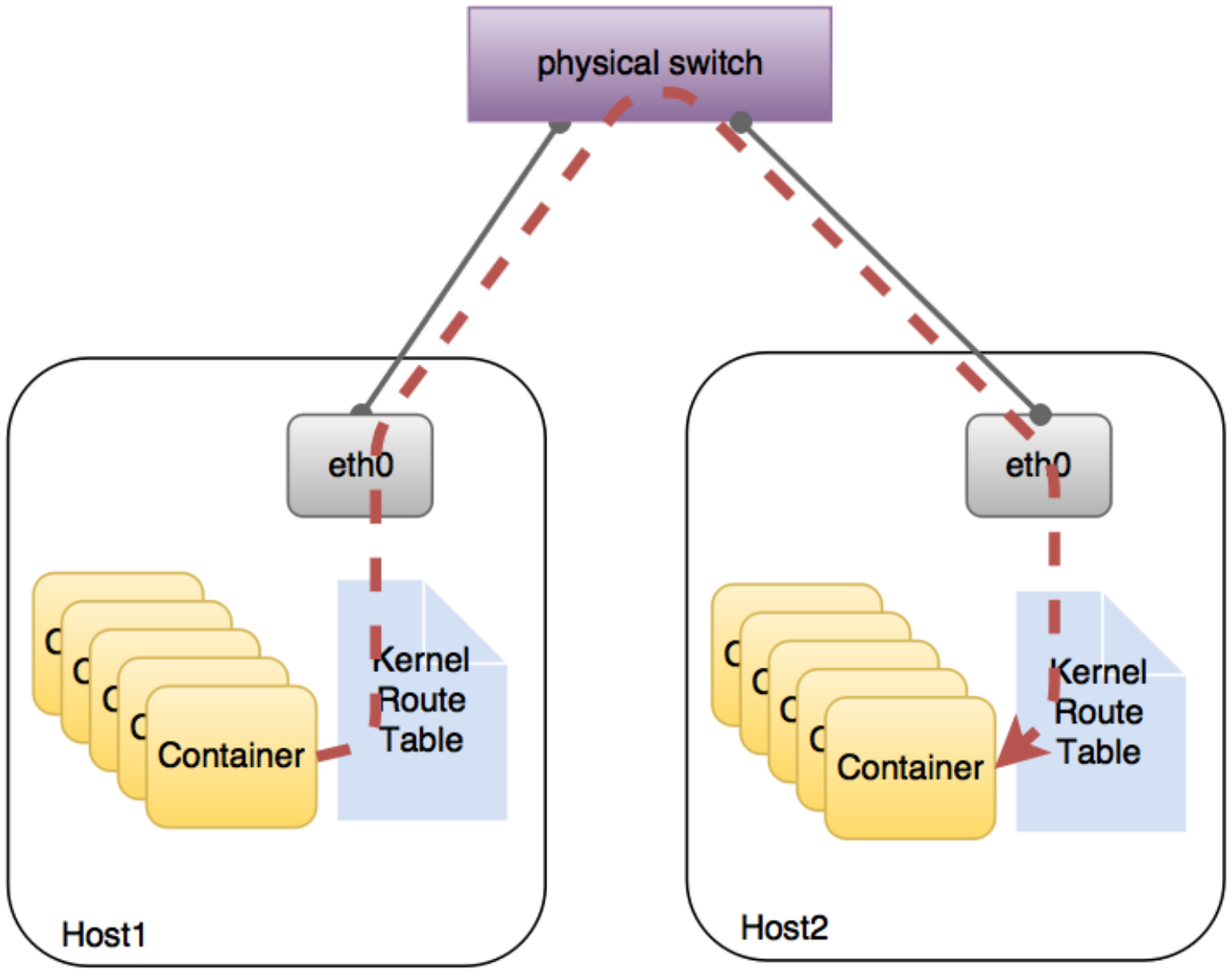


- 将操作系统协议栈化为 Router
- 模拟传统网络拓扑实现思路做路由转发

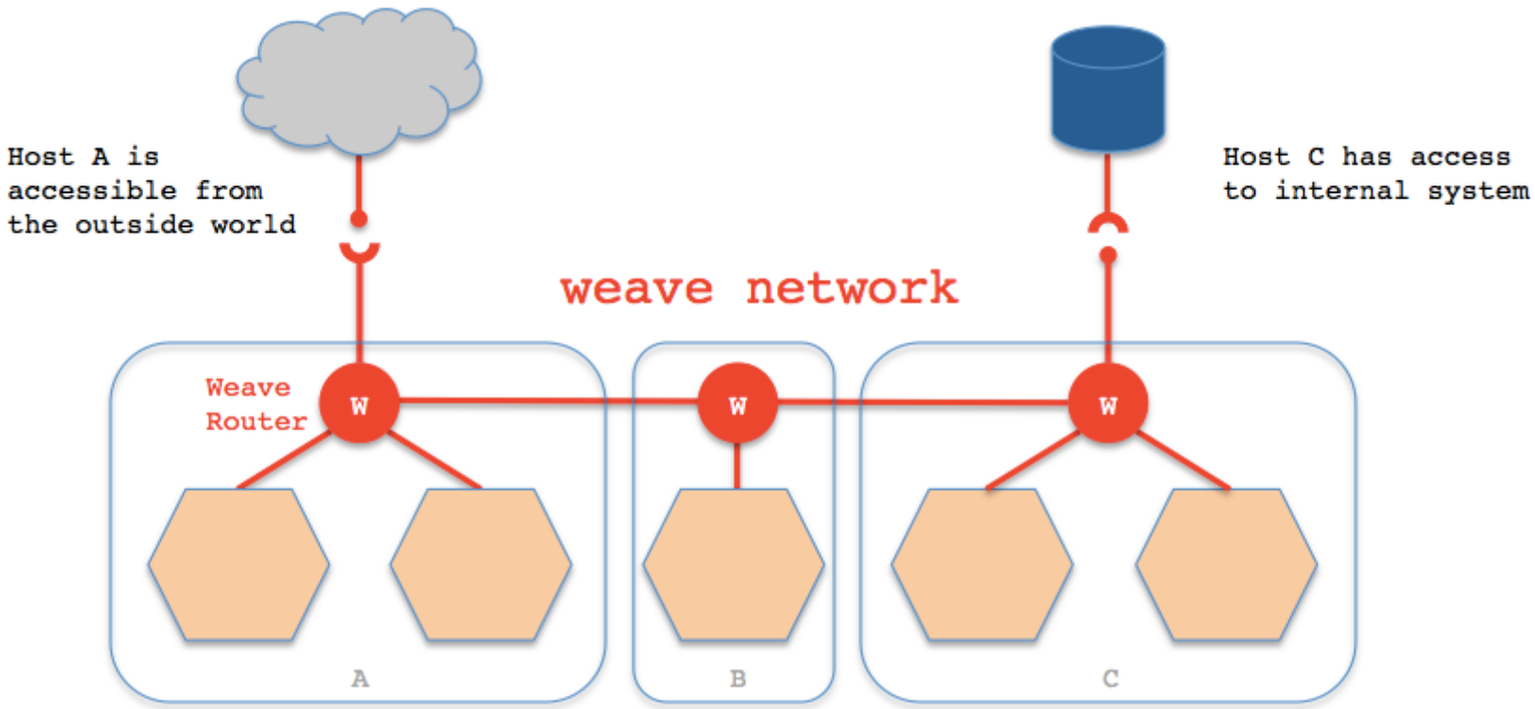
控制平面


etcd
Route Reflector
BGP Mesh
felix
bird
Kernel
Route
Table
Host1
felix
bird
Kernel
Route
Table
Host2

转发平面/隔离


physical switch
eth0
Container
Kernel
Route
Table
Host1
eth0
Container
Kernel
Route
Table
Host2

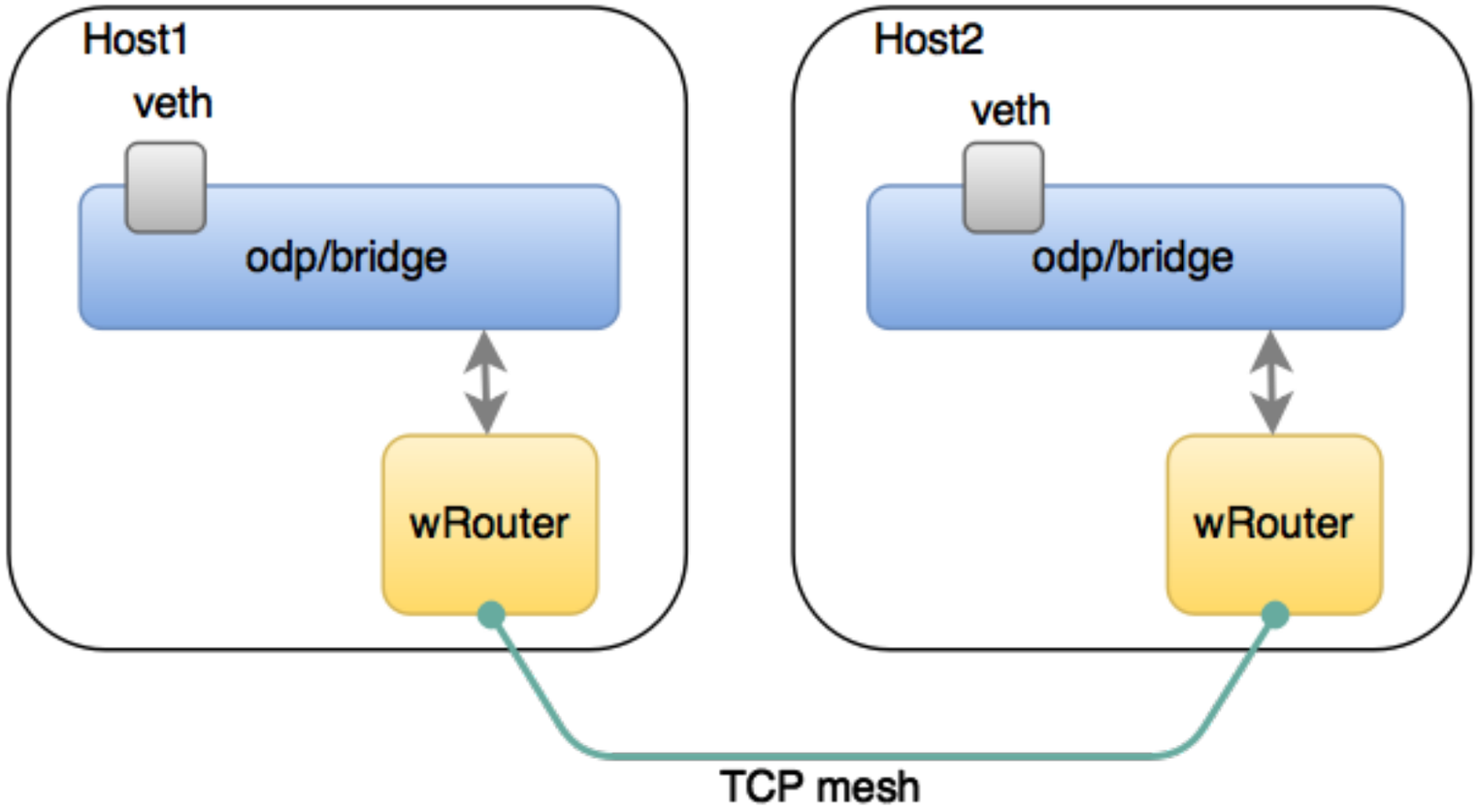
Weave



Hosts A, B & C running the containers shown in previous figure

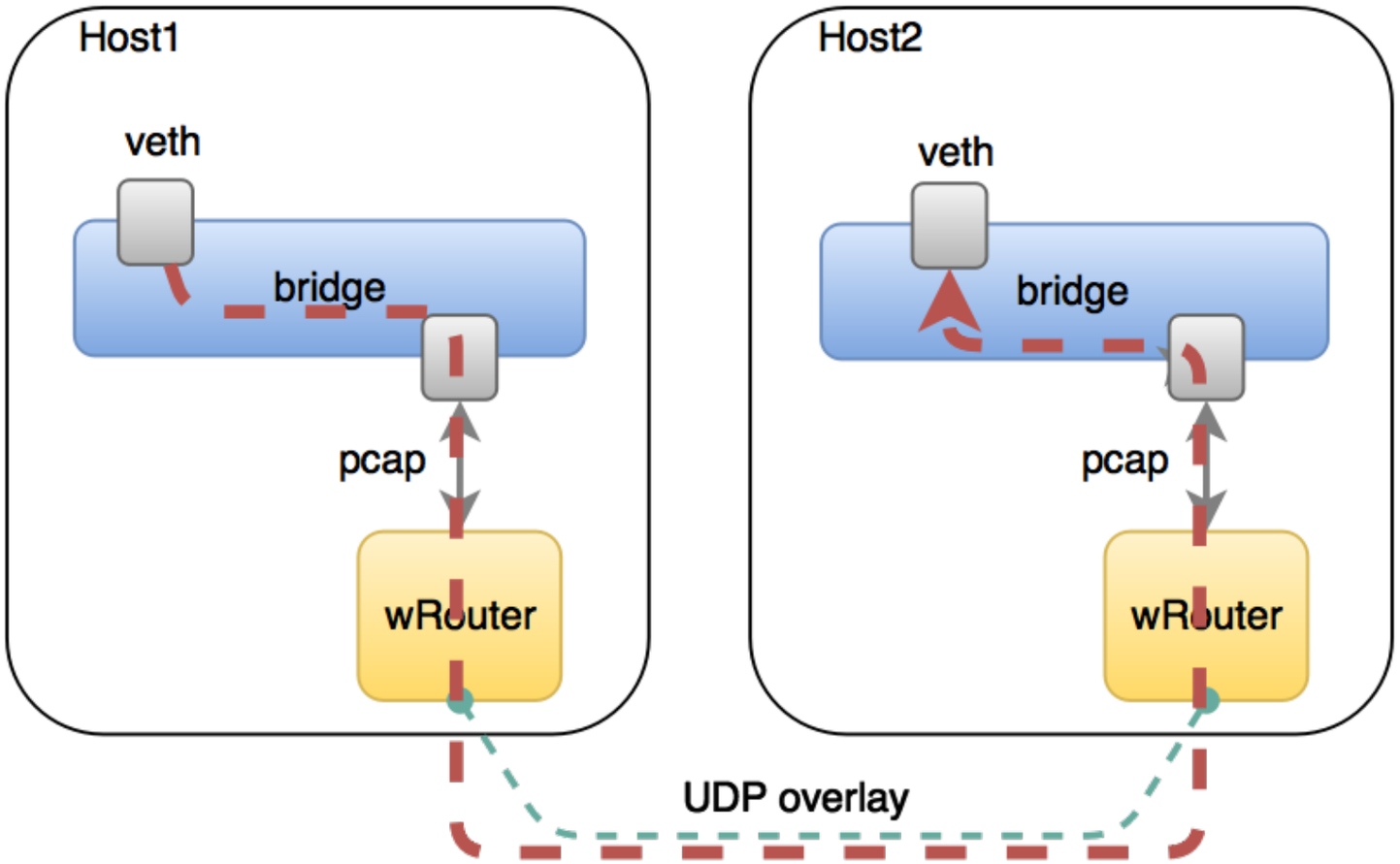
- 分布式 Router（container）
- Router 间建立 TCP 学习路由，形成控制平面
- Router 间建立 UDP/vxlan 通道，形成数据转发平面

控制平面



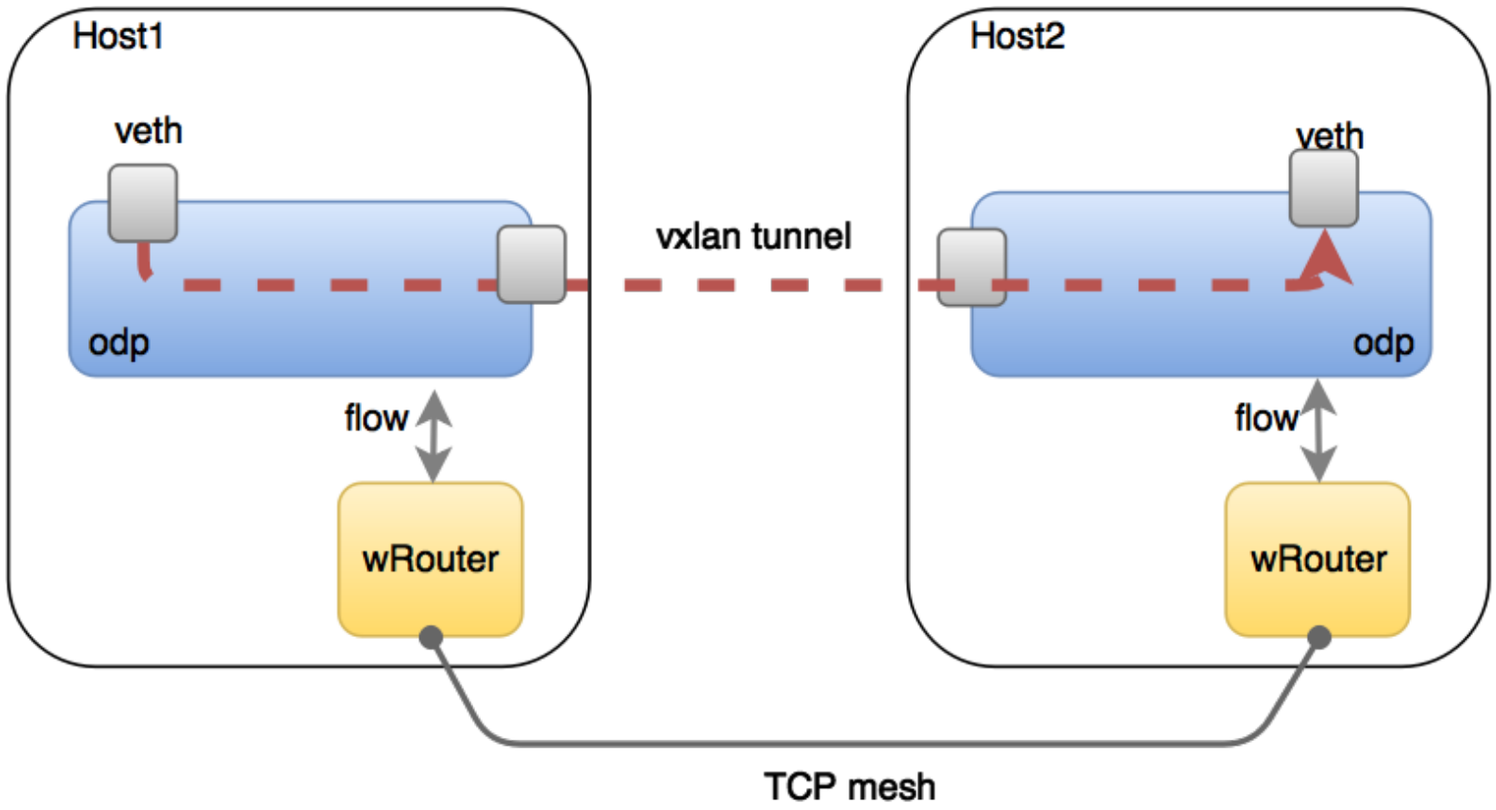
- 分布式 Router 之间 Full Mesh 结构
- 通过路由协议自主学习路由
- 自主进行节点拓扑计算与收敛

转发平面-pcap



- 从网桥设备上通过 pcap 获取原始帧
- wRouter 查询下一跳地址，将原始帧封成 UDP 发送
- 对端解封 UDP，将原始帧注入网桥设备

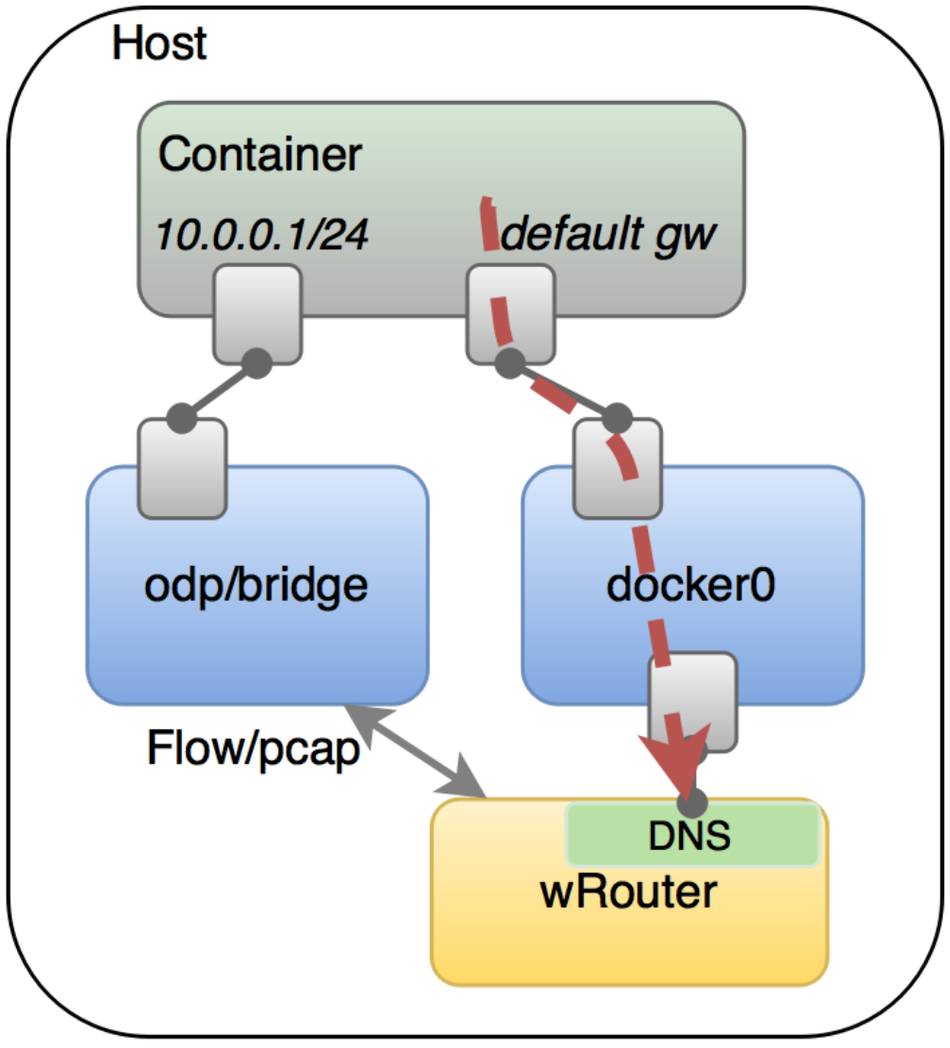
转发平面-odp



- wRouter 不参与实际流量转发
- wRouter 通过控制 odp Flow 规则达到路由目的
- odp 间通过 vxlan 隧道互联

服务发现/负载均衡

- DNS Server 监听在 docker0
- Container 的 dns server 指向 docker0
- DNS Server 从 wRouter 中查取



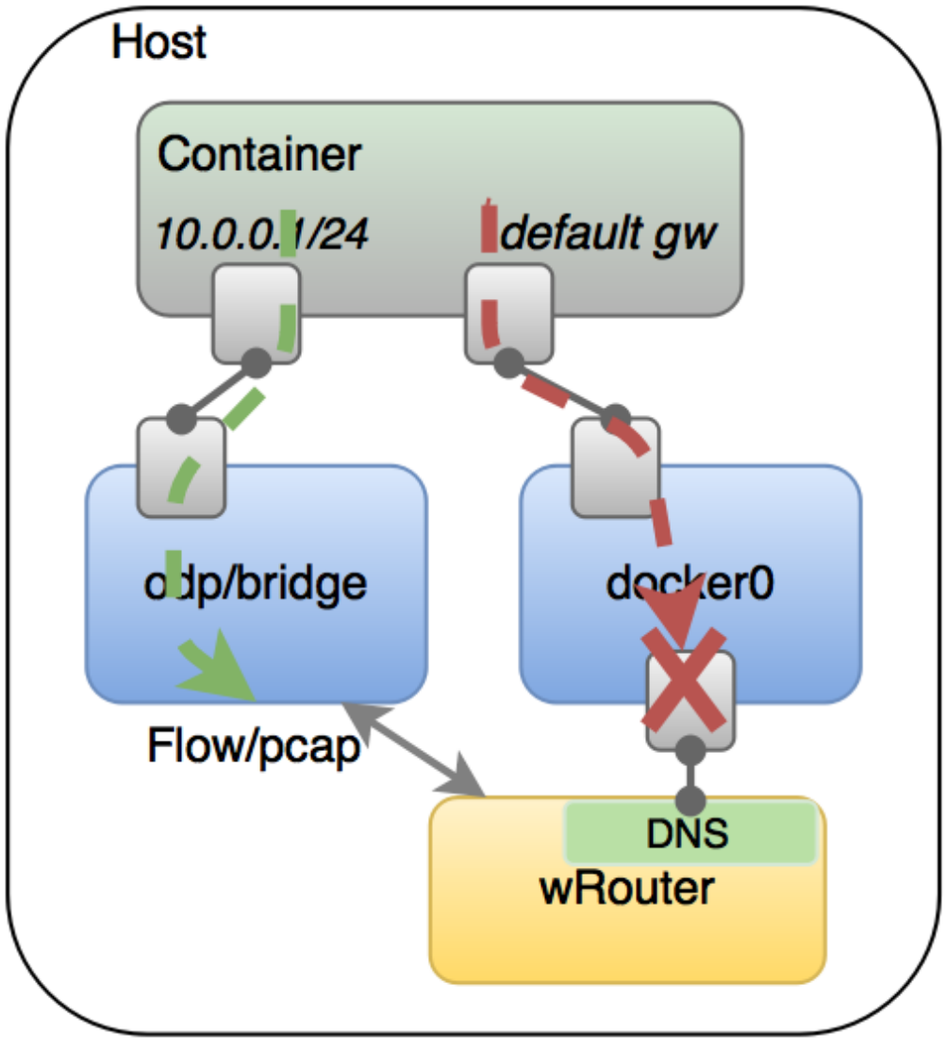
子网隔离

- 支持子网级隔离

  依赖容器的网络配置权限

- 仅子网内流量可走 weave bridge

- 非法流量会被导入默认网关（即 docker0）



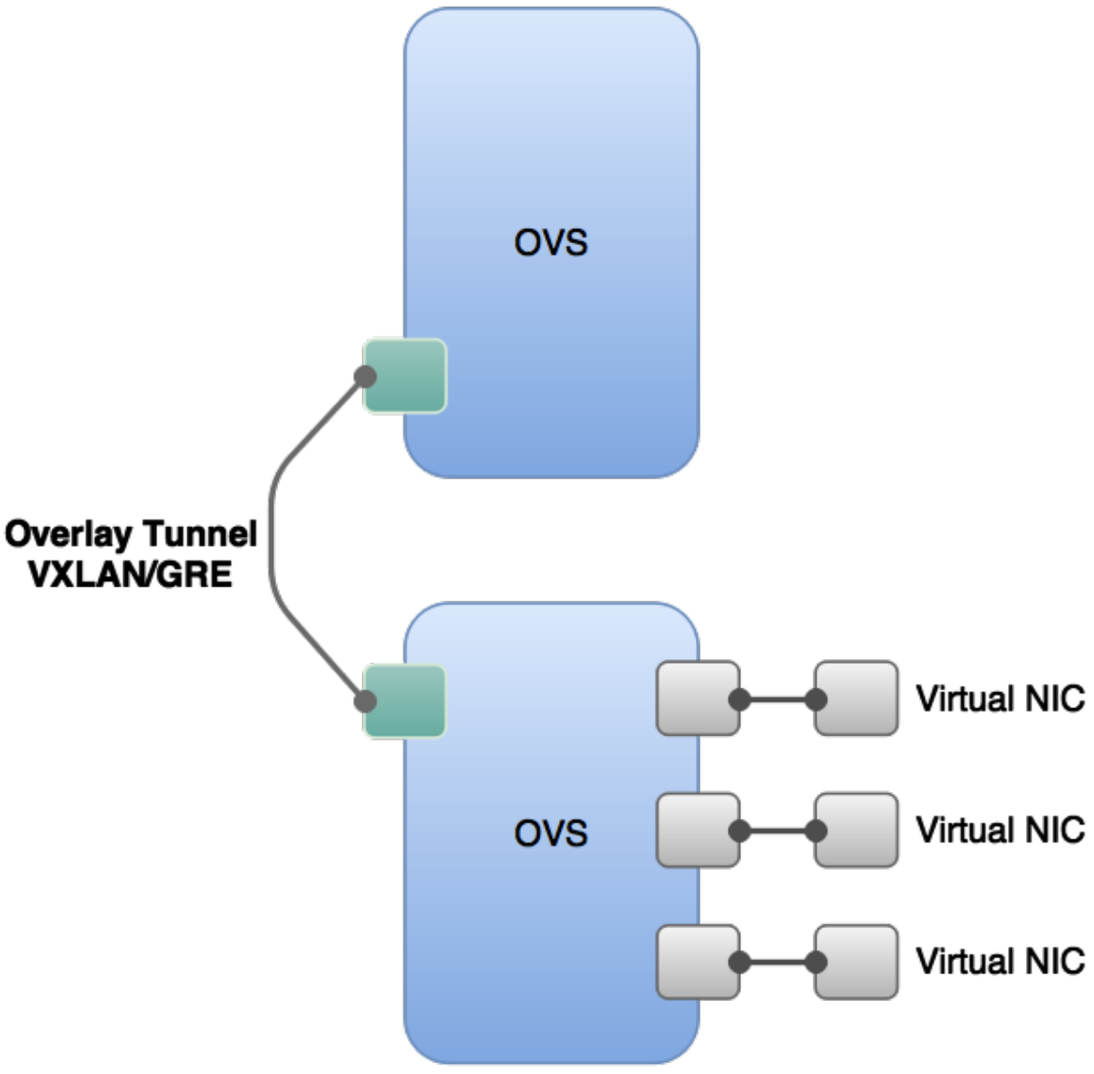
	Docker Bridge	Flannel	Weave	Calico
隔离粒度	粗	无	粗，子网级	细
地址管理	子网内分配	etcd	IPAM	IPAM
性能	NAT 损耗	udp 差 vxlan 好	pcap 差 odp 好	好
服务发现 负载均衡	无	无	DNS	无
控制平面	子网分配 二层学习	etcd	TCP mesh 路由协议	RR/BGP mesh
转发平面	Bridge + NAT	UDP/vxlan 隧道	UDP/vxlan 隧道	Linux 协议栈
对基础架构的要求	IP可达	IP可达	IP可达	二层互通 或BGP打通

业务需求

- 适应复杂异构的基础架构
- 端点可迁移，并保持 IP 不变
- 服务发现、负载均衡
  - 精细均衡（L4/L7）
  - 对业务无侵入性（无需端口配置）
- 端到端流量精细 ACL

转发平面

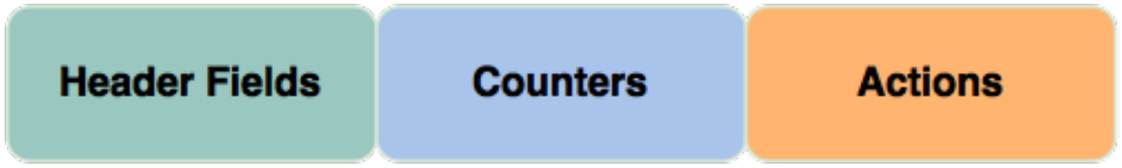
- 对原有物理基础网络侵入小
- offload 隧道计算量提升性能
- 硬件厂商支持相对完善



控制平面

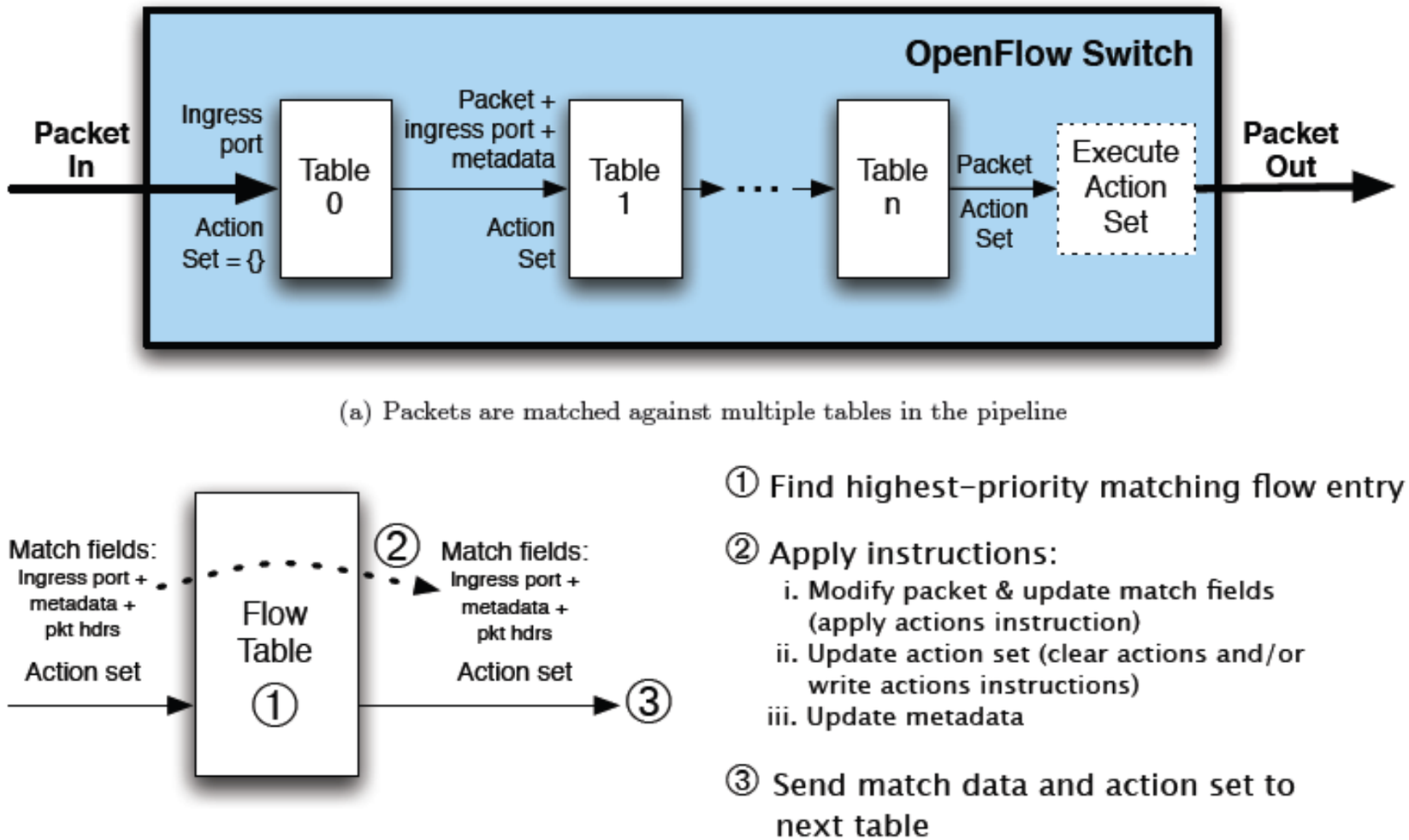
- 容器和虚机的一点不同
  - 容器间依赖关系相对固定（职责细化的结果）
  - 单机可推演可能产生的 outbound
  - 实际产生的 outbound << 可能产生的 outbound

OpenFlow



- 表达能力强
  - Header Fields 几乎可以匹配任意字段
  - 配合 Table 和 Priority 可实现复杂的处理逻辑
- 流表淘汰机制：
  - 被动：Idle/Hard Timeout
  - 主动：FlowDelete
- 支持第三方扩展，例如 Nicira

OpenFlow



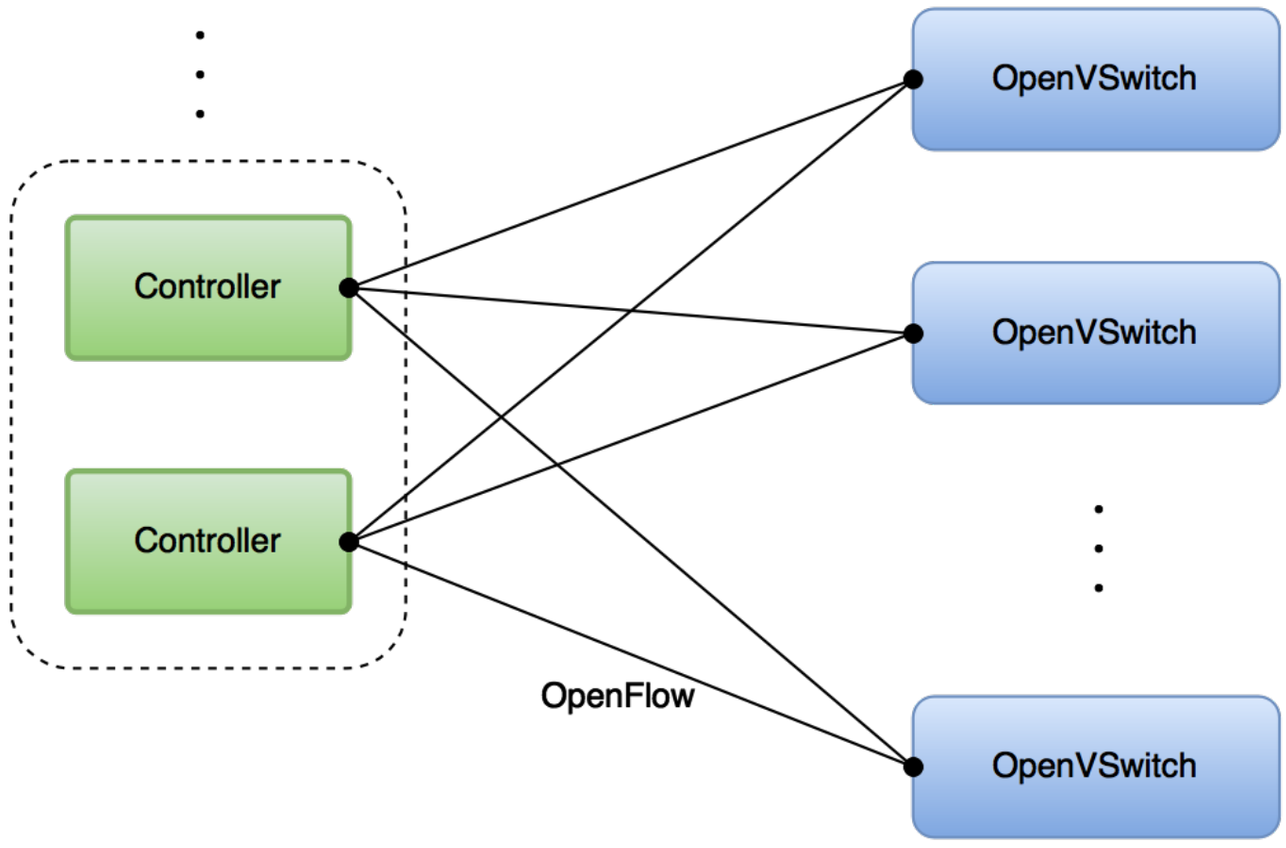
(a) Packets are matched against multiple tables in the pipeline

(b) Per-table packet processing

Figure 2: Packet flow through the processing pipeline.

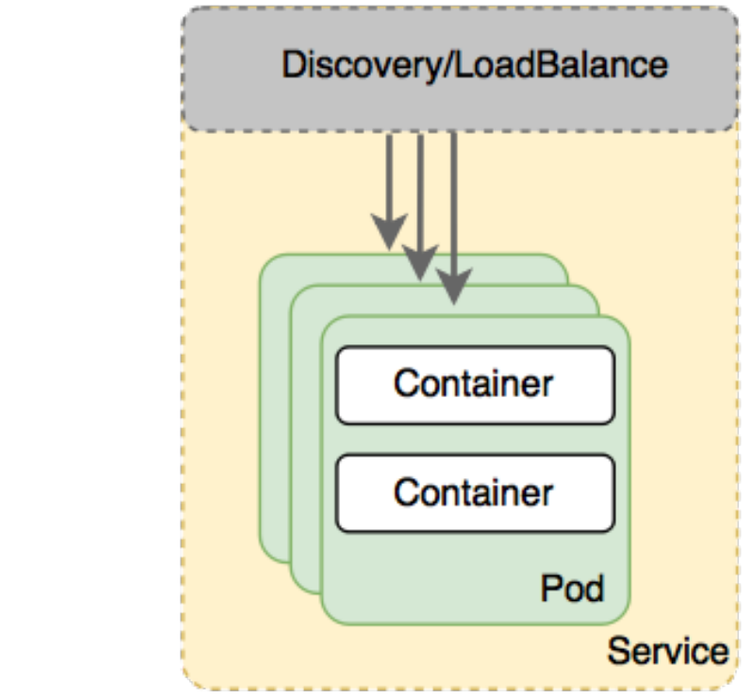
控制平面

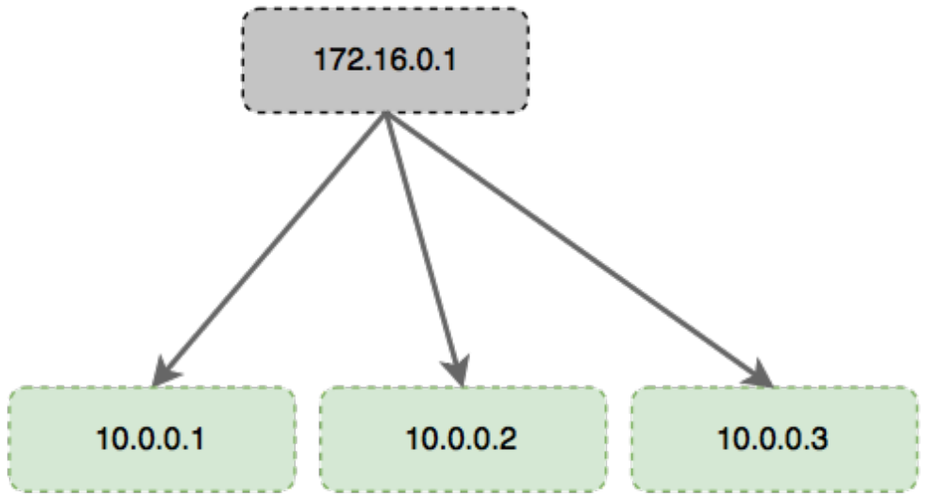
- 多控制器实例一组
  - 负载均衡
  - 高可用
- 多组分区域管辖



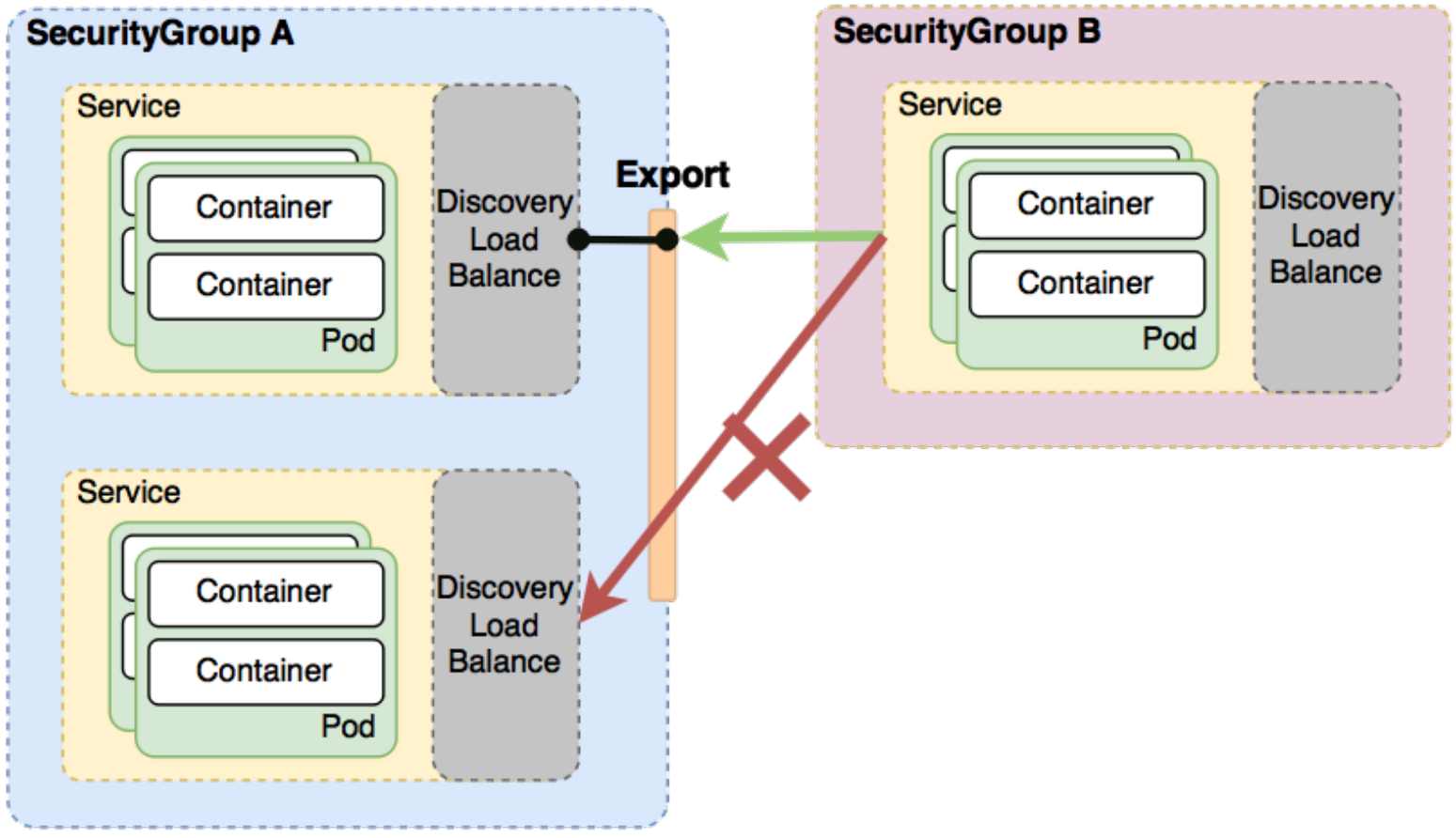
服务发现/负载均衡

- 分布式负载均衡器
  - 在本机即被代理，无额外内网流量开销
- IP 级负载均衡
  - 业务无需配置端口（后端监听即用）
- L7 自定义配置丰富
  - 支持常用 Nginx 配置
  - 可指定访问某个后端
  - etc.





安全组

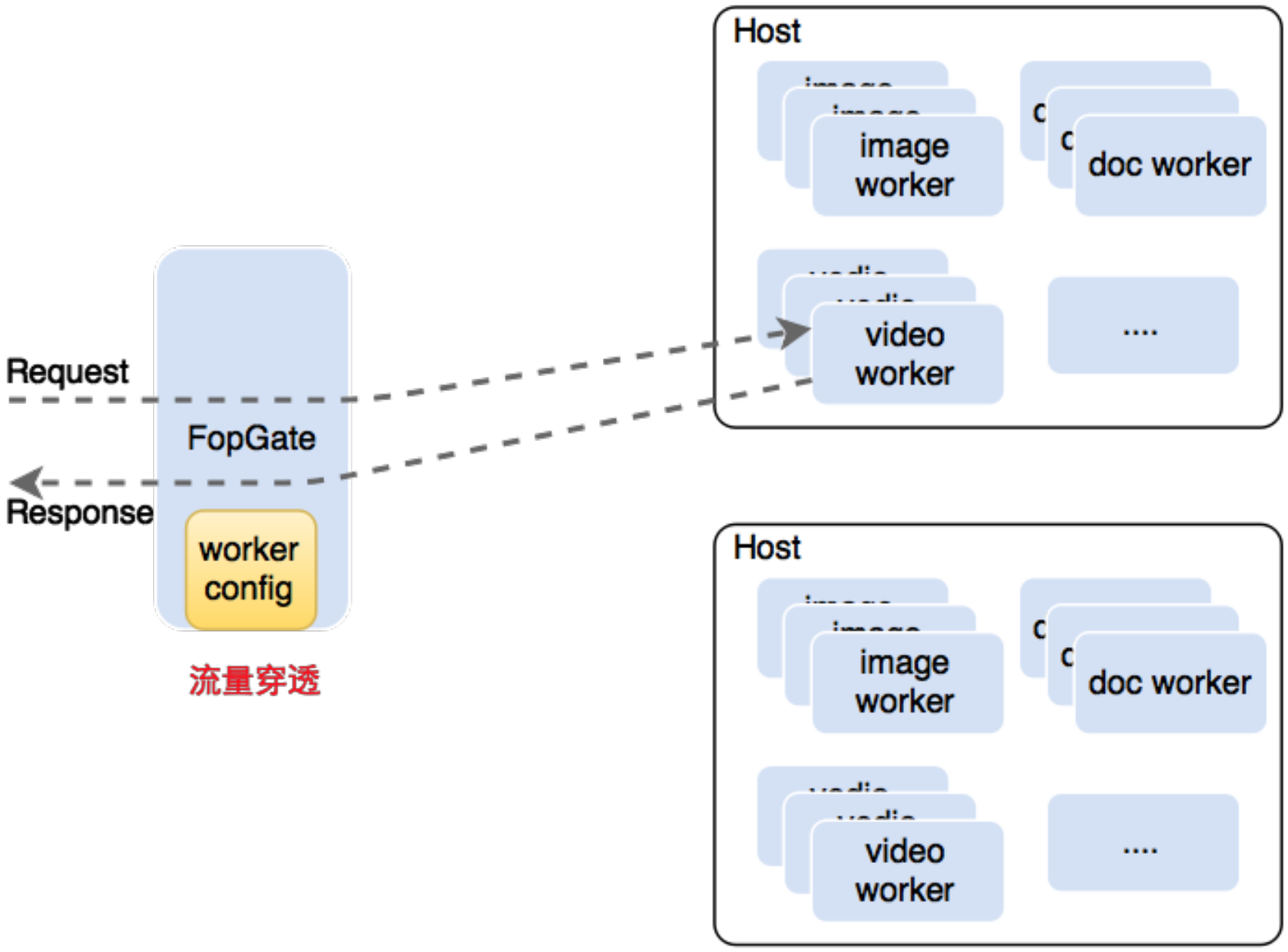


- 支持 Pod/Service/SecurityGroup <-> Pod/Service/SecurityGroup
- 可向目标安全组 Export 指定 Pod/Service
- 支持 ICMP/TCP/UDP

案例一

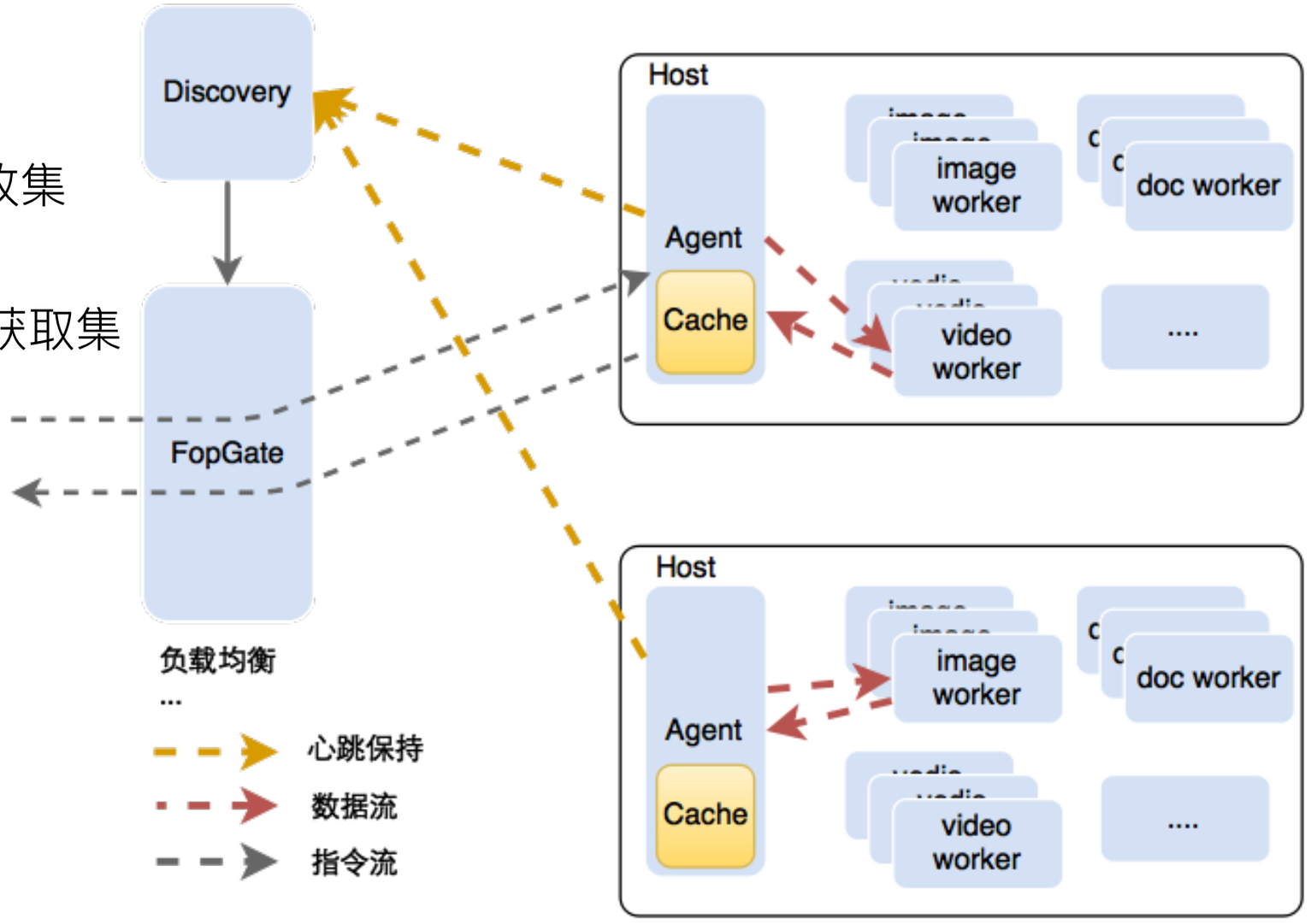
七牛文件处理FOP架构演变

文件处理架构Stage1



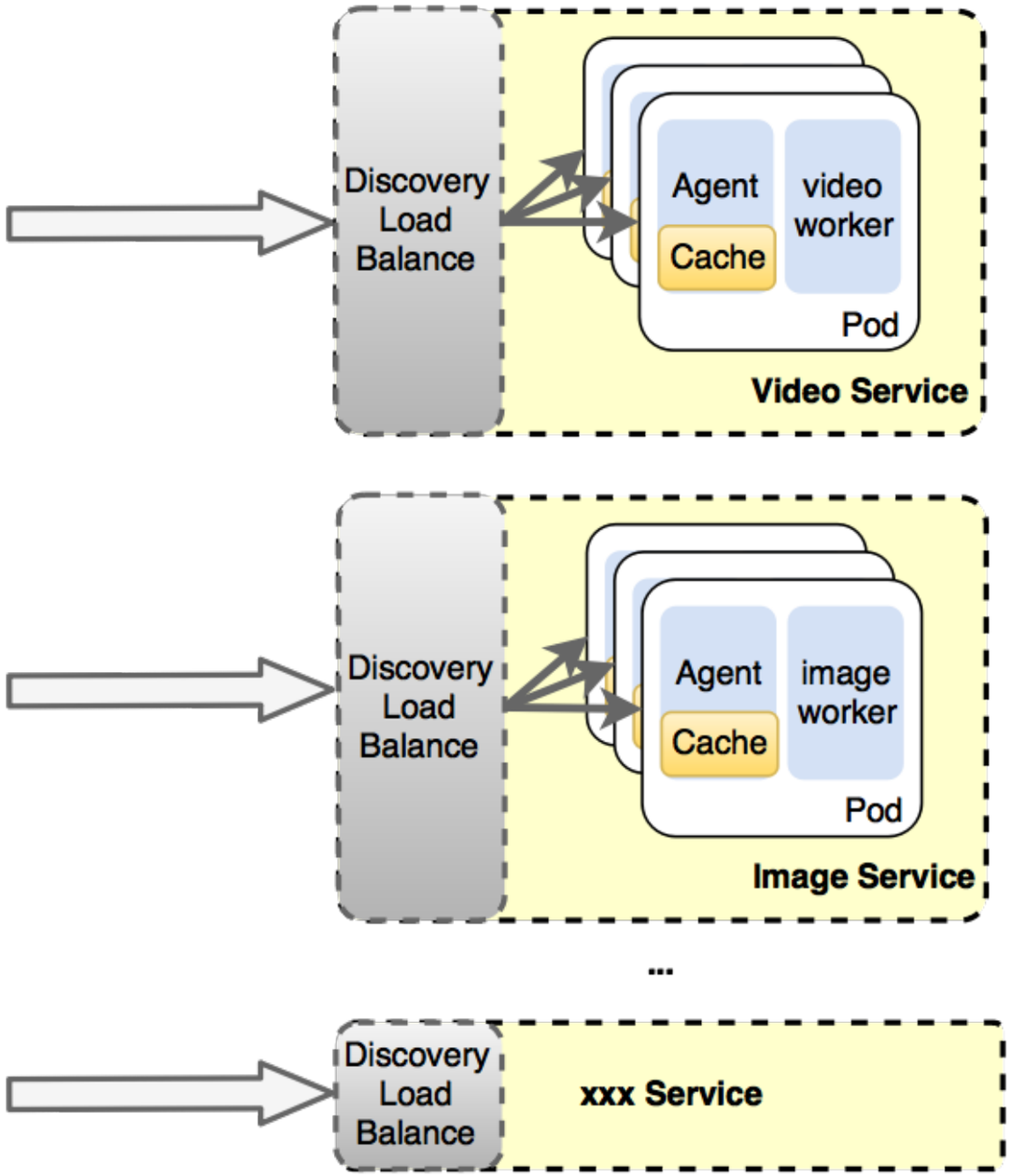
文件处理架构Stage2

- 增加 Discovery 组件，收集 Agent 上报信息
- FopGate 从 Discovery 获取集群信息，做 LB
- 增加业务 Agent
  - 上报后端信息
  - 上报保活信息
  - 单机内 worker LB



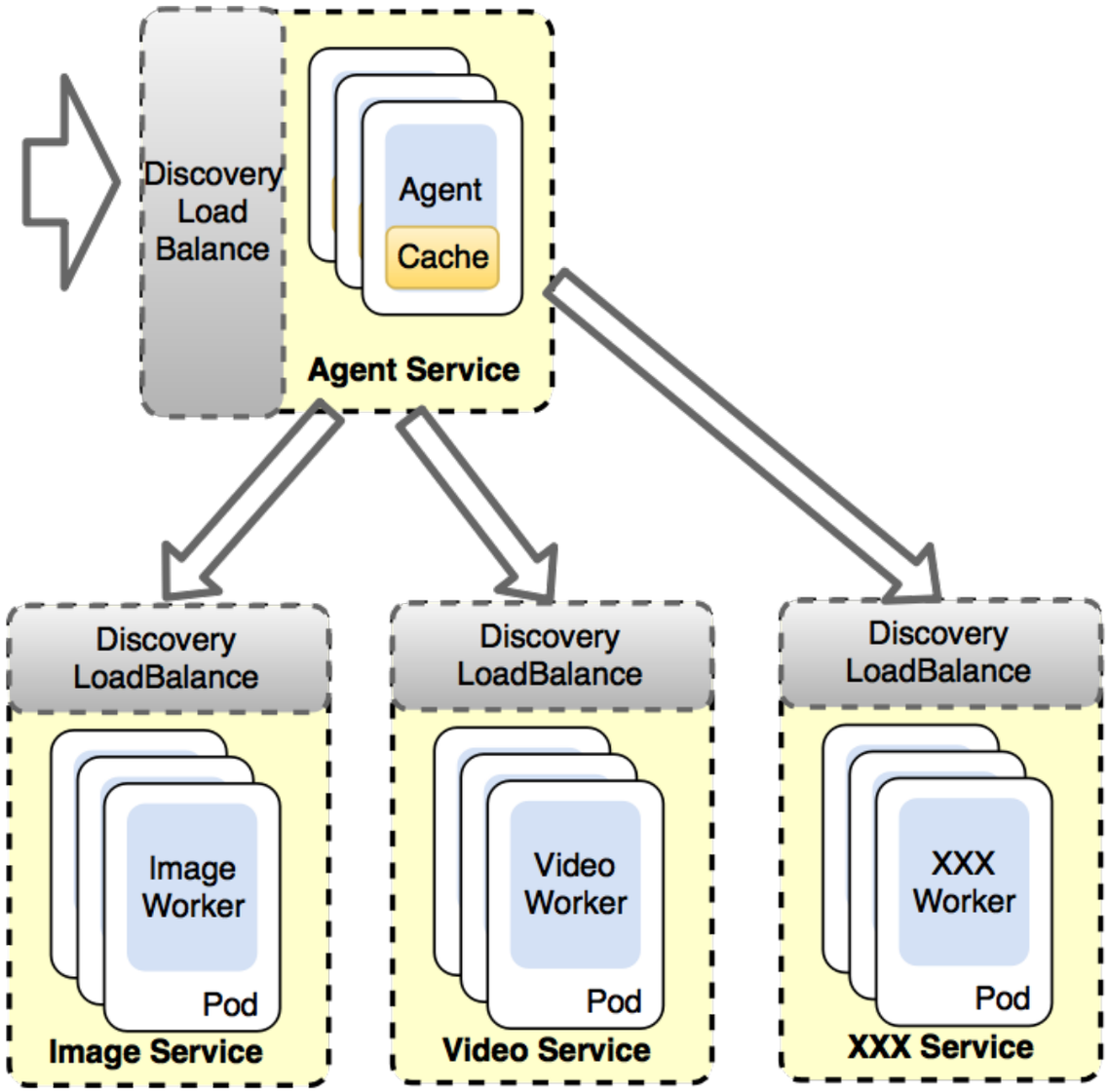
文件处理架构Stage3

- 取消业务层 Discovery 服务
- 业务 Agent 功能退化
  - 无需与 Discovery 心跳保持
  - 简化 Agent 后端，无需 LB
- FopGate 取消 LB 逻辑



文件处理架构Stage4

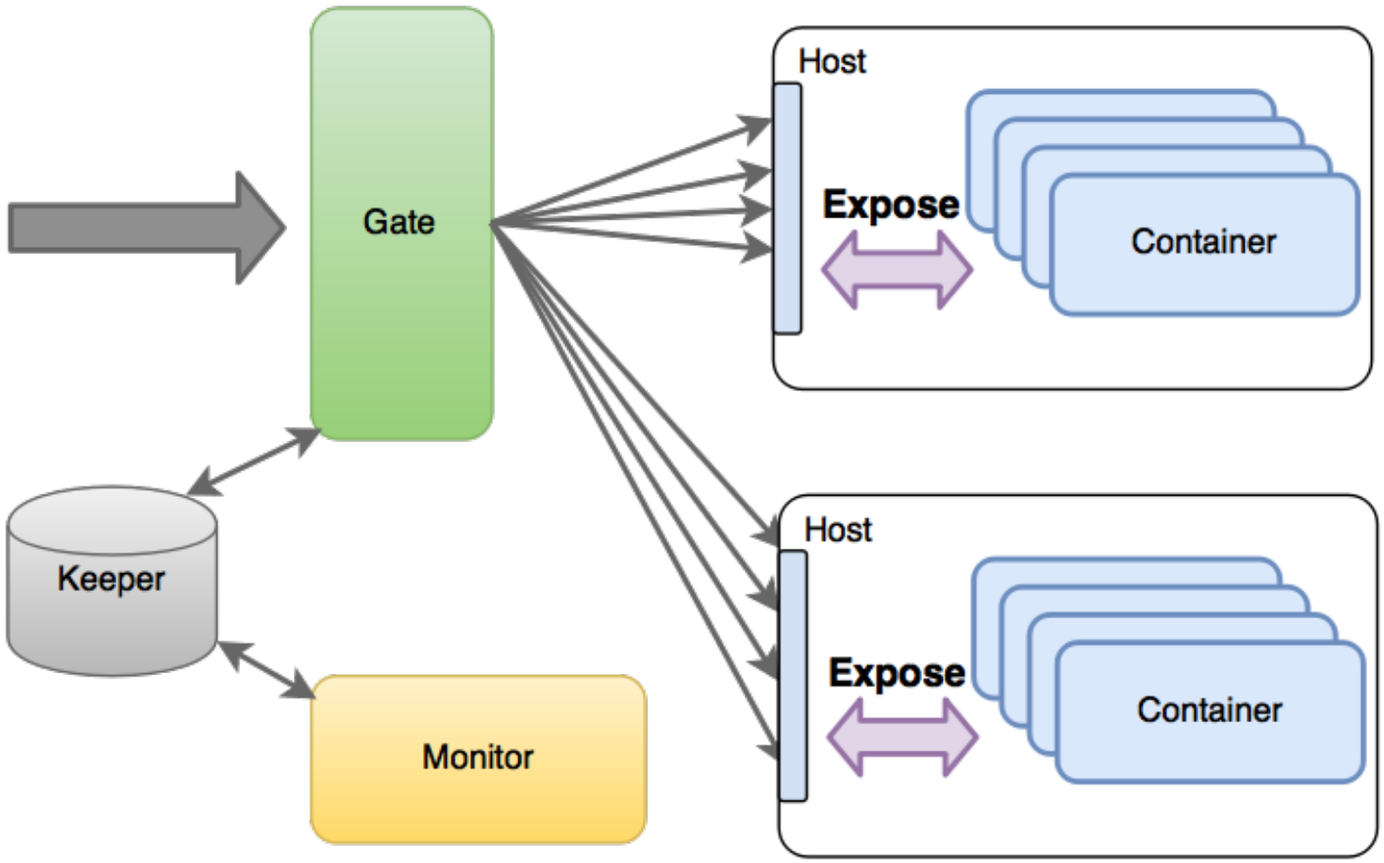
- 进一步分离
  - 有状态服务 Agent
  - 无状态服务 worker
- 通过调度调配 worker 数量



案例二

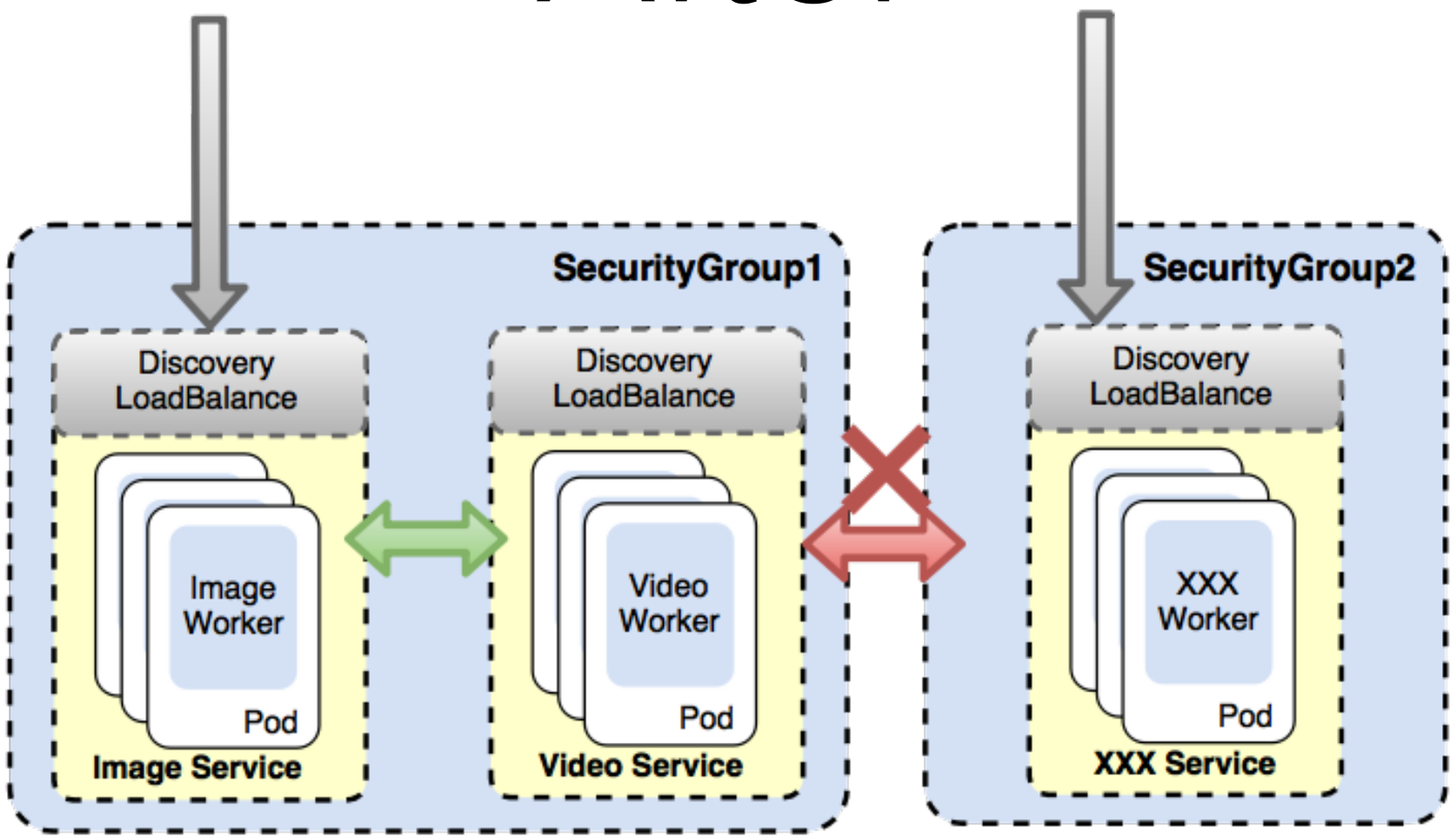
用户自定义文件处理UFOP架构演变

Before



- 端口映射，管理麻烦，业务使用不灵活
- 隔离粗暴，容器间通讯一律禁止

After



- 天然 Discovery/LoadBalance
- 划分安全组，容器间关系灵活
- 完整端口空间，业务使用方便

Q & A