大规模场景下Kubernetes Service负载均衡性能优化

杜军

华为Cloud BU - PaaS开源组
Github: @m1093782566

Geekbang.
极客邦科技

极客搜索

全站干货，一键触达，只为技术

s.geekbang.org

扫描二维码立即体验

有没有一种搜索方式，能整合 InfoQ 中文站、极客邦科技旗下12大微信公众号矩阵的全部资源?
极客搜索，这款针对极客邦科技全站内容资源的轻量级搜索引擎，做到了!

扫描上方二维码，极客搜索!

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

这里只有
技术领导者

EGO会员第二季招募季正式开启

报名时间：9月1日-9月15日

扫描添加E小欧，

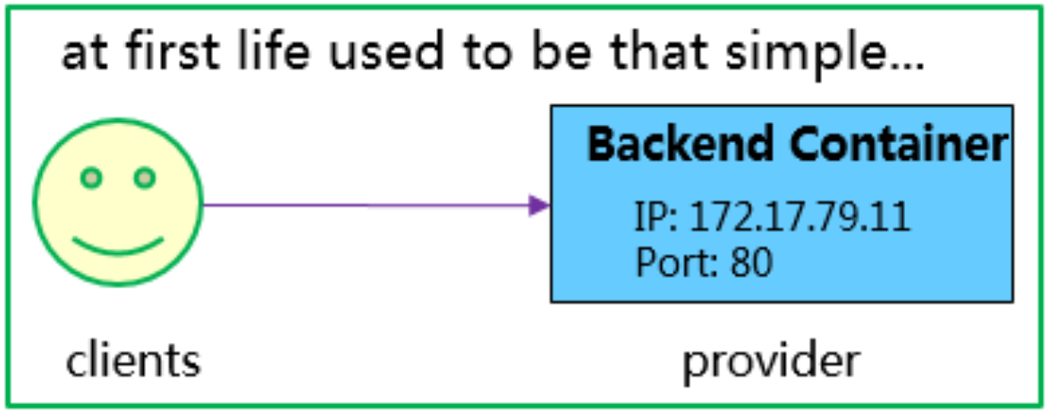
邀您进入EGO会员预报名群

立即报名

E小欧

TABLE OF **CONTENTS**

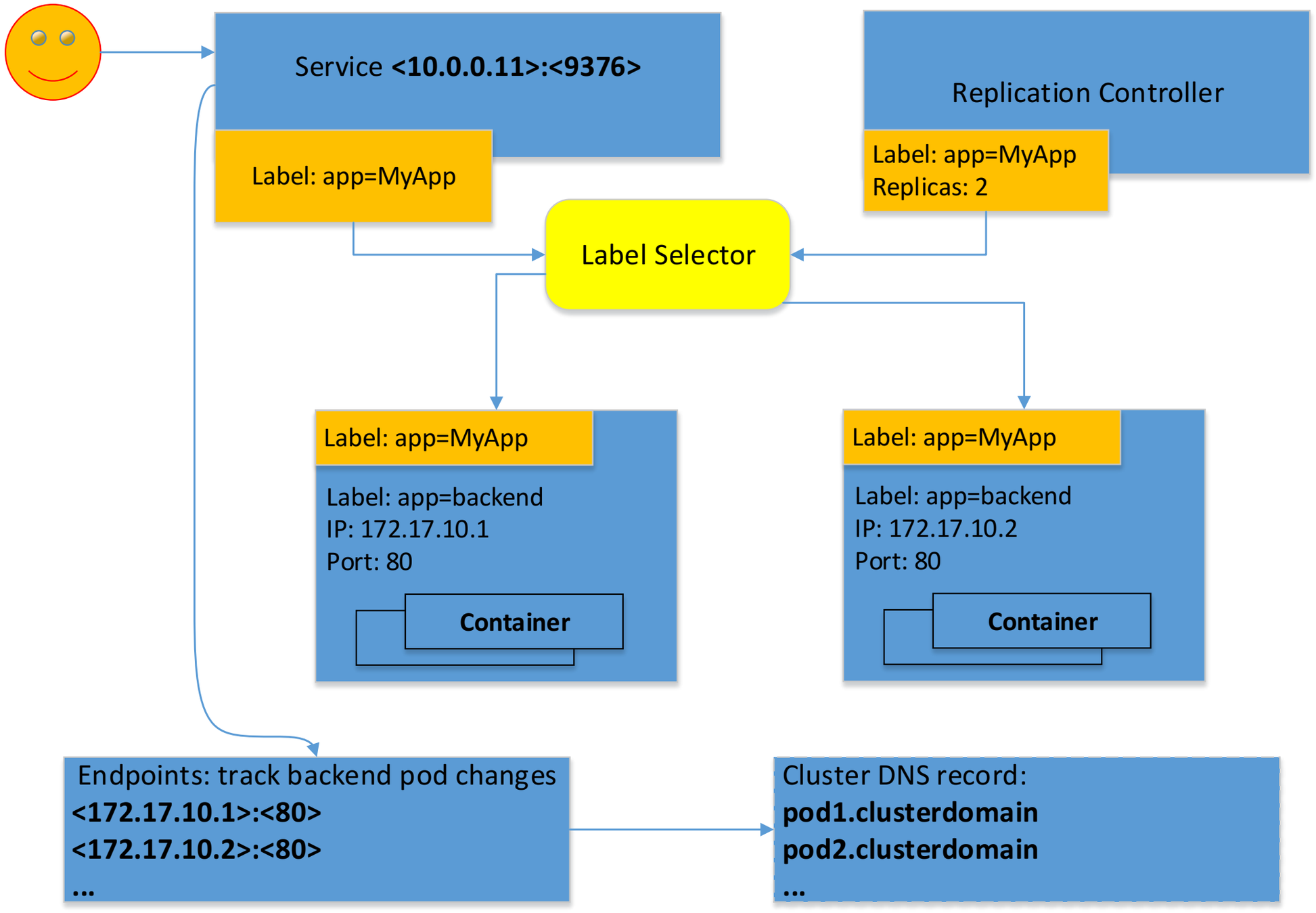
Kubernetes的Service



但，简单的生活总是暂时的：

- 多个后端实例，如何做到负载均衡？
- 如何保持会话亲和性？
- 容器迁移，IP发生变化如何访问？
- 健康检查怎么做？
- 怎么通过域名访问？

Kubernetes Service与Endpoints



Service与Endpoints定义

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: default
spec:
  clusterIP: 10.101.28.148
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
```

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: nginx-service
  namespace: default
subsets:
- addresses:
  - ip: 172.17.0.2
    nodeName: 100-106-179-237.node
    targetRef:
      kind: Pod
      name: nginx-rc-c8tw2
      namespace: default
  - ip: 172.17.0.3
    nodeName: 100-106-179-238.node
    targetRef:
      kind: Pod
      name: nginx-rc-x14tv
      namespace: default
  ports:
  - name: http
    port: 8080
    protocol: TCP
```

Service 内部逻辑

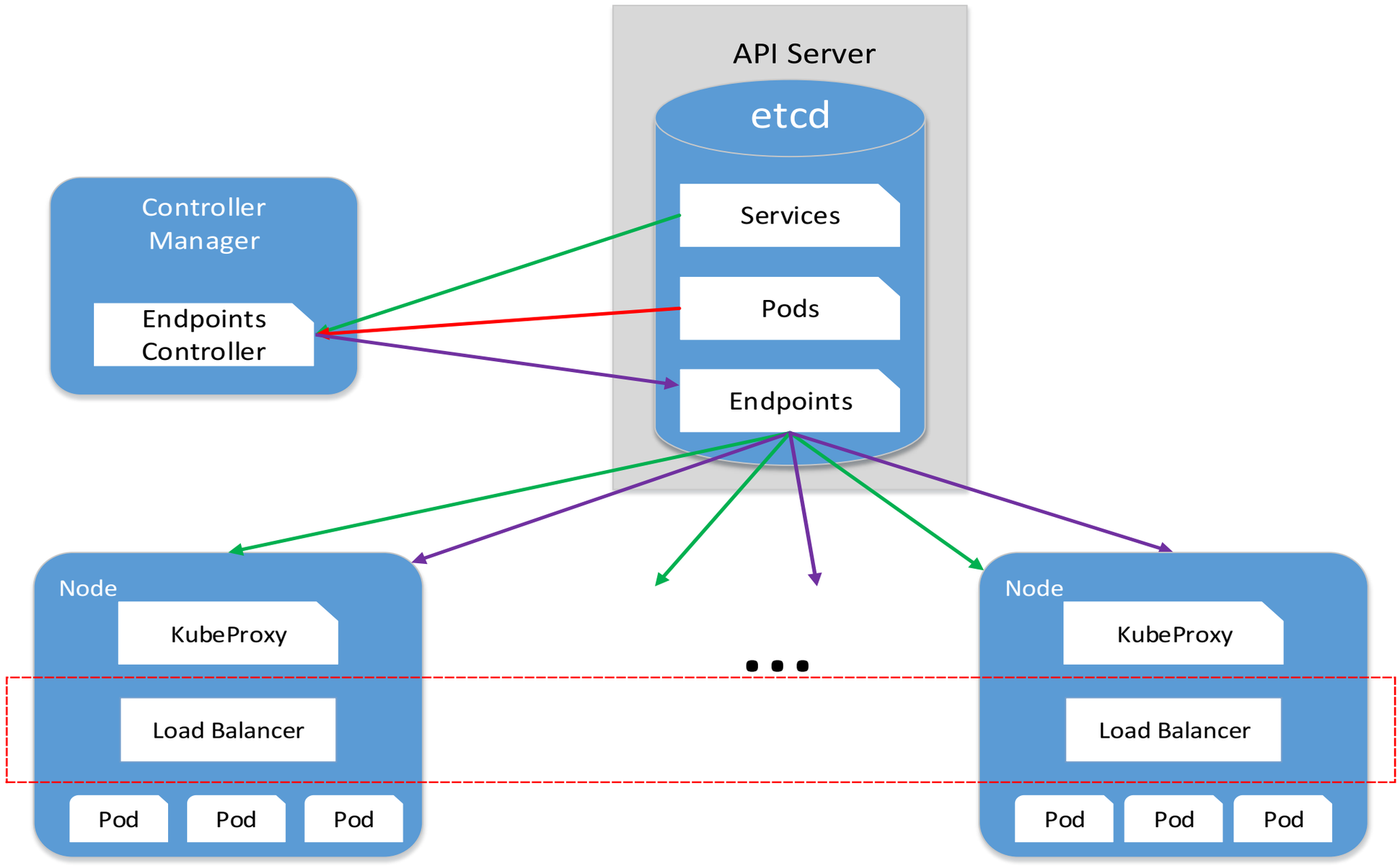
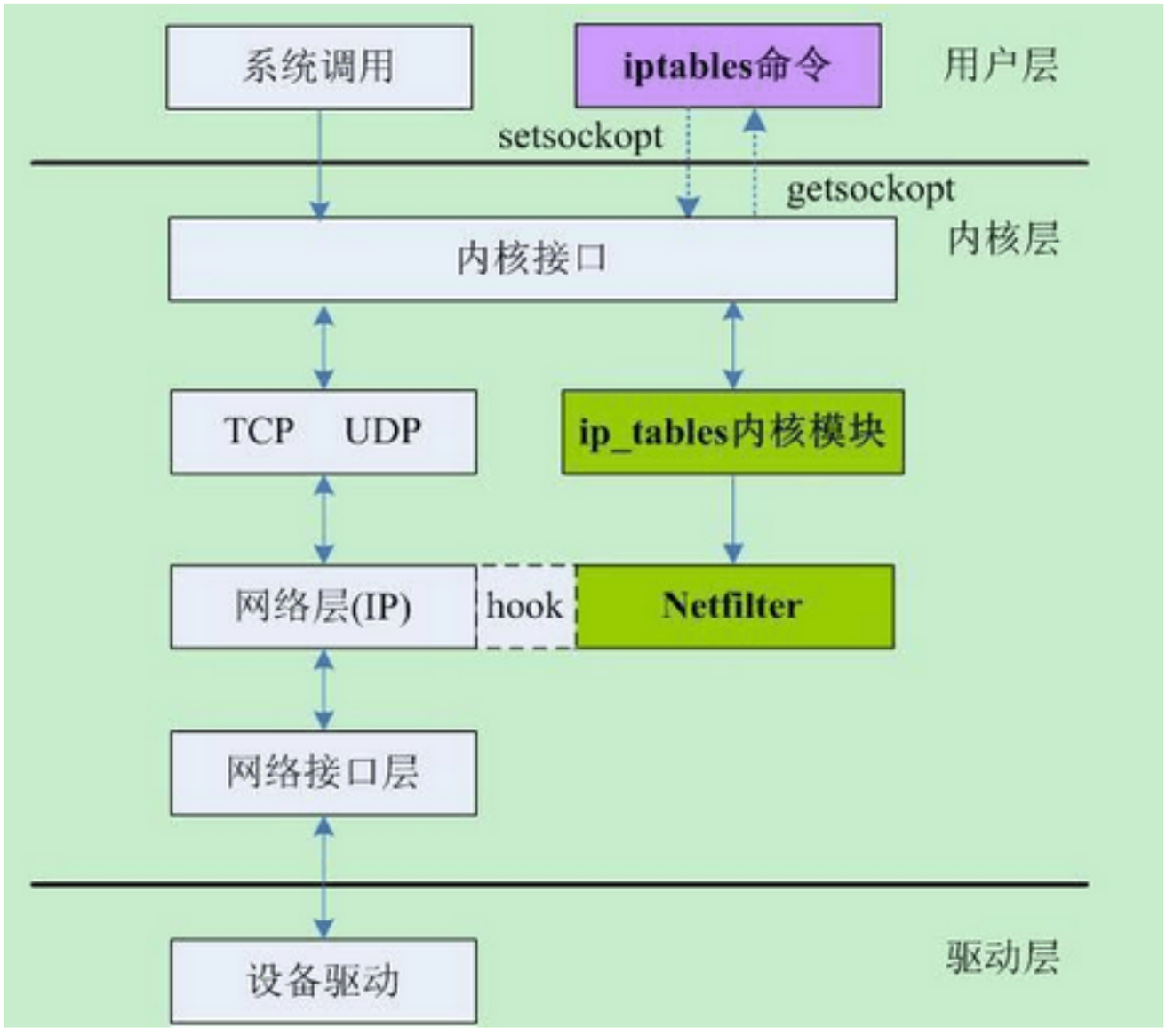


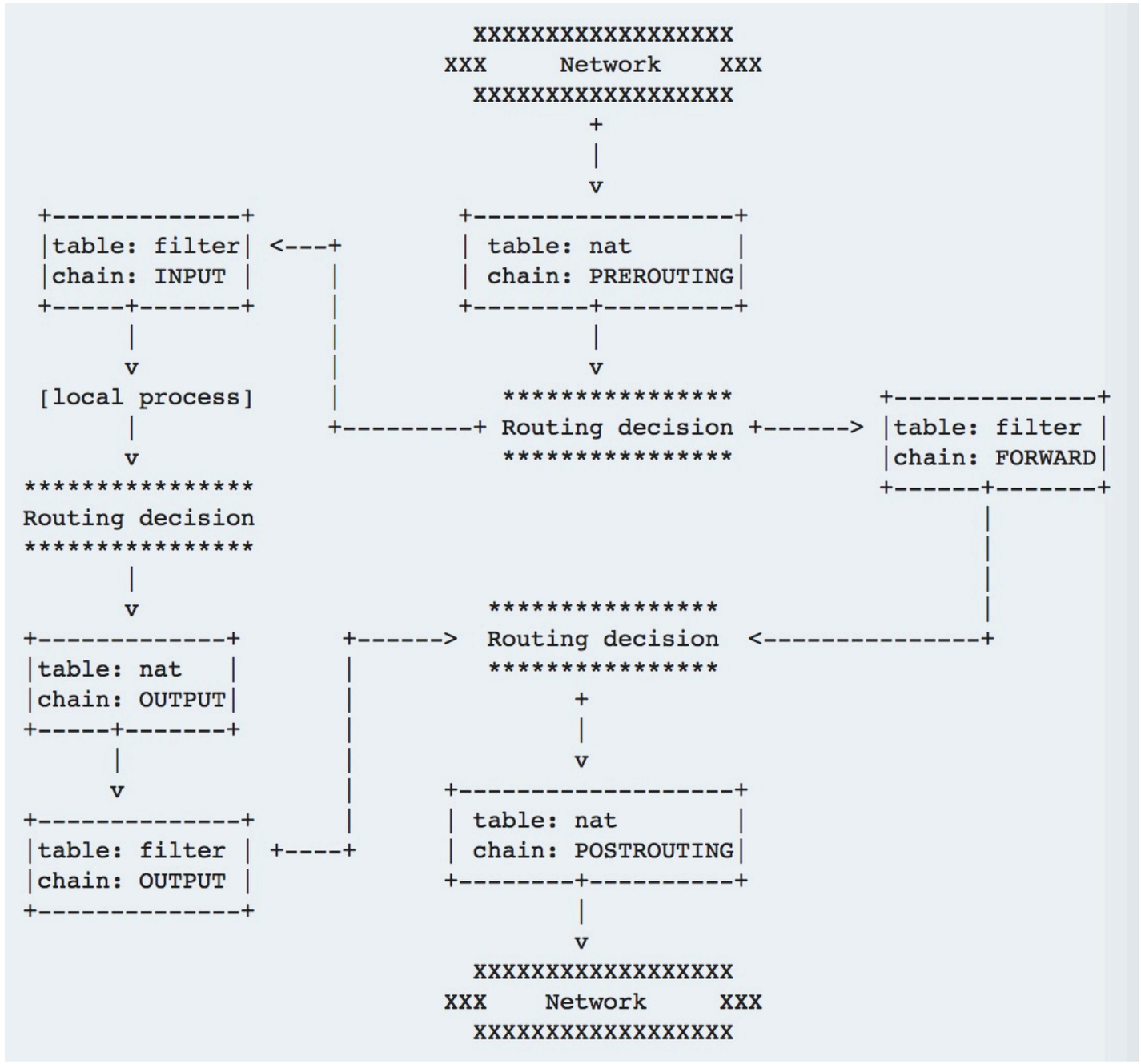
TABLE OF **CONTENTS**

Iptables是什么？

- 用户空间应用程序，通过配置Netfilter规则表（ Xtables ）来构建linux内核防火墙。



网络包通过Iptables全过程



Iptables实现流量转发与负载均衡

- **Iptables如何做流量转发？**
  - **DNAT实现IP地址和端口映射**

iptables -t nat -A PREROUTING -d 1.2.3.4/32 --dport 80 **-j DNAT --to-destination** 10.20.30.40:8080

- **Iptables如何做负载均衡？**
  - **statistic模块为每个后端设置权重**

iptables -t nat -A PREROUTING -d 1.2.3.4 --dport 80 **-m statistic --mode random --probability .25** -j DNAT --to-destination 10.20.30.40:8080

- **Iptables如何做会话保持？**
  - **recent模块设置会话保持时间**

iptables -t nat –A FOO -m recent --rcheck --seconds 3600 --reap --name BAR -j BAR

Iptables在Kubernetes中的应用举例

VIP:Port -> PREROUTING(OUTPUT) -> KUBE-SERVICES -> KUBE-SVC-XXX -> KUBE-SEP-XXX -> RIP:Port

```
Chain PREROUTING (policy ACCEPT)                                    1
target     prot opt source               destination
KUBE-SERVICES  all  --  0.0.0.0/0            0.0.0.0/0

Chain KUBE-SERVICES (2 references)                                  2
target     prot opt source                     destination
KUBE-SVC-6IM33IEVEEV7U3GP  tcp  --  0.0.0.0/0      10.20.30.40  tcp dpt:80

Chain KUBE-SVC-6IM33IEVEEV7U3GP (1 references)                      3
target     prot opt source                     destination
KUBE-SEP-Q3UCPZ54E6Q2R4UT  all  --  0.0.0.0/0            0.0.0.0/0

Chain KUBE-SEP-Q3UCPZ54E6Q2R4UT (1 references)                      4
target     prot opt source                     destination
DNAT       tcp  --  0.0.0.0/0                0.0.0.0/0           tcp to:172.17.0.2:8080
```

TABLE OF **CONTENTS**

Iptables做负载均衡的问题

- **规则线性匹配时延**

  KUBE-SERVICES链挂了一长串KUBE-SVC-*链；访问每个service，要遍历每条链直到匹配，时间复杂度**O(N)**

- **规则更新时延**

  非增量式

- **可扩展性**

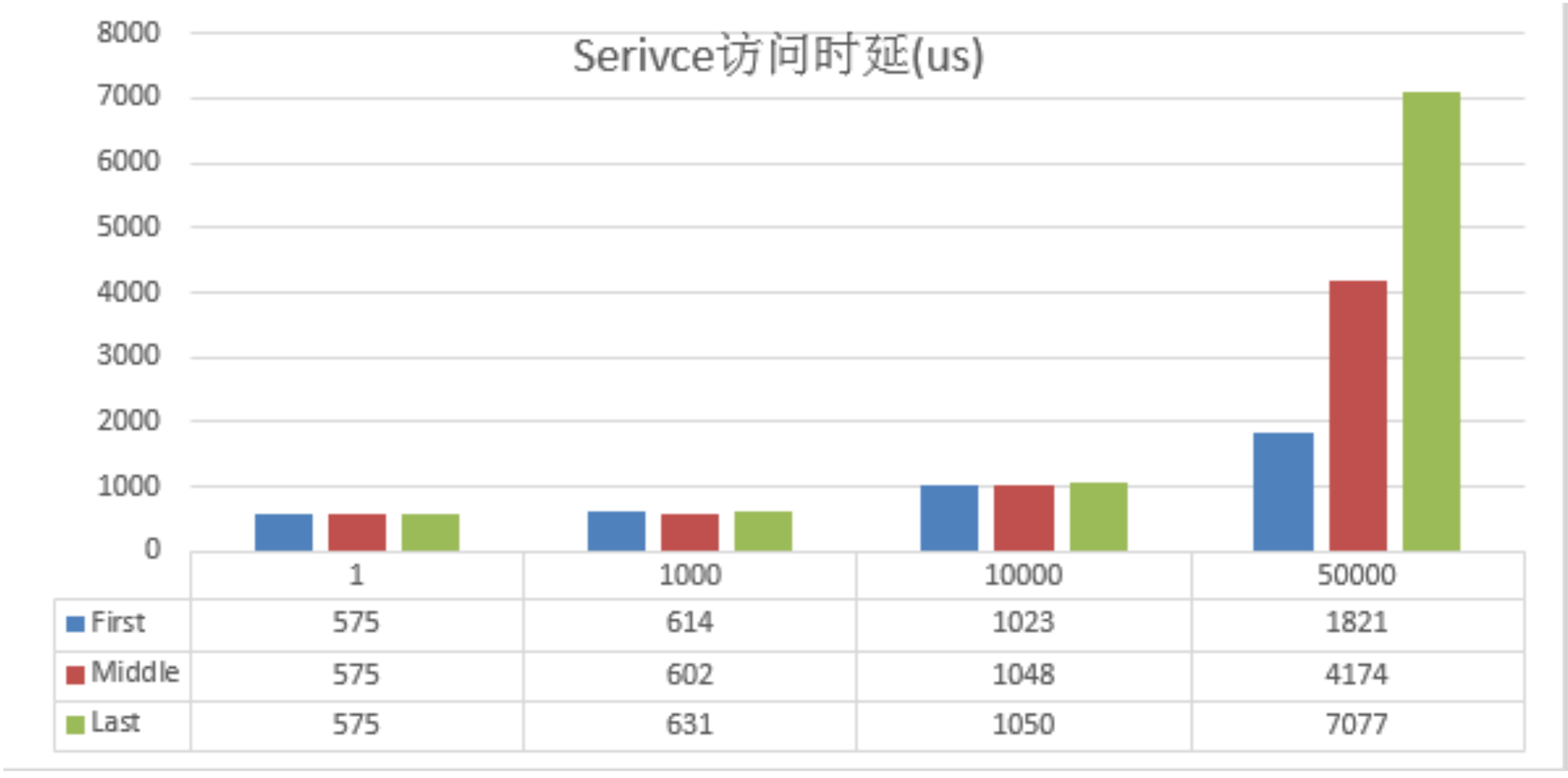
  当系统存在大量iptables规则链时，增加/删除规则会出现kernel lock

Another app is currently holding the xtables lock. Perhaps you want to use the -w option?

- **可用性**

  后端实例扩容，服务会话保持时间更新等都会导致连接断开。

Iptables规则匹配时延



	1	1000	10000	50000
First	575	614	1023	1821
Middle	575	602	1048	4174
Last	575	631	1050	7077

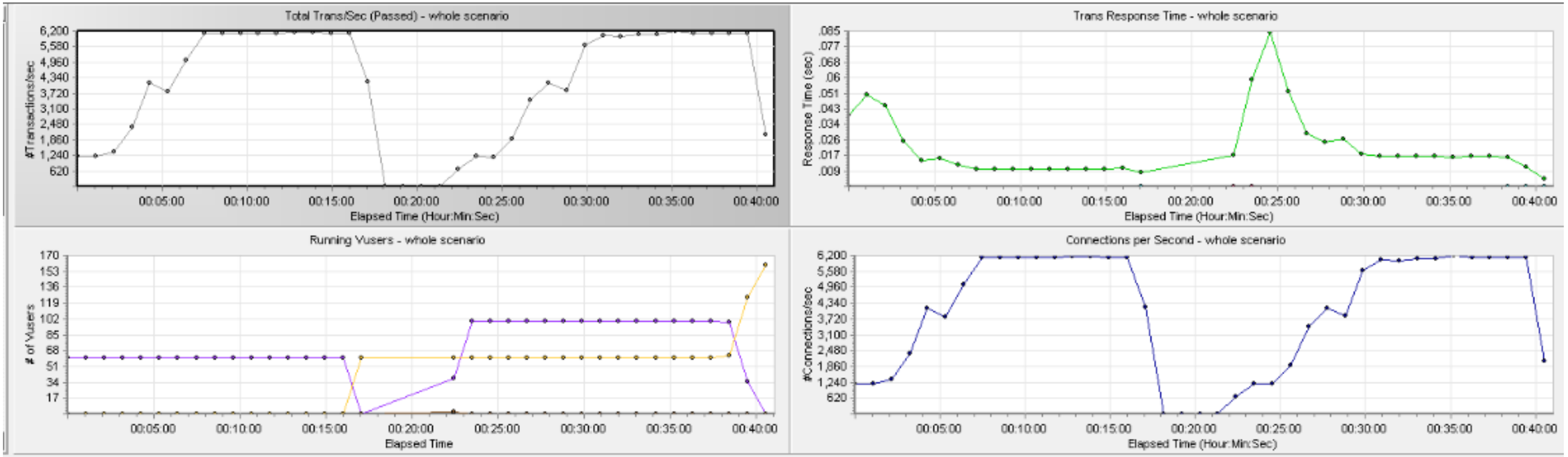
注：上面测试中，每个service在kube-services对应1条chain

更新Iptables规则的时延

- **时延出现在哪？**
  - **非增量式，即使加上—no-flush(iptables-restore)选项**
  - **Kube-proxy定期同步iptables状态：**
    - ✓ 拷贝所有规则 iptables-save
    - ✓ 在内存中更新规则
    - ✓ 在内核中修改规则 iptables-restore
    - ✓ 规则更新期间存在kernel lock

- **5K service（40K规则），增加一条iptables规则，耗时11min**
- **20K service（160K规则），增加一条iptables规则，耗时5h**

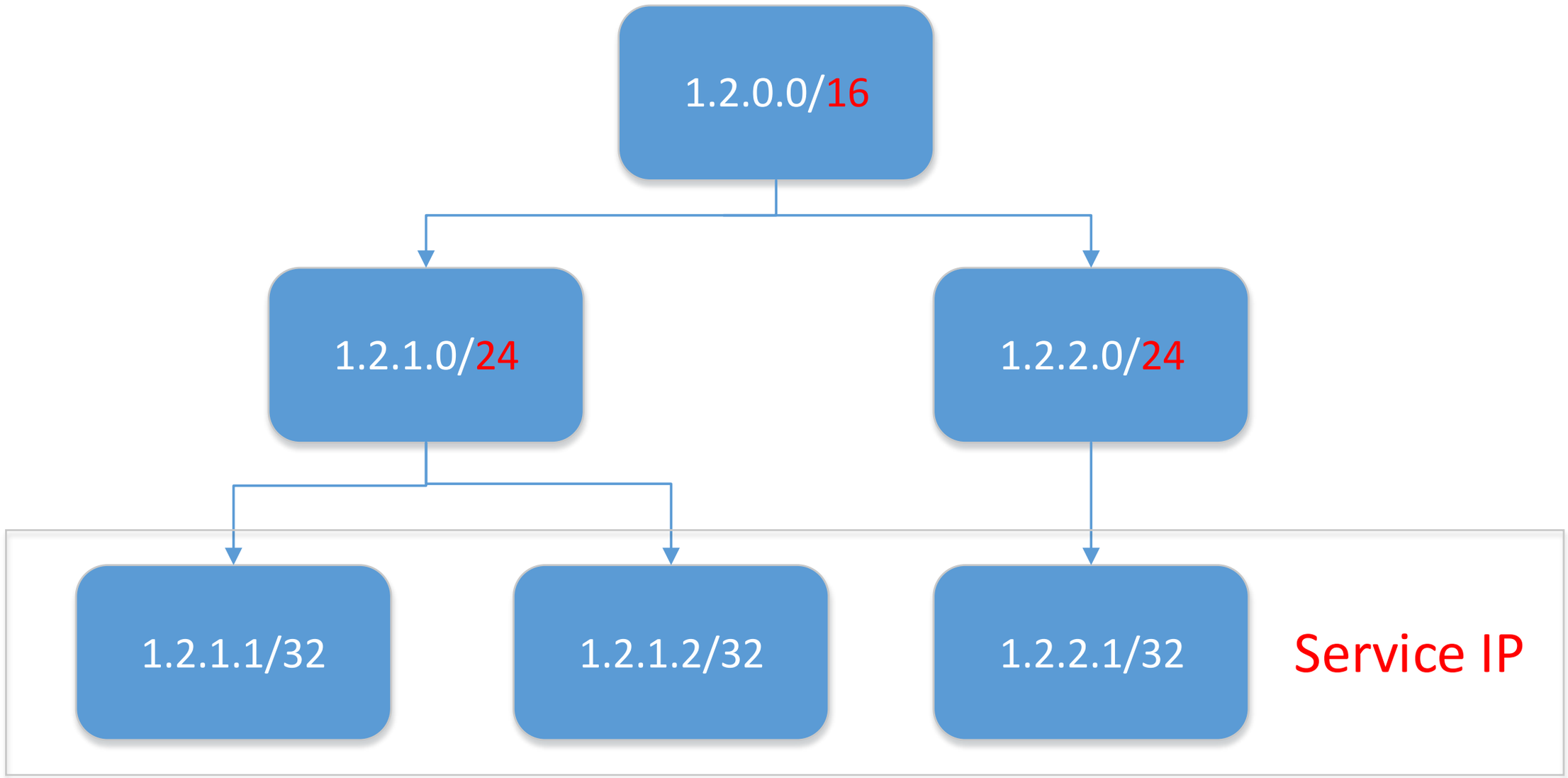
Iptables周期性刷新导致TPS抖动



优化方案

- **使用树形结构组织iptables规则**
- **IPVS**

树形结构的iptables规则



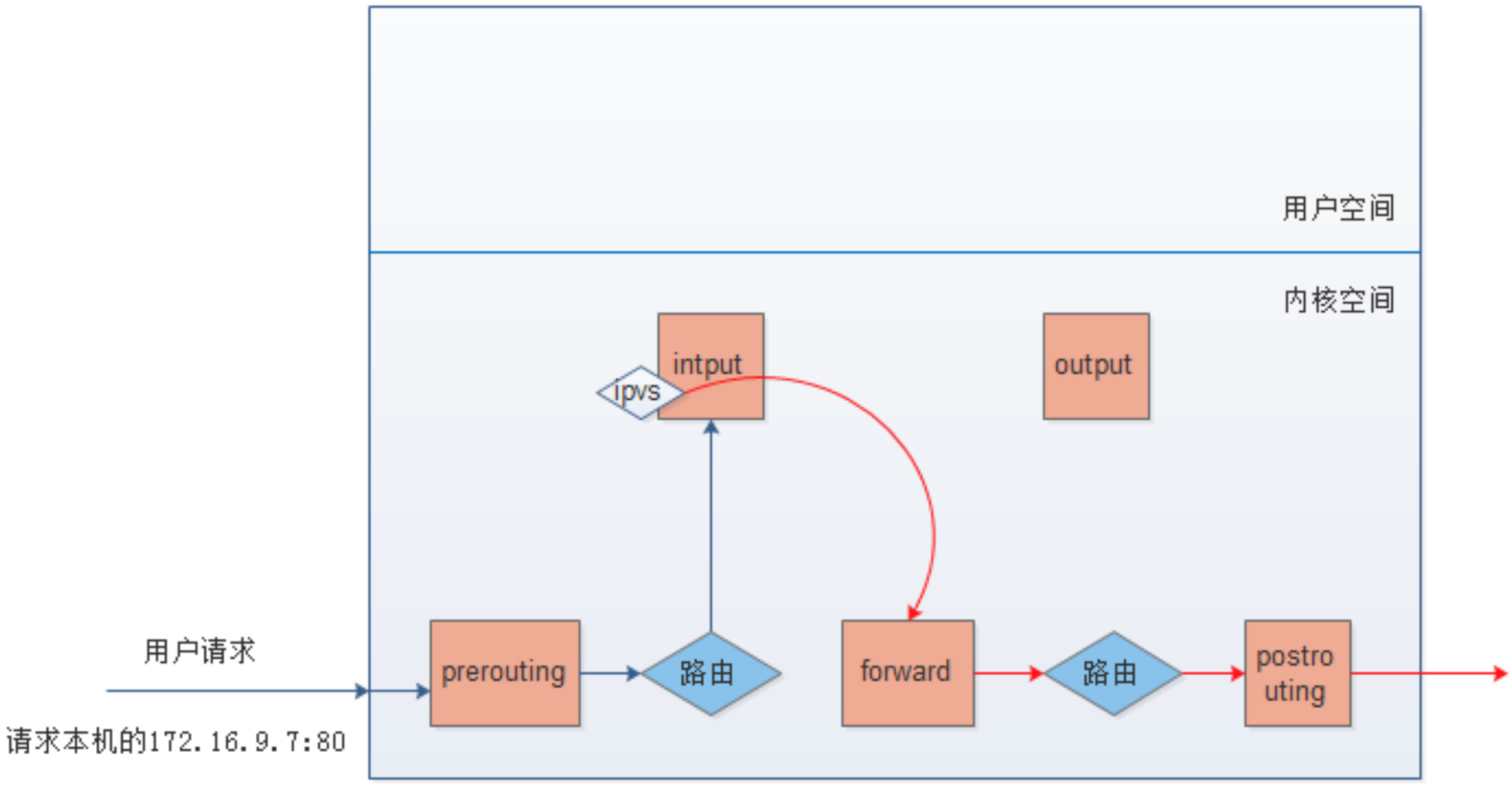
路由时间复杂度取决于搜索树的高度(m)，时间复杂度O($\sqrt[m]{N}$)

TABLE OF **CONTENTS**

什么是IPVS（IP Virtual Server）

- Linux内核实现的L4 LB，LVS负载均衡的实现
- 基于netfilter, hash table
- 支持TCP, UDP，SCTP协议，IPV4，IPV6
- 支持多种负载均衡策略
  - rr, wrr, lc, wlc, sh, dh, lblc…
- 支持会话保持
  - persistent connection调度算法

IPVS工作流

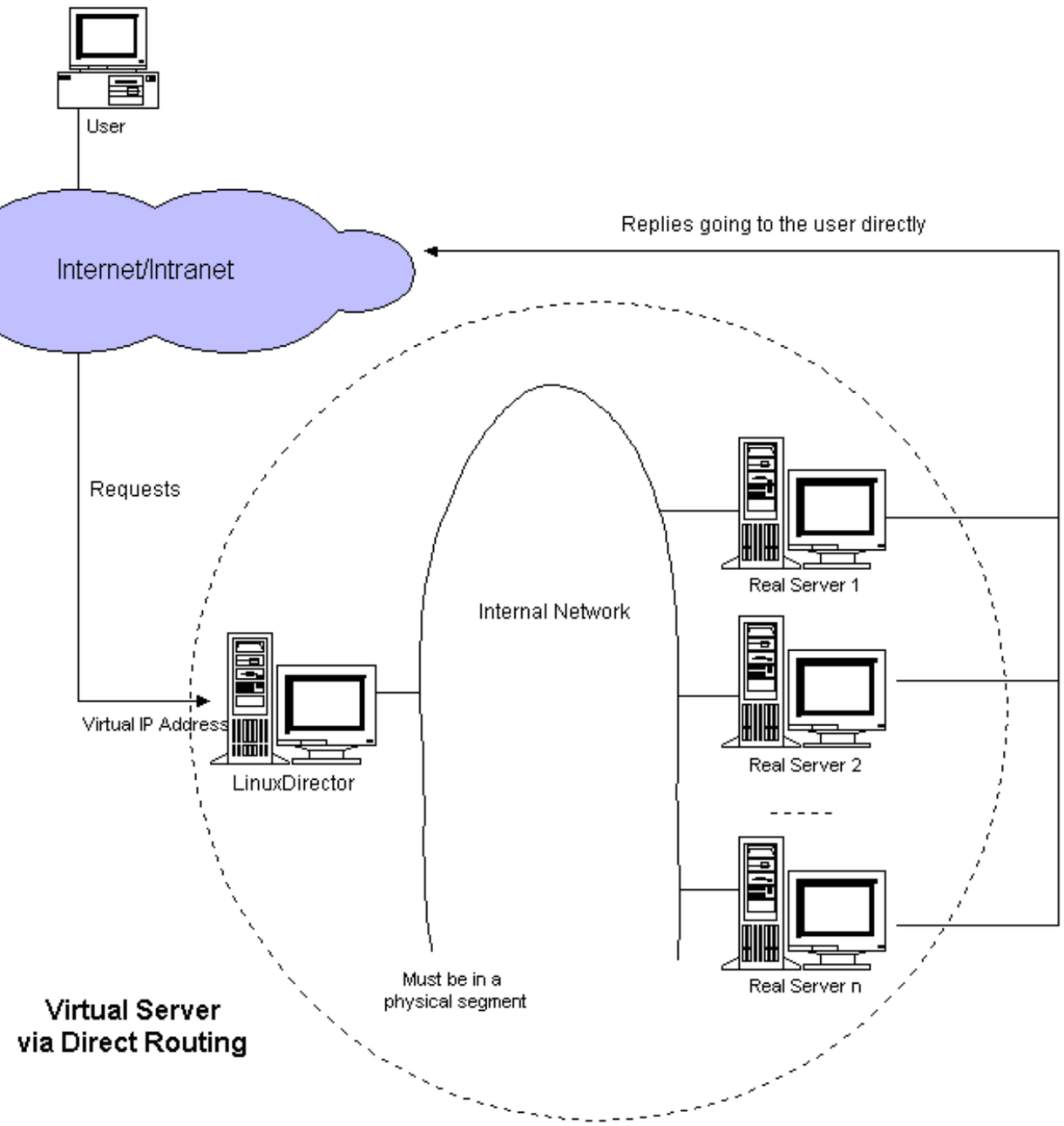


LVS工作原理

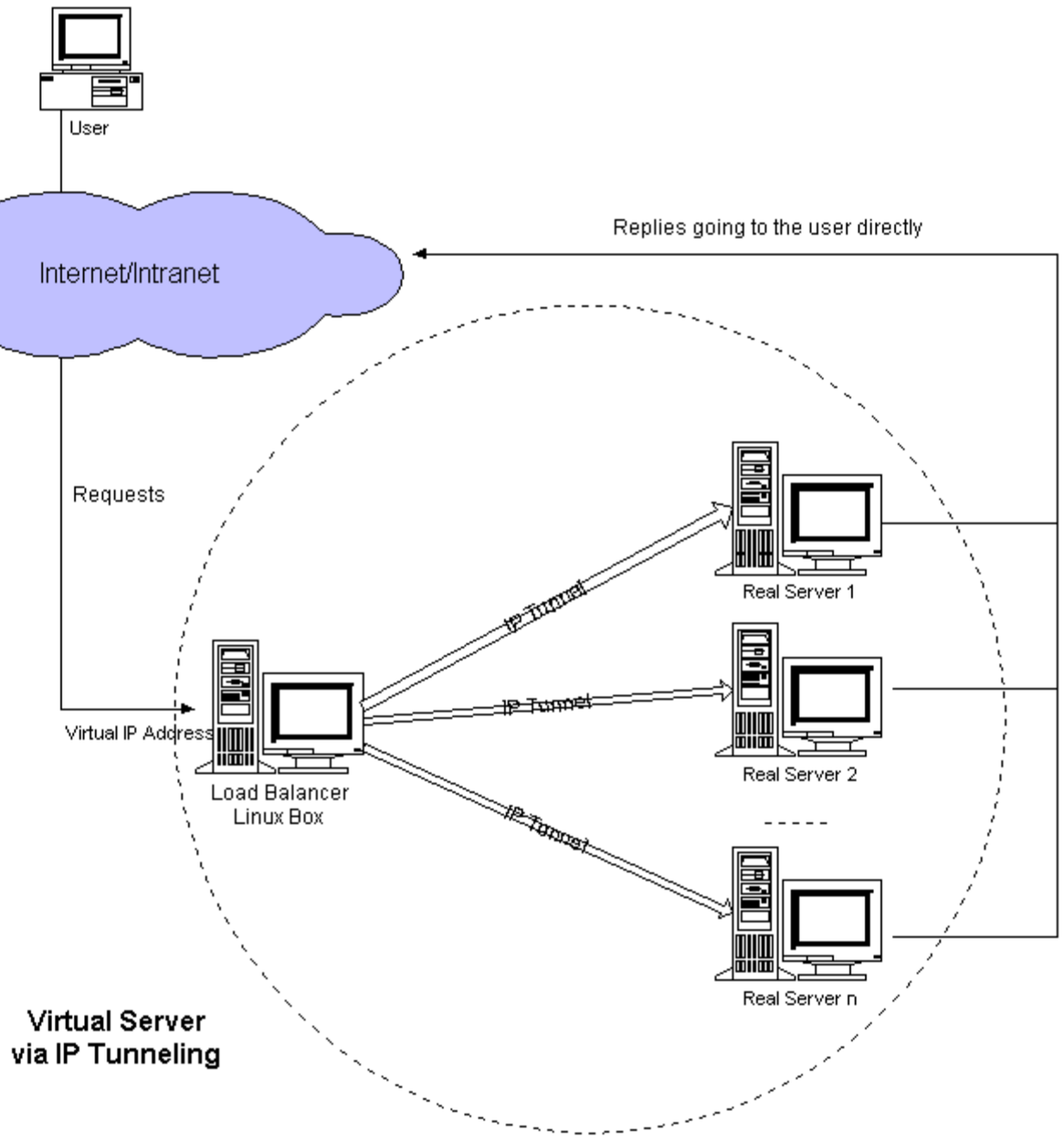
IPVS三种转发模式

- 支持三种LB模式: Direct Routing(DR), Tunneling, NAT
  - DR模式工作在L2，最快，但不支持端口映射
  - Tunneling模式用IP包封装IP包，不支持端口映射
  - DR和Tunneling模式，回程报文不会经过IPVS Director
  - NAT模式支持端口映射，回程报文经过IPVS Director - 内核原生版本只做DNAT，不做SNAT

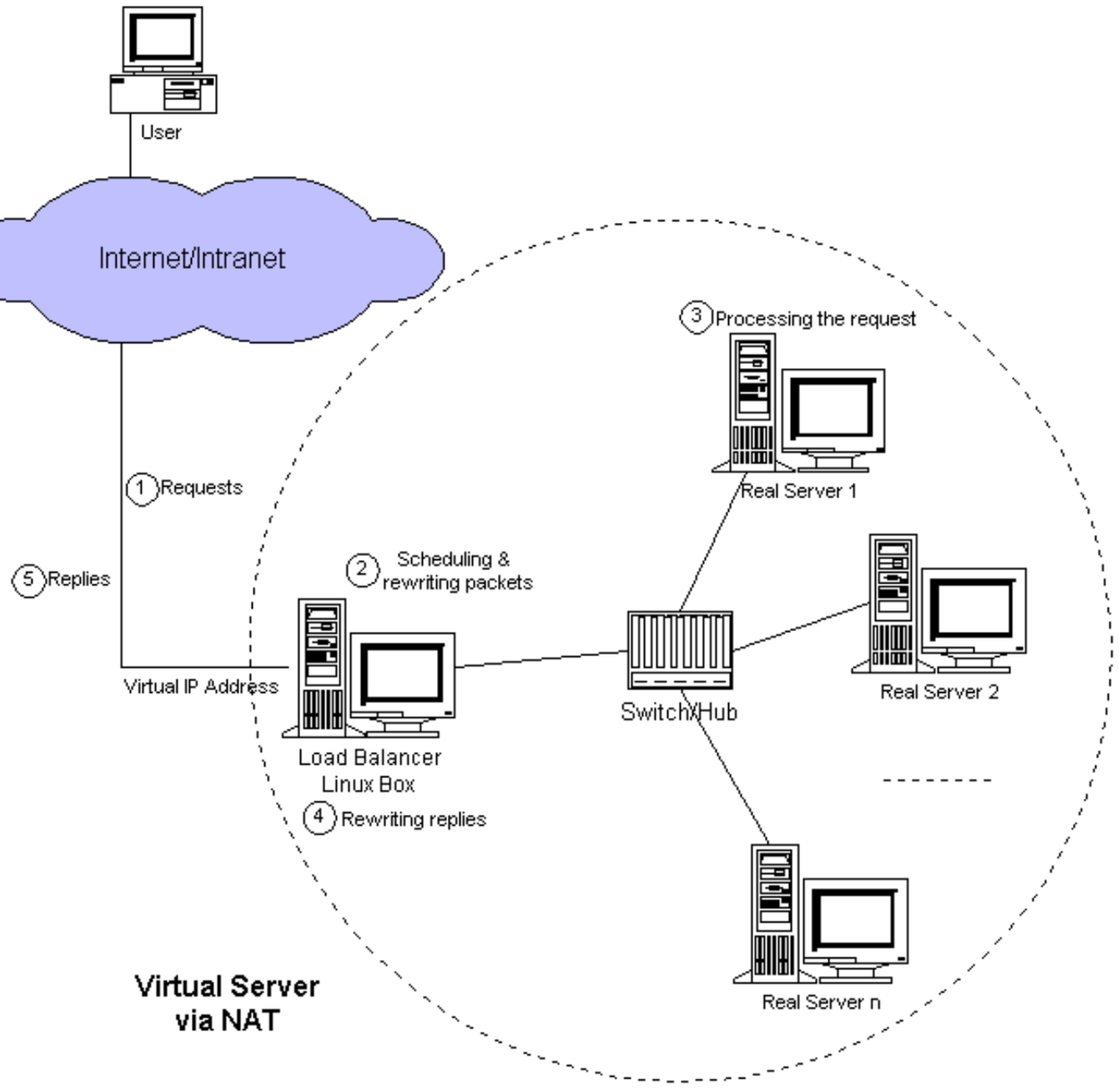
DR


User
Internet/Intranet
Replies going to the user directly
Requests
Virtual IP Address
LinuxDirector
Internal Network
Real Server 1
Real Server 2
Real Server n
Must be in a
physical segment
Virtual Server
via Direct Routing

Tunneling


User
Internet/Intranet
Replies going to the user directly
Requests
Virtual IP Address
Load Balancer
Linux Box
IP Tunnel
IP Tunnel
IP Tunnel
Real Server 1
Real Server 2
Real Server n
Virtual Server
via IP Tunneling

NAT


User
Internet/Intranet
1 Requests
5 Replies
Virtual IP Address
2 Scheduling & rewriting packets
Load Balancer Linux Box
4 Rewriting replies
SwitchHub
3 Processing the request
Real Server 1
Real Server 2
Real Server n
Virtual Server via NAT

IPVS做流量转发

- **绑定VIP**

✓ **dummy网卡**

```
# ip link add dev dummy0 type dummy
# ip addr add 192.168.2.2/32 dev dummy0
```

✓ **本地路由表**

```
# ip route add to local 192.168.2.2/32 dev eth0 proto kernel
```

✓ **网卡别名**

```
# ifconfig eth0:1 192.168.2.2 netmask 255.255.255.255 up
```

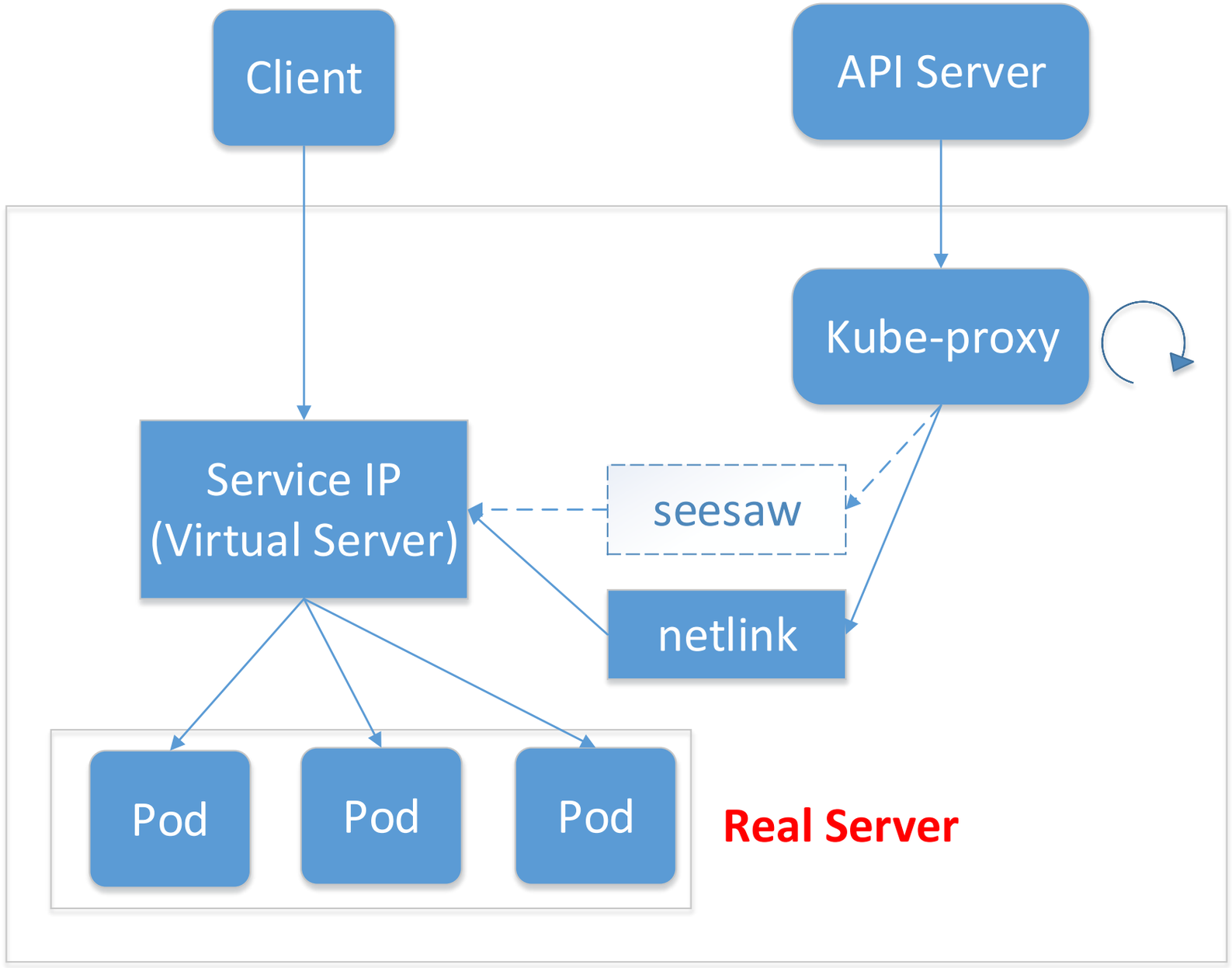
- **IPVS Virtual Server**

```
# ipvsadm -A -t 192.168.60.200:80 -s rr -p 600
```

- **IPVS Real Server**

```
# ipvsadm -a -t 192.168.60.200:80 -r 172.17.1.2:80 –m
# ipvsadm -a -t 192.168.60.200:80 -r 172.17.2.3:80 –m
```

方案实现


Client
API Server
Kube-proxy
Service IP
(Virtual Server)
seesaw
netlink
Pod
Pod
Pod
Real Server

IPVS实现Kubernetes Service

```
Name:              nginx-service
Type:              ClusterIP
IP:                10.102.128.4
Port:              http    80/TCP
Endpoints:         10.244.0.235:8080,10.244.1.237:8080
Session Affinity:   10800

# ip addr
73: kube-ipvs0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 1a:ce:f5:5f:c1:4d brd ff:ff:ff:ff:ff:ff
    inet 10.102.128.4/32 scope global kube-ipvs0
       valid_lft forever preferred_lft forever

# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.102.128.4:80 rr persistent 10800
  -> 10.244.0.235:8080            Masq    1      0          0
  -> 10.244.1.237:8080            Masq    1      0          0
```

Kubernetes支持IPVS模式

- [merged] PR #46580
- 16K LOCs
- 200+ 讨论
- 社区1.8 Alpha特性, Owner @Huawei，目标1.9进beta
- 支持ClusterIP，NodePort，External IP，Load Balancer，OnlyLocalNode…
- 依赖iptables做SNAT和访问控制

TABLE OF **CONTENTS**

Iptables vs. IPVS 增加规则时延

Service基数	1	5000	20000
Rules基数	8	40000	160000
增加1条Iptables规则	50 us	11 min	5 hours
增加1条IPVS规则	30 us	50 us	70 us

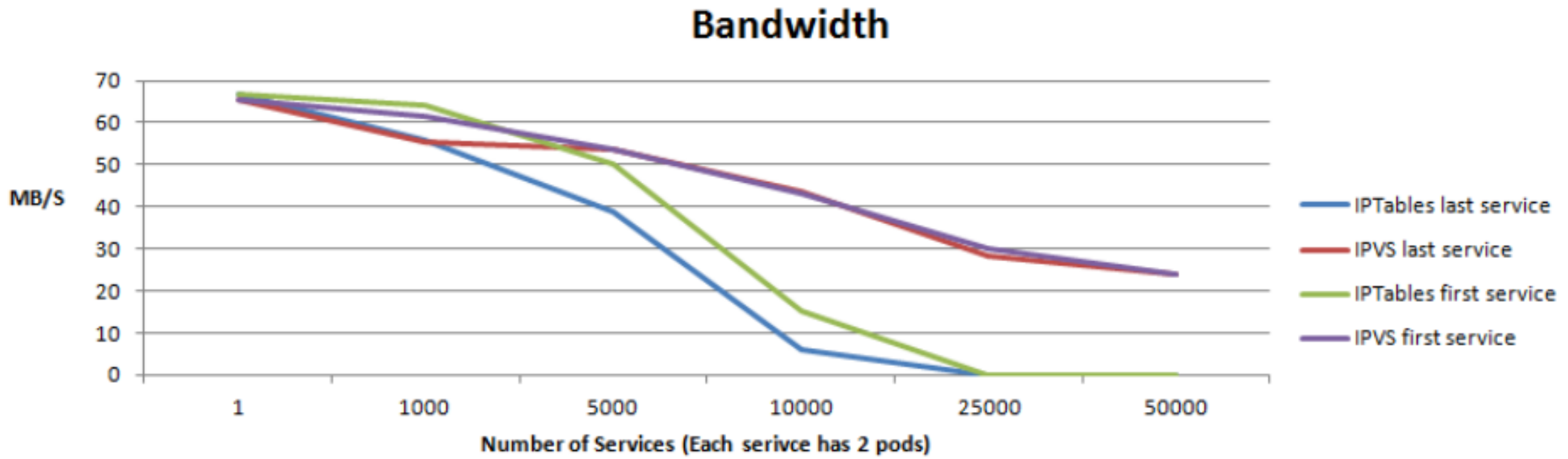
观察结果：

✓ 增加一条Iptables的时延，随着规则数的增加“指数”上升

✓ 增加一条IPVS的时延，规则基数对其几乎没影响

Iptables vs. IPVS 网络带宽

- 使用iperf测量
- 每个Service暴露4个端口（KUBE-SERVICES下挂4条KUBE-SVC-*）



service数	1	1000	5000	10000	25000	50000
带宽,iptables,first	66.6	64	50	15	0	0
带宽,iptables,last	66.6	56	38.6	6	0	0
带宽,IPVS,first	65.3	61.7	53.5	43	30	24
带宽,IPVS,last	65.3	55.3	53.8	43.5	28.5	23.8

Iptables vs. IPVS CPU/内存消耗

Metrics	number of service	IPVS	Iptables
Memory Usage	1000	386 MB	1.1 G
	5000	N/A	1.9 G
	10000	542 MB	2.3 G
	15000	N/A	OOM
	50000	1272 MB	OOM
CPU Usage	1000	0%	N/A
	5000		50% - 85%
	10000		50%-100%
	15000		N/A
	50000		N/A

Iptables vs. IPVS

- **Iptables**
- ✓ 灵活，功能强大
- ✓ 在prerouting, postrouting, forward, input, output不同阶段都能对包进行操作
- **IPVS**
- ✓ 更好的性能(hash vs. chain)
- ✓ 更多的负载均衡算法
  - rr, wrr, lc, wlc, ip hash…
- ✓ 连接保持
  - IPVS service更新期间，保持连接不断开
- ✓ 预先加载内核模
  - nf_conntrack_ipv4, ip_vs, ip_vs_rr, ip_vs_wrr, ipvs_sh
- ✓ # echo 1 > /proc/sys/net/ipv4/vs/conntrack
- ✓ 原生不支持port range

THANKS!

智 能 时 代 的 新 运 维

CNUTCon 2017