Kubernetes

Suresh Kumar Ponnusamy

October 8, 2019

Outline

Topic

Topic

Network basics I

- Introduction
  See here for a run-down of various concepts:
  `https://jvns.ca/blog/2016/12/22/container-networking/`
- Network interface(s)

```
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 12:86:18:6c:53:40 brd ff:ff:ff:ff:ff:ff
    inet 172.16.208.20/24 brd 172.16.208.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::1086:18ff:fe6c:5340/64 scope link
       valid_lft forever preferred_lft forever
```

  - MAC address
  - IP Address
    - Subnet
- IP address obtained using DHCP
  - Broadcast DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK
- Communication

Network basics II

- Route table

```
# ip route show
default via 172.16.208.1 dev eth0
169.254.169.254 dev eth0
172.16.208.0/24 dev eth0  proto kernel  scope link  src 172
172.17.0.0/16 dev docker0  proto kernel  scope link  src 17
```

- Within Subnet
  - ARP
  - Neighbour or Layer 2 switch
- Outside of subnet
  - ARP
  - Gateway/Layer 3 switch

- Cloud and multi-tenancy
  - How do we provide the network to multiple unrelated customers with them not stepping on each other?
    - Tunneling intro: `https://en.wikipedia.org/wiki/Tunneling_protocol`

Network basics III

- VXLAN: `https://tools.ietf.org/html/rfc7348`
- IPIP: `https://tools.ietf.org/html/rfc2003`
- GRE: `https://tools.ietf.org/html/rfc2784`
- NVGRE: `https://tools.ietf.org/html/rfc7637`
- EVPN: `https://tools.ietf.org/html/rfc8365`
- Cost of cloud: `https://www.microsoft.com/en-us/research/wp-content/uploads/2009/01/p68-v39n1o-greenberg.pdf`
- etc

AWS I

- VPC
  - A virtual network within region
  - CIDR range(s)
    - x.x.x.0 reserved, network address
    - x.x.x.1 reserved, VPC router
    - x.x.x.2 reserved, DNS address
    - x.x.x.3 reserved
    - x.x.x.255 reserved, broadcast address
  - Subnets
    - can only span one AZ == (Multi)
  - Route tables
    - main route table
    - custom route table
  - ACLs/NCLs
  - No broadcast support!
    - DHCP, ARP etc has special handling

AWS II

- Internals: `https://www.youtube.com/watch?v=3qln2u1Vr2E&feature=youtu.be`
  - Looks like "Subnet ~= VLAN" and "VPC ~= VRF", so why not use standard hardware?
    - VLAN_ID problem (12 bits, 4096 VLANs)
    - VRF size limitations
    - Control plane scaling problems
  - Virtual network completely decoupled from physical network
- Latency Measurements on us-east, on m5.4xlarge machine

```
netperf -v 2 -t TCP_RR -H $TARGET_HOST -l 60
```

What	Latency (µs)	Comment
Intra AZ	92	Within us-east-1b
Inter AZ	445	us-east-1a to us-east-1b

Linux network internals I

- Introduction
  - At high level
    - NIC -> Network stack -> routing or deliver to usespace
    - How packets are delivered to network stack Interrupt driven: Multiple RX queues, multile IRQs, NAPI polling, Busy polling
  - Increasing incoming throughput - Large Receive Offload (LRO)
    - Generic Receive Offload (GRO): group incoming packets belonging to same stream and process them together
    - See here: `https://medium.com/netflix-techblog/serving-100-gbps-from-an-open-connect-appliance-cdb51dda3b99`
  - Increase outgoing throughput - Large Send Offload (LSO)
    - Generic Segmentation Offload (GSO) / TCP Segmentation Offload (TSO)
- Internals
  - Physical and software networks (veth, bridge etc)
  - Network namespaces
  - VLAN: Overlay network

Linux network internals II

- Bonding: redundancy / aggregation
- veth (virtual ethernet)
- Bridge (layer 2)
  - Flood on all ports
  - Spanning Tree Protocol (STP)
- MACVLAN
  - Multiple MAC addresses on single interface
- IPVLAN
  - Multiple containers within single MAC Address
- TUN (L2)
- TAP (L3)
- vxlan
- ipsec
- BPF
- XDP
  -
- References

Linux network internals III

- https://www.slideshare.net/ThomasGraf5/linuxcon-2015-linux-kernel-networking-walkthrough
- https://www.slideshare.net/ThomasGraf5/linux-networking-explained

Container networking

- https://www.youtube.com/watch?v=bfE_pQS4JPg
- https://www.youtube.com/watch?v=Yf_INdTWIHI

Kubernetes networking I

- Introduction
  - https://itnext.io/an-illustrated-guide-to-kubernetes-networking-part-1-d1ede3
  - https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/
  - Life of a packet: https://www.youtube.com/watch?v=0Omvgd7Hg1I
  - Hands on: https://www.youtube.com/watch?v=3jaZlwM-2rs&list=PLAz0FOwiBi6tVRl4bPbs_G_ucM3N7a1ES
- AWS EKS CNI
  - https://github.com/aws/amazon-vpc-cni-k8s/blob/master/docs/cni-proposal.md
  - https://www.slideshare.net/AmazonWebServices/deep-dive-on-container-networking-at-scale-on-amazon-eks-a
- Why overlay network
- CNIs
- EKS CNI

Kubernetes networking II

```
[root@ip-172-16-80-208 ec2-user]# docker ps --format "{{.ID}} {{.Command}} {{.Names}}" | grep busybox
bb795e9f26e0 "sh" k8s_busybox_busybox_default_c0073dc2-e5af-11e9-97c7-121ce268d264_0
7d41f449bc2e "/pause" k8s_POD_busybox_default_c0073dc2-e5af-11e9-97c7-121ce268d264_0

[root@ip-172-16-80-208 ec2-user]# docker inspect --format '{{.State.Pid}}' 7d41f449bc2e
17102

######################### Inside pod #########################
[root@ip-172-16-80-208 ec2-user]# nsenter -n -t 17102 ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: eth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc noqueue state UP group default
    link/ether e2:52:72:b6:05:fd brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.16.80.156/32 brd 172.16.80.156 scope global eth0
       valid_lft forever preferred_lft forever

[root@ip-172-16-80-208 ec2-user]# nsenter -n -t 17102 ip route show
default via 169.254.1.1 dev eth0
169.254.1.1 dev eth0 scope link

[root@ip-172-16-80-208 ec2-user]# nsenter -n -t 17102 ip rule list
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default

[root@ip-172-16-80-208 ec2-user]# nsenter -n -t 17102 arp -a
gateway (169.254.1.1) at 6a:16:85:38:bc:e0 [ether] PERM on eth0
ip-172-16-80-208.ec2.internal (172.16.80.208) at 6a:16:85:38:bc:e0 [ether] on eth0
```

Kubernetes networking III

```
######################### On the host ##########################
[root@ip-172-16-80-208 ec2-user]# ip rule list
0:      from all lookup local
512:    from all to 172.16.80.212 lookup main
512:    from all to 172.16.80.201 lookup main
512:    from all to 172.16.80.146 lookup main
512:    from all to 172.16.80.130 lookup main
512:    from all to 172.16.80.193 lookup main
512:    from all to 172.16.80.242 lookup main
512:    from all to 172.16.80.24 lookup main
512:    from all to 172.16.80.156 lookup main
1024:   from all fwmark 0x80/0x80 lookup main
32766:  from all lookup main
32767:  from all lookup default

[root@ip-172-16-80-208 ec2-user]# ip route show table main
default via 172.16.80.1 dev eth0
169.254.169.254 dev eth0
172.16.80.0/24 dev eth0 proto kernel scope link src 172.16.80.208
172.16.80.24 dev eni3c105fa2c78 scope link
172.16.80.130 dev eni709a9477bbd scope link
172.16.80.146 dev eni67b8b59cdbd scope link
172.16.80.156 dev eni12d4a061371 scope link
172.16.80.193 dev enib88a3c82709 scope link
172.16.80.201 dev eni2b4d5fb8dba scope link
172.16.80.212 dev enie2f0d94a658 scope link
172.16.80.242 dev enie37ffccef01 scope link
```

Kube service internals: Life of a http request I

- Background
  - Kubernetes services
    - https://kubernetes.io/docs/concepts/services-networking/service/
    - https://www.slideshare.net/Docker/deep-dive-in-container-service-discovery
    - https://msazure.club/kubernetes-services-and-iptables/
  - kube-proxy iptables mode
    - https://github.com/kubernetes/kubernetes/blob/3c0bc3c5adc36c0e89b9dc537f1585ff47314689/pkg/proxy/iptables/proxier.go#L683
- Setup
  - 'echoheaders' app

Kube service internals: Life of a http request II

```
kubectl apply -f - <<EOF

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    run: echoheaders
  name: echoheaders
spec:
  replicas: 10
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      run: echoheaders
  template:
    metadata:
      labels:
        run: echoheaders
    spec:
      containers:
      - image: k8s.gcr.io/echoserver:1.4
        imagePullPolicy: IfNotPresent
        name: echoheaders
        ports:
        - containerPort: 8080
          protocol: TCP
        resources: {}
      dnsPolicy: ClusterFirst
      restartPolicy: Always
EOF
```

Kube service internals: Life of a http request III

- 'echoheaders' service

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Service
metadata:
  labels:
    run: echoheaders
  name: echoheaders
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    run: echoheaders
  type: ClusterIP
EOF
```

- 'busybox' terminal

```
$ kubectl run -i --tty busybox --image=busybox --restart=Never -- sh
If you don't see a command prompt, try pressing enter.
/ #
```

- Internals
  - Pods

Kube service internals: Life of a http request IV

```
$ kubectl get pods -l 'run=echoheaders' -o json | jq '.items|.[]|.status.podIP'
"172.16.82.10"
"172.16.82.54"
"172.16.80.193"
"172.16.81.86"
"172.16.81.129"
"172.16.81.248"
"172.16.82.91"
"172.16.80.242"
"172.16.80.130"
"172.16.80.24"
```

- Service and endpoints

```
$ kubectl describe services echoheaders
Name:              echoheaders
...
...
Type:              ClusterIP
IP:                10.100.92.97
Port:              <unset>  80/TCP
...
...

$ kubectl describe endpoints echoheaders
Name:         echoheaders
...
...
Subsets:
  Addresses:          172.16.80.130,172.16.80.193,172.16.80.24,172.16.80.242,172.16.81.129,172.16
  NotReadyAddresses:  <none>
```

Kube service internals: Life of a http request V

```
  Ports:
    Name     Port  Protocol
    ----     ----  --------
    <unset>  8080  TCP
...
...
```

- iptables

```
$ kubectl get pods -o wide | grep busybox
busybox                          1/1     Running   0          115s   172.16.80.156   ip-172-16-80-2

$ ssh -i ~/.ssh/node.pem  ec2-user@ip-172-16-80-208.ec2.internal
[ec2-user@ip-172-16-80-208 ~]$ sudo su

[root@ip-172-16-80-208 ec2-user]# iptables-save
...
...
*nat
:PREROUTING ACCEPT [40:3447]
:INPUT ACCEPT [11:660]
:OUTPUT ACCEPT [21:1276]
:POSTROUTING ACCEPT [35:3147]
...
...
:KUBE-MARK-DROP - [0:0]
:KUBE-MARK-MASQ - [0:0]
:KUBE-NODEPORTS - [0:0]
:KUBE-POSTROUTING - [0:0]
:KUBE-SEP-CVYM5BDNXHDZCPXO - [0:0]
:KUBE-SEP-FI3XCYYICK5OABRW - [0:0]
```

Kube service internals: Life of a http request VI

```
:KUBE-SEP-KZSOVFOE46D5TP4N - [0:0]
:KUBE-SEP-RW7KL2NJOFWM66TU - [0:0]
:KUBE-SEP-S7IDRVJCQL6J2T3B - [0:0]
:KUBE-SEP-VFPFQDI2HSZFHEV6 - [0:0]
:KUBE-SEP-WRHHNELO4CLB4A4H - [0:0]
:KUBE-SEP-XF5F3RDE5OCU6I4H - [0:0]
:KUBE-SEP-YVJ73CWV53T5LCRW - [0:0]
:KUBE-SEP-YVK2IVPA77O4VBIG - [0:0]
:KUBE-SVC-OZ62J7OHEXWZ4NCX - [0:0]
...
...
-A PREROUTING -m comment --comment "kubernetes service portals" -j KUBE-SERVICES
-A OUTPUT -m comment --comment "kubernetes service portals" -j KUBE-SERVICES
-A POSTROUTING -m comment --comment "kubernetes postrouting rules" -j KUBE-POSTROUTING
...
...
-A KUBE-MARK-DROP -j MARK --set-xmark 0x8000/0x8000
-A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
...
...
-A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic requiring SNAT" -m mark --ma
...
...
-A KUBE-SEP-CVYM5BDNXHDZCPXO -s 172.16.82.54/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-CVYM5BDNXHDZCPXO -p tcp -m tcp -j DNAT --to-destination 172.16.82.54:8080
-A KUBE-SEP-FI3XCYYICK5OABRW -s 172.16.80.130/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-FI3XCYYICK5OABRW -p tcp -m tcp -j DNAT --to-destination 172.16.80.130:8080
-A KUBE-SEP-KZSOVFOE46D5TP4N -s 172.16.82.91/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-KZSOVFOE46D5TP4N -p tcp -m tcp -j DNAT --to-destination 172.16.82.91:8080
-A KUBE-SEP-RW7KL2NJOFWM66TU -s 172.16.80.24/32 -j KUBE-MARK-MASQ
```

Kube service internals: Life of a http request VII

```
-A KUBE-SEP-RW7KL2NJOFWM66TU -p tcp -m tcp -j DNAT --to-destination 172.16.80.24:8080
-A KUBE-SEP-S7IDRVJCQL6J2T3B -s 172.16.81.248/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-S7IDRVJCQL6J2T3B -p tcp -m tcp -j DNAT --to-destination 172.16.81.248:8080
-A KUBE-SEP-VFPFQDI2HSZFHEV6 -s 172.16.80.193/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-VFPFQDI2HSZFHEV6 -p tcp -m tcp -j DNAT --to-destination 172.16.80.193:8080
-A KUBE-SEP-WRHHNELO4CLB4A4H -s 172.16.81.129/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-WRHHNELO4CLB4A4H -p tcp -m tcp -j DNAT --to-destination 172.16.81.129:8080
-A KUBE-SEP-XF5F3RDE5OCU6I4H -s 172.16.80.242/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-XF5F3RDE5OCU6I4H -p tcp -m tcp -j DNAT --to-destination 172.16.80.242:8080
-A KUBE-SEP-YVJ73CWV53T5LCRW -s 172.16.82.10/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-YVJ73CWV53T5LCRW -p tcp -m tcp -j DNAT --to-destination 172.16.82.10:8080
-A KUBE-SEP-YVK2IVPA77O4VBIG -s 172.16.81.86/32 -j KUBE-MARK-MASQ
-A KUBE-SEP-YVK2IVPA77O4VBIG -p tcp -m tcp -j DNAT --to-destination 172.16.81.86:8080
-A KUBE-SERVICES -d 10.100.92.97/32 -p tcp -m comment --comment "default/echoheaders: cluster IP"
...
...
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.10000000009 -j KUBE-SEP-F
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.11110999994 -j KUBE-SEP-V
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.12500000000 -j KUBE-SEP-R
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.14286000002 -j KUBE-SEP-X
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.16667000018 -j KUBE-SEP-W
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.20000000019 -j KUBE-SEP-S
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.25000000000 -j KUBE-SEP-Y
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.33332999982 -j KUBE-SEP-Y
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-C
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -j KUBE-SEP-KZSOVFOE46D5TP4N
...
...
```

Kube service internals: Life of a http request VIII

Example traversal, assume that a pod is trying to consume the service (aka "curl `http://my-service-ip/`"):

- It will hit NAT output chain, which forwards it to KUBE-SERVICES

```
*nat
-A OUTPUT -m comment --comment "kubernetes␣service␣portals" -j KUBE-
    SERVICES
```

- which will check if destination IP is 10.100.92.97, port is 80 and transport TCP, then it will forward it to KUBE-SVC-OZ62J7OHEXWZ4NCX

```
-A KUBE-SERVICES -d 10.100.92.97/32 -p tcp -m comment --comment "default
    /echoheaders:␣cluster␣IP" -m tcp --dport 80 -j KUBE-SVC-
    OZ62J7OHEXWZ4NCX
```

- which will randomly choose (based on the probability configured), assume, KUBE-SEP-YVK2IVPA77O4VBIG is chosen for us

Kube service internals: Life of a http request IX

```
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.10000000009 -j KUBE-SEP-FI3XCYYICK5OABRW
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.11110999994 -j KUBE-SEP-VFPFQDI2HSZFHEV6
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.12500000000 -j KUBE-SEP-RW7KL2NJOFWM66TU
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.14286000002 -j KUBE-SEP-XF5F3RDE5OCU6I4H
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.16667000018 -j KUBE-SEP-WRHHNELO4CLB4A4H
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.20000000019 -j KUBE-SEP-S7IDRVJCQL6J2T3B
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.25000000000 -j KUBE-SEP-YVK2IVPA77O4VBIG
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.33332999982 -j KUBE-SEP-YVJ73CWV53T5LCRW
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -m statistic --mode random --probability
    0.50000000000 -j KUBE-SEP-CVYM5BDNXHDZCPXO
-A KUBE-SVC-OZ62J7OHEXWZ4NCX -j KUBE-SEP-KZSOVFOE46D5TP4N
```

- which will mark the packet, is source ip is the using KUBE-MARK-MASQ jump target (will not happen for us in this flow)

```
-A KUBE-SEP-YVK2IVPA77O4VBIG -s 172.16.81.86/32 -j KUBE-MARK-MASQ
-A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
```

Kube service internals: Life of a http request X

- and then do a DNAT to 172.16.81.86:8080

```
-A KUBE-SEP-YVK2IVPA77O4VBIG -p tcp -m tcp -j DNAT --to-destination
    172.16.81.86:8080
```

- and then go to POSTROUTING chain and do SNAT if mark is set (will not happen for us in this flow)

```
-A POSTROUTING -m comment --comment "kubernetes␣postrouting␣rules" -j
    KUBE-POSTROUTING
-A KUBE-POSTROUTING -m comment --comment "kubernetes␣service␣traffic␣
    requiring␣SNAT" -m mark --mark 0x4000/0x4000 -j MASQUERADE
```

- Life of a request
  - Command

Kube service internals: Life of a http request XI

```
# Run on busybox terminal

# wget http://10.100.92.97/hello -O -
Connecting to 10.100.92.97 (10.100.92.97:80)
writing to stdout
CLIENT VALUES:
client_address=172.16.80.156
command=GET
real path=/hello
query=nil
request_version=1.1
request_uri=http://10.100.92.97:8080/hello

SERVER VALUES:
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
connection=close
host=10.100.92.97
user-agent=Wget
BODY:
-                    100%
     |*******************************************************************************
        300  0:00:00 ETA
written to stdout
```

- Outgoing: from 'busybox' pod to 'echoheaders' service

Kube service internals: Life of a http request XII

- iptable rule traversal

```
'busybox' pod => nat(OUTPUT) => nat(POSTROUTING)
```

busybox Pod IP	172.16.80.156
echoheaders Service ClusterIP	10.100.92.97
node ip	172.16.80.208
Chosen echoheader Pod IP	172.16.81.86

```
[root@ip-172-16-80-208 ec2-user]# tshark -i eni12d4a061371
 83 10.461975673 172.16.80.156 -> 10.100.92.97 TCP 74 50760 > http [SYN] Se
 84 10.462551728 10.100.92.97 -> 172.16.80.156 TCP 74 http > 50760 [SYN, AC
 85 10.462595853 172.16.80.156 -> 10.100.92.97 TCP 66 50760 > http [ACK] Se
 86 10.462606293 172.16.80.156 -> 10.100.92.97 HTTP 146 GET /hello HTTP/1.1
 87 10.463057940 10.100.92.97 -> 172.16.80.156 TCP 66 http > 50760 [ACK] Se
 88 10.463224171 10.100.92.97 -> 172.16.80.156 HTTP 617 HTTP/1.1 200 OK
(text/plain)
 89 10.463232126 10.100.92.97 -> 172.16.80.156 TCP 66 http > 50760 [FIN, AC
 90 10.463261700 172.16.80.156 -> 10.100.92.97 TCP 66 50760 > http [ACK] Se
 91 10.463492130 172.16.80.156 -> 10.100.92.97 TCP 66 50760 > http [FIN, AC
 92 10.463953830 10.100.92.97 -> 172.16.80.156 TCP 66 http > 50760 [ACK] Se

[root@ip-172-16-80-208 ec2-user]# tshark -i any -f 'host 172.16.80.156' -P
Running as user "root" and group "root". This could be dangerous.

Capturing on 'any'
```

Kube service internals: Life of a http request XIII

```
144 10.007791191 172.16.80.156 -> 172.16.81.86 TCP 76 50746 > http-alt [SYN]
145 10.008374657 172.16.81.86 -> 172.16.80.156 TCP 76 http-alt > 50746 [SYN
149 10.008442132 172.16.80.156 -> 172.16.81.86 TCP 68 50746 > http-alt [ACK]
150 10.008446009 172.16.80.156 -> 172.16.81.86 HTTP 148 GET /hello HTTP/1.1
151 10.008933547 172.16.81.86 -> 172.16.80.156 TCP 68 http-alt > 50746 [ACK]
153 10.009104138 172.16.81.86 -> 172.16.80.156 HTTP 619 HTTP/1.1 200 OK
(text/plain)
155 10.009113080 172.16.81.86 -> 172.16.80.156 TCP 68 http-alt > 50746 [FIN
158 10.009135128 172.16.80.156 -> 172.16.81.86 TCP 68 50746 > http-alt [ACK]
160 10.009293474 172.16.80.156 -> 172.16.81.86 TCP 68 50746 > http-alt [FIN
161 10.009750688 172.16.81.86 -> 172.16.80.156 TCP 68 http-alt > 50746 [ACK]
```

- Incoming: from 'busybox' pod to 'echoheaders' pod
  - iptable rule traversal

```
nat(PREROUTING) => 'echoheaders' pod
```

Kube service internals: Life of a http request XIV

```
[root@ip-172-16-81-30 ec2-user]# tshark -i any -f 'host 172.16.80.156'
Running as user "root" and group "root". This could be dangerous.

Capturing on 'any'
  1 0.000000000 172.16.80.156 -> 172.16.81.86 TCP 76 50746 > http-alt [SYN]
  2 0.000056840 172.16.80.156 -> 172.16.81.86 TCP 76 [TCP Out-Of-Order] 507
  3 0.000084371 172.16.81.86 -> 172.16.80.156 TCP 76 http-alt > 50746 [SYN,
  4 0.000091565 172.16.81.86 -> 172.16.80.156 TCP 76 [TCP Out-Of-Order] htt
  5 0.000626523 172.16.80.156 -> 172.16.81.86 TCP 68 50746 > http-alt [ACK]
  6 0.000631557 172.16.80.156 -> 172.16.81.86 TCP 68 [TCP Dup ACK 5#1] 5074
  7 0.000632881 172.16.80.156 -> 172.16.81.86 HTTP 148 GET /hello HTTP/1.1
  8 0.000635384 172.16.80.156 -> 172.16.81.86 HTTP 148 [TCP Retransmission]
  9 0.000679619 172.16.81.86 -> 172.16.80.156 TCP 68 http-alt > 50746 [ACK]
 10 0.000686251 172.16.81.86 -> 172.16.80.156 TCP 68 [TCP Dup ACK 9#1] http
 11 0.000834778 172.16.81.86 -> 172.16.80.156 HTTP 619 HTTP/1.1 200 OK  (te
 12 0.000851575 172.16.81.86 -> 172.16.80.156 HTTP 619 [TCP Retransmission]
(text/plain)
 13 0.000865107 172.16.81.86 -> 172.16.80.156 TCP 68 http-alt > 50746 [FIN,
 14 0.000870337 172.16.81.86 -> 172.16.80.156 TCP 68 [TCP Out-Of-Order] htt
 15 0.001317349 172.16.80.156 -> 172.16.81.86 TCP 68 50746 > http-alt [ACK]
 16 0.001321729 172.16.80.156 -> 172.16.81.86 TCP 68 [TCP Dup ACK 15#1] 507
 17 0.001473148 172.16.80.156 -> 172.16.81.86 TCP 68 50746 > http-alt [FIN,
 18 0.001476728 172.16.80.156 -> 172.16.81.86 TCP 68 [TCP Out-Of-Order] 507
 19 0.001492076 172.16.81.86 -> 172.16.80.156 TCP 68 http-alt > 50746 [ACK]
 20 0.001499724 172.16.81.86 -> 172.16.80.156 TCP 68 [TCP Dup ACK 19#1] htt
```

- Overall flow

Kube service internals: Life of a http request XV