20221010-Pod-Pod(Different Node)-[VxLAN Mode]

```
env 介绍: [kernel: 5.11.0-051100-generic]
root@bpf1:~# uname -a
Linux bpf1 5.11.0-051100-generic #202102142330 SMP Sun Feb 14 23:33:21 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
root@bpf1:~# kubectl get nodes -o wide
NAME   STATUS   ROLES                  AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION          CONTAINER-RUNTIME
bpf1   Ready    control-plane,master   19h   v1.23.2   192.168.2.61   <none>        Ubuntu 20.04.3 LTS   5.11.0-051100-generic   docker://20.10.12
bpf2   Ready    <none>                 19h   v1.23.2   192.168.2.62   <none>        Ubuntu 20.04.3 LTS   5.11.0-051100-generic   docker://20.10.12
root@bpf1:~#
root@bpf1:~# kubectl -nkube-system exec -it cilium-dqnsk -- cilium status
Defaulted container "cilium-agent" out of: cilium-agent, mount-cgroup (init), clean-cilium-state (init)
KVStore:                 Ok   Disabled
Kubernetes:              Ok   1.23 (v1.23.2) [linux/amd64]
Kubernetes APIs:         ["cilium/v2::CiliumClusterwideNetworkPolicy", "cilium/v2::CiliumEndpoint", "cilium/v2::CiliumNetworkPolicy", "cilium/v2::CiliumNode", "core/v1::Namespace", "core/v1::Node",
"core/v1::Pods", "core/v1::Service", "discovery/v1::EndpointSlice", "networking.k8s.io/v1::NetworkPolicy"]
KubeProxyReplacement:    Strict   [ens33 192.168.2.61 (Direct Routing)]
Host firewall:           Disabled
Cilium:                  Ok   1.11.1 (v1.11.1-76d34db)
NodeMonitor:             Disabled
Cilium health daemon:    Ok
IPAM:                    IPv4: 6/254 allocated from 10.0.1.0/24,
BandwidthManager:        Disabled
Host Routing:            BPF
Masquerading:            BPF   [ens33]   10.0.1.0/24 [IPv4: Enabled, IPv6: Disabled]
Controller Status:       39/39 healthy
Proxy Status:            OK, ip 10.0.1.159, 0 redirects active on ports 10000-20000
Hubble:                  Disabled
Encryption:              Disabled
Cluster health:          2/2 reachable   (2022-01-22T08:42:27Z)
root@bpf1:~#
```

1.Pod-Pod DIFF Node[轻量级隧道类型]

```
对于不不同节点的通信，是我们通常需要关注的重点。此例以VxLAN Backend来说明此过程。对于VxLAN实际上我们已经相对来说较为熟悉了。但是在Cilium中VxLAN稍微有些不同:
root@bpf1:~# ip -d link show cilium_vxlan
6: cilium_vxlan: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether de:41:40:10:e3:62 brd ff:ff:ff:ff:ff:ff promiscuity 0 minmtu 68 maxmtu 65535
    vxlan external addrgenmode eui64 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535      # 1.这里有一个external的字段，需要做一个说明:
root@bpf1:~#
root@bpf1:~# man ip link -h    #0.查看帮助
       VXLAN Type Support
              For a link of type VXLAN the following additional arguments are supported:

              ip link add DEVICE type vxlan id VNI [ dev PHYS_DEV  ] [ { group | remote } IPADDR ] [ local { IPADDR | any } ] [ ttl TTL ] [ tos
              TOS ] [ df DF ] [ flowlabel FLOWLABEL ] [ dstport PORT ] [ srcport MIN MAX ] [ [no]learning ] [ [no]proxy ] [ [no]rsc ] [
              [no]l2miss ] [ [no]l3miss ] [ [no]udpcsum ] [ [no]udp6zerocsumtx ] [ [no]udp6zerocsumrx ] [ ageing SECONDS ] [ maxaddress NUMBER
              ] [ [no]external ] [ gbp ] [ gpe ]

                     id VNI - specifies the VXLAN Network Identifer (or VXLAN Segment Identifier) to use.

                     dev PHYS_DEV - specifies the physical device to use for tunnel endpoint communication.

                     group IPADDR - specifies the multicast IP address to join.  This parameter cannot be specified with the remote parameter.

                     remote IPADDR - specifies the unicast destination IP address to use in outgoing packets when the destination link layer
                     address is not known in the VXLAN device forwarding database. This parameter cannot be specified with the group parame-
                     ter.

                     local IPADDR - specifies the source IP address to use in outgoing packets.

                     ttl TTL - specifies the TTL value to use in outgoing packets.

                     tos TOS - specifies the TOS value to use in outgoing packets.

                     df DF - specifies the usage of the Don't Fragment flag (DF) bit in outgoing packets with IPv4 headers. The value inherit
                     causes the bit to be copied from the original IP header. The values unset and set cause the bit to be always unset or al-
                     ways set, respectively. By default, the bit is not set.

                     flowlabel FLOWLABEL - specifies the flow label to use in outgoing packets.

                     dstport PORT - specifies the UDP destination port to communicate to the remote
                       VXLAN tunnel endpoint.

                     srcport MIN MAX - specifies the range of port numbers to use as UDP source ports to communicate to the remote VXLAN tun-
                     nel endpoint.

                     [no]learning - specifies if unknown source link layer addresses and IP addresses are entered into the VXLAN device for-
                     warding database.

                     [no]rsc - specifies if route short circuit is turned on.

                     [no]proxy - specifies ARP proxy is turned on.

                     [no]l2miss - specifies if netlink LLADDR miss notifications are generated.

                     [no]l3miss - specifies if netlink IP ADDR miss notifications are generated.

                     [no]udpcsum - specifies if UDP checksum is calculated for transmitted packets over IPv4.

                     [no]udp6zerocsumtx - skip UDP checksum calculation for transmitted packets over IPv6.

                     [no]udp6zerocsumrx - allow incoming UDP packets over IPv6 with zero checksum field.

                     ageing SECONDS - specifies the lifetime in seconds of FDB entries learnt by the kernel.

                     maxaddress NUMBER - specifies the maximum number of FDB entries.

                     [no]external - specifies whether an external control plane (e.g. ip route encap) or the internal FDB should be used.

                     gbp - enables the Group Policy extension (VXLAN-GBP).
```

```
                        Allows to transport group policy context across VXLAN network peers.  If enabled, includes the mark of a packet in
                        the VXLAN header for outgoing packets and fills the packet mark based on the information found in the VXLAN header
                        for incoming packets.

                        Format of upper 16 bits of packet mark (flags);

                          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                          |-|-|-|-|-|-|-|-|D|-|-|A|-|-|-|-|
                          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                          D := Don't Learn bit. When set, this bit indicates that the egress VTEP MUST NOT learn the source address of the
                          encapsulated frame.

                          A := Indicates that the group policy has already been applied to this packet. Policies MUST NOT be applied by de-
                          vices when the A bit is set.

                        Format of lower 16 bits of packet mark (policy ID):

                          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                          |        Group Policy ID        |
                          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                        Example:
                          iptables -A OUTPUT [...] -j MARK --set-mark 0x800FF

                    gpe - enables the Generic Protocol extension (VXLAN-GPE). Currently, this is only supported together with the external
                    keyword.
 #
                     [no]external - specifies whether an external control plane (e.g. ip route encap) or the internal FDB should be used.
                     1.这里的解释：把隧道信息封装在路由中。
 #
 这里既然说道了轻量级Tunnel，我们就多说一点：

1.首先是常规情况下的Linux kernel中VxLAN配置：
ip link add vxlan0 type vxlan id 5 dstport 4789 remote 172.12.1.12 local 172.12.1.11 dev ens33
ip addr add 10.20.1.2/24 dev vxlan0
ip link set vxlan0 up
此时并没有路由相关的配置，那么当实现VxLAN的通信的时候，需要借助于FDB表的查询实现。

2.轻量级Tunnel:
ip link add vxlan1 type vxlan dstport 4789 external
ip link set dev vxlan1 up
ip addr add 20.1.1.1/24 dev vxlan1
ip route add 10.1.1.1 encap ip id 30001 dst 20.1.1.2 dev vxlan1
此时我们可以看到，隧道信息被封装在路由中。这时一种基础的轻量级实现方式。

3.还有一种是BPF介入：
有了 BPF 可编程性之后，能为出向流量（入向是只读的）做封装。
与 tc 类似，ip route 支持直接将 BPF 程序 attach 到网络设备：
$ ip route add 192.168.253.2/32 encap bpf out obj lwt_len_hist_kern.o section len_hist dev veth0
$ ip route add <route+prefix> encap bpf out obj <bpf object file.o> section <ELF section> dev <device>

目：通常情况下：我们的VXLAN在通信的时候，在同一VNI ID的我们认为是一个Tunnel。但是在Cilium的环境中，我们会有现在（Request侧）和（Replay侧）使用的VxLAN的 VNI ID是不同的。
```

2.Pod-Pod DIFF Nodes[cilium monitor -vv]分析

Properties

```
分析不同节点Pod之间通信，对应此前我们熟悉的CNI（Calico Flannel）均是使用路由表，FDB表，ARP表等网络知识便可以分析的非常清楚，但是在Cilium中我们发现此种分析思路便"失效"了。究其原因，是由于Cilium的CNI实现结合eBPF技术实现了datapath的"跳跃式"转发。

所以我们需要结合Cilium提供的Tools来辅助分析：
我们还是由诸多分析方式来进行分析：
1.cilium monitor -vv
2.pwru
3.iptables TRACE分析
4.源码分析(我们不做重点解析，有余力可结合pwru和bpf以及kernel来分析源码)
5.tcpdump抓包分析


首先使用cilium monitor -vv来辅助分析：
env介绍：
root@bpf1:~# kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE   NOMINATED NODE   READINESS GATES
cni-6vf5d   1/1     Running   0          41h   10.0.1.180   bpf1   <none>           <none>
cni-j22kt   1/1     Running   0          41h   10.0.0.23    bpf2   <none>           <none>
root@bpf1:~#

先给出地址信息，我们可以在cilium中查询：
root@bpf1:/home/cilium# cilium bpf endpoint list
IP ADDRESS       LOCAL ENDPOINT INFO
10.0.1.180:0     id=3663  flags=0x0000 ifindex=24  mac=FE:55:FA:02:EE:E5 nodemac=D2:F1:29:4B:BD:9D     # 说明：这里实际上就是我们pod的eth0网卡和其对应的lxc网卡的信息。
root@bpf2:/home/cilium# cilium bpf endpoint list
IP ADDRESS       LOCAL ENDPOINT INFO
10.0.0.23:0      id=2720  flags=0x0000 ifindex=28  mac=42:4A:CA:D9:16:B5 nodemac=4E:51:12:28:98:B2     # 说明：这里实际上就是我们pod的eth0网卡和其对应的lxc网卡的信息。
root@bpf2:/home/cilium#

我们在bpf1节点上monitor log:
root@bpf1:~# kubectl exec -it cni-6vf5d -- ping -c 1 10.0.0.23
PING 10.0.0.23 (10.0.0.23): 56 data bytes
64 bytes from 10.0.0.23: seq=0 ttl=63 time=1.067 ms

--- 10.0.0.23 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.067/1.067/1.067 ms
root@bpf1:~#

[1.bpf1节点上的log: ]
CPU 05: MARK 0x0 FROM 3663 DEBUG: Conntrack lookup 1/2: src=10.0.1.180:15872 dst=10.0.0.23:0
CPU 05: MARK 0x0 FROM 3663 DEBUG: Conntrack lookup 2/2: nexthdr=1 flags=1
CPU 05: MARK 0x0 FROM 3663 DEBUG: CT verdict: New, revnat=0
CPU 05: MARK 0x0 FROM 3663 DEBUG: Successfully mapped addr=10.0.0.23 to identity=37483
CPU 05: MARK 0x0 FROM 3663 DEBUG: Conntrack create: proxy-port=0 revnat=0 src-identity=37483 lb=0.0.0.0
CPU 05: MARK 0x0 FROM 3663 DEBUG: Encapsulating to node 3232236094 (0xc0a8023e) from seclabel 37483
------------------------------------------------------------------------ # 1.这里分为3个部分，第一个部分CT是指的是Pod cni-6vf5d 内部数据出来的时候经过的iptables过程。
Ethernet        {Contents=[..14..] Payload=[..86..] SrcMAC=fe:55:fa:02:ee:e5 DstMAC=d2:f1:29:4b:bd:9d EthernetType=IPv4 Length=0}
IPv4    {Contents=[..20..] Payload=[..64..] Version=4 IHL=5 TOS=0 Length=84 Id=55780 Flags=DF FragOffset=0 TTL=64 Protocol=ICMPv4 Checksum=19194 SrcIP=10.0.1.180 DstIP=10.0.0.23 Options=[] Padding=[]}
ICMPv4  {Contents=[..8..] Payload=[..56..] TypeCode=EchoRequest Checksum=47137 Id=15872 Seq=0}
  Failed to decode layer: No decoder for layer type Payload
CPU 05: MARK 0x0 FROM 3663 to-overlay: 98 bytes (98 captured), state new, interface cilium_vxlan, , identity 37483->unknown, orig-ip 0.0.0.0
CPU 05: MARK 0x0 FROM 83 DEBUG: Conntrack lookup 1/2: src=192.168.2.61:36535 dst=192.168.2.62:8472
CPU 05: MARK 0x0 FROM 83 DEBUG: Conntrack lookup 2/2: nexthdr=17 flags=1
CPU 05: MARK 0x0 FROM 83 DEBUG: CT verdict: New, revnat=0
CPU 05: MARK 0x0 FROM 83 DEBUG: Conntrack create: proxy-port=0 revnat=0 src-identity=0 lb=0.0.0.0
```

```
CPU 05: MARK 0x0 FROM 83 DEBUG: Conntrack create: proxy-port=0 revnat=0 src-identity=0 lb=0.0.0.0
CPU 05: MARK 0x0 FROM 83 DEBUG: Successfully mapped addr=192.168.2.62 to identity=6
CPU 05: MARK 0x0 FROM 0 DEBUG: Tunnel decap: id=37483 flowlabel=0
CPU 05: MARK 0x0 FROM 0 DEBUG: Attempting local delivery for container id 3663 from seclabel 37483
---------------------------------------------------------------------------- # 2.这第二部分是HOST NS中iptables 处理逻辑。
这里我们需要和cilium的blog中做一下说明：https://cilium.io/static/7b77faac1700b51b5612abb7ec0c8f40/0bb32/ebpf_hostrouting.png
此图中描述的内容为：当数据包出来的时候，不再经过HOST NS中的iptables处理，只是需要HOST NS中的路由查询即可，但是我们在cilium monitor中观察到也经过HOST NS处理了。这是为何呢?
我们的host Routing的Feature也是enable的：cilium  status：Host Routing:           BPF
经过仔细查阅资料，最终发现这里我们的环境是VxLAN的环境，和Blog中的描述的Native Routing是有一定的区别的。具体就是cilium_vxlan网卡这里。
我们的数据包从Pod中出来以后，首先经过lxc网卡的from-container HOOK(tc)，可通过tc filter show dev $lxc-name ingress,具体的逻辑需要依照源码介绍这里我们先不做介绍，只是告知最终的结果是把数据包通过bpf_redirect_neigh()送到cilium_vxlan对应的接口，此过程的确不需要经过iptables的处理，此时数据包就被cilium_vxlan接收到了。接收到了以后，由于cilium_vxlan是一个vxlan device:
root@bpf1:~# ip -d link show dev cilium_vxlan
6: cilium_vxlan: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether de:41:40:10:e3:62 brd ff:ff:ff:ff:ff:ff promiscuity 0 minmtu 68 maxmtu 65535
    vxlan external addrgenmode eui64 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
root@bpf1:~#
所以通常情况下：我们会在内核的指引下进行VxLAN的数据封装了。那么：此时数据包从cilium_vxlan出去的时候需要经过cilium_vxlan的HOOK(tc):
root@bpf1:~# tc filter show dev cilium_vxlan egress
filter protocol all pref 1 bpf chain 0
filter protocol all pref 1 bpf chain 0 handle 0x1 bpf_overlay.o:[to-overlay] direct-action not_in_hw id 8479 tag dc956072f3083d63 jited
root@bpf1:~#
关于这里，实际上此HOOK只是记录一些必要信息，并没有实际处理。也就是此HOOK只是记录而已。
那么经过此以后，数据已经在HOST NS中了，由于cilium_vxlan的HOOK目前设计的是没有"实质性的BPF HOOK"，所以此时内核就会正常的对待这样的数据包，也就是说，此时的数据包不是特等公民了，那么就会被HOST NS中的iptables处理。
这也就是我们在cilium monitor中所看到封装完成的数据包中有iptables处理的过程。这点和我们Blog中描述"不同"，那么实际上由于cilium_vxlan以后的处理。
那么封装完成的VxLAN数据格式我们这里先不做展开介绍，可以暂时把其理解为一个普通的VxLAN的数据包。
那么接下里数据就会被送到我们VxLAN封装的Outer_IP接口（这里还是一样：可以指定这个Outer_IP，没有指定的情况就是default gw对应的网卡）。
而此时ens33网卡上也有其对应的HOOK(tc)。
root@bpf1:~# tc filter show dev ens33 egress
filter protocol all pref 1 bpf chain 0
filter protocol all pref 1 bpf chain 0 handle 0x1 bpf_netdev_ens33.o:[to-netdev] direct-action not_in_hw id 8597 tag 1470e4d1d2faa158 jited
root@bpf1:~#
这里实际上也没有做太多处理，在egress的方向上。注意这里仅仅是egress的方向的处理。[实际上，此HOOK主要是处理N-S的流量]，我们这里的是[E-W]的流量。所以，数据就被送到了对端Node的ens33的接口上。
此时我们再引用一个图：https://github.com/BurlyLuo/train/blob/main/Cilium/cilium%20datapath.png
这里看到，当对端bpf2上的ens33收到这个数据报文以后，此时会redirect到cilium_vxlan上，由于我的内接是5.11.所以不再经过HOST NS中的iptables。这里我们一会看对端流程。这里先看后边的内容。我们下边又看到了iptables处理的逻辑，但是请注意的是，此时Src_IP 和 Dst_IP发生变化，说明什么？说明此时该项记录是回来的ICMP的Replay，这样理解起来就没有什么难度了。
----------------------------------------------------------------------------
CPU 05: MARK 0x0 FROM 3663 DEBUG: Conntrack lookup 1/2: src=10.0.0.23:0 dst=10.0.1.180:15872
CPU 05: MARK 0x0 FROM 3663 DEBUG: Conntrack lookup 2/2: nexthdr=1 flags=0
CPU 05: MARK 0x0 FROM 3663 DEBUG: CT entry found lifetime=16940931, revnat=0
CPU 05: MARK 0x0 FROM 3663 DEBUG: CT verdict: Reply, revnat=0
----------------------------------------------------------------------------
Ethernet        {Contents=[..14..] Payload=[..86..] SrcMAC=d2:f1:29:4b:bd:9d DstMAC=fe:55:fa:02:ee:e5 EthernetType=IPv4 Length=0}
IPv4    {Contents=[..20..] Payload=[..64..] Version=4 IHL=5 TOS=0 Length=84 Id=56500 Flags= FragOffset=0 TTL=63 Protocol=ICMPv4 Checksum=35114 SrcIP=10.0.0.23 DstIP=10.0.1.180 Options=[] Padding=[]}
ICMPv4  {Contents=[..8..] Payload=[..56..] TypeCode=EchoReply Checksum=49185 Id=15872 Seq=0}
  Failed to decode layer: No decoder for layer type Payload
CPU 05: MARK 0x0 FROM 3663 to-endpoint: 98 bytes (98 captured), state reply, interface lxcf9d0a91de5fb, , identity 37483->37483, orig-ip 10.0.0.23, to endpoint 3663
此以上是数据bpf1 节点Egress的数据分析。下边我们分析到达对端bpf2的datapath。

*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*

CPU 05: MARK 0x0 FROM 0 DEBUG: Tunnel decap: id=37483 flowlabel=0    # 1.这里看到数据被decap，此时便发到cilium_vxlan接口（我们通过在cilium_vxlan上抓包，便可得到此结论）
CPU 05: MARK 0x0 FROM 0 DEBUG: Attempting local delivery for container id 2720 from seclabel 37483
CPU 05: MARK 0x0 FROM 2720 DEBUG: Conntrack lookup 1/2: src=10.0.1.180:15872 dst=10.0.0.23:0          # 1.此时数据从cilium_vxlan送到bpf2上的pod中的eth0网卡.(此时也是通过tcpdump抓包可得)
CPU 05: MARK 0x0 FROM 2720 DEBUG: Conntrack lookup 2/2: nexthdr=1 flags=0
CPU 05: MARK 0x0 FROM 2720 DEBUG: CT verdict: New, revnat=0
CPU 05: MARK 0x0 FROM 2720 DEBUG: Conntrack create: proxy-port=0 revnat=0 src-identity=37483 lb=0.0.0.0
----------------------------------------------------------------------------
Ethernet        {Contents=[..14..] Payload=[..86..] SrcMAC=4e:51:12:28:98:b2 DstMAC=42:4a:ca:d9:16:b5 EthernetType=IPv4 Length=0}
IPv4    {Contents=[..20..] Payload=[..64..] Version=4 IHL=5 TOS=0 Length=84 Id=55780 Flags=DF FragOffset=0 TTL=63 Protocol=ICMPv4 Checksum=19450 SrcIP=10.0.1.180 DstIP=10.0.0.23 Options=[] Padding=[]}
ICMPv4  {Contents=[..8..] Payload=[..56..] TypeCode=EchoRequest Checksum=47137 Id=15872 Seq=0}
  Failed to decode layer: No decoder for layer type Payload

:3.这里我们又看到数据包被iptables处理了。但是此时Src_IP 和 Dst_IP发生变化了，也就是此时是一个ICMP的Replay，然后还是送到cilium_vxlan的接口。

CPU 05: MARK 0x0 FROM 2720 to-endpoint: 98 bytes (98 captured), state new, interface lxc806b0ea36f81, , identity 37483->37483, orig-ip 10.0.1.180, to endpoint 2720
CPU 05: MARK 0x0 FROM 2720 DEBUG: Conntrack lookup 1/2: src=10.0.0.23:0 dst=10.0.1.180:15872
CPU 05: MARK 0x0 FROM 2720 DEBUG: Conntrack lookup 2/2: nexthdr=1 flags=1
CPU 05: MARK 0x0 FROM 2720 DEBUG: CT entry found lifetime=16940928, revnat=0
CPU 05: MARK 0x0 FROM 2720 DEBUG: CT verdict: Reply, revnat=0
CPU 05: MARK 0x0 FROM 2720 DEBUG: Successfully mapped addr=10.0.1.180 to identity=37483
CPU 05: MARK 0x0 FROM 2720 DEBUG: Encapsulating to node 3232236093 (0xc0a8023d) from seclabel 37483
----------------------------------------------------------------------------
Ethernet        {Contents=[..14..] Payload=[..86..] SrcMAC=42:4a:ca:d9:16:b5 DstMAC=4e:51:12:28:98:b2 EthernetType=IPv4 Length=0}
IPv4    {Contents=[..20..] Payload=[..64..] Version=4 IHL=5 TOS=0 Length=84 Id=56500 Flags= FragOffset=0 TTL=64 Protocol=ICMPv4 Checksum=34858 SrcIP=10.0.0.23 DstIP=10.0.1.180 Options=[] Padding=[]}
ICMPv4  {Contents=[..8..] Payload=[..56..] TypeCode=EchoReply Checksum=49185 Id=15872 Seq=0}
  Failed to decode layer: No decoder for layer type Payload
CPU 05: MARK 0x0 FROM 2720 to-overlay: 98 bytes (98 captured), state new, interface cilium_vxlan, , identity 37483->unknown, orig-ip 0.0.0.0
CPU 05: MARK 0x0 FROM 415 DEBUG: Conntrack lookup 1/2: src=192.168.2.62:47380 dst=192.168.2.61:8472
CPU 05: MARK 0x0 FROM 415 DEBUG: Conntrack lookup 2/2: nexthdr=17 flags=1                              # 4.此时就是被封装完成的VxLAN报文格式，所以此时就又是HOST NS的iptables处理，和路由查询。
CPU 05: MARK 0x0 FROM 415 DEBUG: CT verdict: New, revnat=0
CPU 05: MARK 0x0 FROM 415 DEBUG: Conntrack create: proxy-port=0 revnat=0 src-identity=0 lb=0.0.0.0


*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*
综述：首先：环境的kernel是：5.11且打开了cilium的host Routing的Feature.所以具有bpf_redirect_peer() 和 bpf_redirect_neigh()的helper函数。
所以我们需要去仔细研读这个：https://github.com/BurlyLuo/train/blob/main/Cilium/cilium%20datapath.png  这个datapath。

通过cilium monitor -vv 分析得到的datapath，通常可以辅助我们理解数据包的转发，这也是观测cilium的一个得力的工具。
```

3.tcpdump抓包分析

```
root@bpf1:~# kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE   NOMINATED NODE   READINESS GATES
cni-6vf5d   1/1     Running   0          47h   10.0.1.180   bpf1   <none>           <none>
cni-j22kt   1/1     Running   0          47h   10.0.0.23    bpf2   <none>           <none>
root@bpf1:~#
先给出地址信息，我们可以在cilium中查询:
root@bpf1:/home/cilium# cilium bpf endpoint list
IP ADDRESS       LOCAL ENDPOINT INFO
10.0.1.180:0     id=3663  flags=0x0000 ifindex=24  mac=FE:55:FA:02:EE:E5 nodemac=D2:F1:29:4B:BD:9D     # 说明：这里实际上就是我们bpf1上pod的eth0网卡和其对应的lxc网卡的信息。
root@bpf2:/home/cilium# cilium bpf endpoint list
IP ADDRESS       LOCAL ENDPOINT INFO
10.0.0.23:0      id=2720  flags=0x0000 ifindex=28  mac=42:4A:CA:D9:16:B5 nodemac=4E:51:12:28:98:B2     # 说明：这里实际上就是我们bpf2上pod的eth0网卡和其对应的lxc网卡的信息。

具体抓包过程：[cni-6vf5d]pod数据egress的datapath:
[1.pod中的eth0网卡]
root@bpf1:~# kubectl exec -it cni-6vf5d -- tcpdump -pne -i eth0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:12:50.047213 fe:55:fa:02:ee:e5 > d2:f1:29:4b:bd:9d, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 19200, seq 0, length 64  # 1.此时在Pod中抓包，出去的ICMP Request。
13:12:50.047930 d2:f1:29:4b:bd:9d > fe:55:fa:02:ee:e5, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 19200, seq 0, length 64
^C
2 packets captured
```

```
2 packets received by filter
0 packets dropped by kernel
root@bpf1:~#
[2.Pod对端的lxc网卡]
24: lxcf9d0a91de5fb@if23: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether d2:f1:29:4b:bd:9d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::d0f1:29ff:fe4b:bd9d/64 scope link
       valid_lft forever preferred_lft forever
root@bpf1:~# tcpdump -pne -i lxcf9d0a91de5fb
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lxcf9d0a91de5fb, link-type EN10MB (Ethernet), capture size 262144 bytes
13:12:50.047218 fe:55:fa:02:ee:e5 > d2:f1:29:4b:bd:9d, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 19200, seq 0, length 64   # 2.此时看到icmp request.没有icmp replay。
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@bpf1:~#
[3.cilium_vxlan的网卡]:
root@bpf1:~# tcpdump -pne -i cilium_vxlan
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on cilium_vxlan, link-type EN10MB (Ethernet), capture size 262144 bytes
13:12:50.047247 fe:55:fa:02:ee:e5 > d2:f1:29:4b:bd:9d, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 19200, seq 0, length 64
13:12:50.047930 42:4a:ca:d9:16:b5 > 4e:51:12:28:98:b2, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 19200, seq 0, length 64
^C
18 packets captured
18 packets received by filter
0 packets dropped by kernel
root@bpf1:~#
[4.VxlAN的Outer_IP接口抓包]:
root@bpf1:~# tcpdump -l -pen -T vxlan -i ens33 port 8472
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
13:28:18.510940 00:0c:29:67:92:63 > 00:0c:29:1f:10:5f, ethertype IPv4 (0x0800), length 148: 192.168.2.61.36535 > 192.168.2.62.8472: VXLAN, flags [I] (0x08), vni 37483 # 3.VNI的地址: 37483
fe:55:fa:02:ee:e5 > d2:f1:29:4b:bd:9d, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 29184, seq 0, length 64
13:28:18.511342 00:0c:29:1f:10:5f > 00:0c:29:67:92:63, ethertype IPv4 (0x0800), length 148: 192.168.2.62.47380 > 192.168.2.61.8472: VXLAN, flags [I] (0x08), vni 37483
42:4a:ca:d9:16:b5 > 4e:51:12:28:98:b2, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 29184, seq 0, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@bpf1:~#

*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#

此时我们分析数据包到bpf2 node上的情况:
[1.ens33的抓包]
root@bpf2:~# tcpdump -pne -T vxlan  port 8472 -i ens33
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
13:29:50.622608 00:0c:29:67:92:63 > 00:0c:29:1f:10:5f, ethertype IPv4 (0x0800), length 148: 192.168.2.61.36535 > 192.168.2.62.8472: VXLAN, flags [I] (0x08), vni 37483
fe:55:fa:02:ee:e5 > d2:f1:29:4b:bd:9d, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 30976, seq 0, length 64
13:29:50.622889 00:0c:29:1f:10:5f > 00:0c:29:67:92:63, ethertype IPv4 (0x0800), length 148: 192.168.2.62.47380 > 192.168.2.61.8472: VXLAN, flags [I] (0x08), vni 37483
42:4a:ca:d9:16:b5 > 4e:51:12:28:98:b2, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 30976, seq 0, length 64
^C
2 packets captured
2 packets received by filter
0 packets dropped by kernel
root@bpf2:~#

[2.bpf2 节点抓包]:
root@bpf2:~# tcpdump -pne -i cilium_vxlan
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on cilium_vxlan, link-type EN10MB (Ethernet), capture size 262144 bytes
13:12:50.047345 fe:55:fa:02:ee:e5 > d2:f1:29:4b:bd:9d, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 19200, seq 0, length 64
13:12:50.047619 42:4a:ca:d9:16:b5 > 4e:51:12:28:98:b2, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 19200, seq 0, length 64
^C
24 packets captured
24 packets received by filter
0 packets dropped by kernel
root@bpf2:~#

[3.bpf2 上pod的lxc网卡抓包: ]
28: lxc806b0ea36f81@if27: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 4e:51:12:28:98:b2 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::4c51:12ff:fe28:98b2/64 scope link
       valid_lft forever preferred_lft forever
root@bpf2:~# tcpdump -pne -i lxc806b0ea36f81
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lxc806b0ea36f81, link-type EN10MB (Ethernet), capture size 262144 bytes
13:12:50.047496 42:4a:ca:d9:16:b5 > 4e:51:12:28:98:b2, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 19200, seq 0, length 64  4.此时只有ICMP的Replay消息。
^C
3 packets captured
3 packets received by filter
0 packets dropped by kernel
root@bpf2:~#

[4.bpf2 节点上的pod中的eth0网卡]:
root@bpf2:~# kubectl exec -it cni-j22kt -- tcpdump -pne -i eth0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:12:50.047345 4e:51:12:28:98:b2 > 42:4a:ca:d9:16:b5, ethertype IPv4 (0x0800), length 98: 10.0.1.180 > 10.0.0.23: ICMP echo request, id 19200, seq 0, length 64
13:12:50.047494 42:4a:ca:d9:16:b5 > 4e:51:12:28:98:b2, ethertype IPv4 (0x0800), length 98: 10.0.0.23 > 10.0.1.180: ICMP echo reply, id 19200, seq 0, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
root@bpf2:~#



*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#
通过以上的分析我们看到此时两边的VNI的ID都是37843,这个可能是一个巧合: 因为这个VNI实际上是IDENTITY的ID:
解析这个包所属的 identity (Cilium 依赖 identity 做安全策略) , 并存储到包的结构体中。
   对于 direct routing 模式, 从 ipcache 中根据 IP 查询 identity。
   对于 tunnel 模式, 直接从 VxLAN 头中携带过来了

所以:
[bpf1 node# cilium endpoint list]
root@bpf1:/home/cilium# cilium endpoint list
ENDPOINT   POLICY (ingress)   POLICY (egress)   IDENTITY   LABELS (source:key[=value])                                      IPv6   IPv4         STATUS
           ENFORCEMENT        ENFORCEMENT
83         Disabled           Disabled          1          k8s:node-role.kubernetes.io/control-plane                                                   ready
                                                           k8s:node-role.kubernetes.io/master
                                                           k8s:node.kubernetes.io/exclude-from-external-load-balancers
```

```
                                                  reserved:host
1429       Disabled           Disabled          4          reserved:health                                                  10.0.1.244   ready
1690       Disabled           Disabled          44045      k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=kube-system       10.0.1.71    ready
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=coredns
                                                           k8s:io.kubernetes.pod.namespace=kube-system
                                                           k8s:k8s-app=kube-dns
2447       Disabled           Disabled          44045      k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=kube-system       10.0.1.93    ready
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=coredns
                                                           k8s:io.kubernetes.pod.namespace=kube-system
                                                           k8s:k8s-app=kube-dns
3663       Disabled           Disabled          37483      k8s:app=cni                                                      10.0.1.180   ready
                                                           k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=default
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=default
                                                           k8s:io.kubernetes.pod.namespace=default
root@bpf1:/home/cilium#

[bpf2 node# cilium endpoint list]
root@bpf2:/home/cilium# cilium endpoint list
ENDPOINT   POLICY (ingress)   POLICY (egress)   IDENTITY   LABELS (source:key[=value])                                      IPv6   IPv4         STATUS
           ENFORCEMENT        ENFORCEMENT
415        Disabled           Disabled          1          reserved:host                                                                 ready
2720       Disabled           Disabled          37483      k8s:app=cni                                                             10.0.0.23    ready
                                                           k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=default
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=default
                                                           k8s:io.kubernetes.pod.namespace=default
3590       Disabled           Disabled          4          reserved:health                                                         10.0.0.148   ready
root@bpf2:/home/cilium#
```

train/cilium datapath.png at main · BurlyLuo/train

Contribute to BurlyLuo/train development by creating an account on GitHub.

GitHub

BurlyLuo/train

```
root@bpf1:/home/cilium# cilium endpoint list
ENDPOINT   POLICY (ingress)   POLICY (egress)   IDENTITY   LABELS (source:key[=value])                                      IPv6   IPv4         STATUS
           ENFORCEMENT        ENFORCEMENT
83         Disabled           Disabled          1          k8s:node-role.kubernetes.io/control-plane                                     ready
                                                           k8s:node-role.kubernetes.io/master
                                                           k8s:node.kubernetes.io/exclude-from-external-load-balancers
                                                           reserved:host
1429       Disabled           Disabled          4          reserved:health                                                         10.0.1.244   ready
1690       Disabled           Disabled          44045      k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=kube-system              10.0.1.71    ready
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=coredns
                                                           k8s:io.kubernetes.pod.namespace=kube-system
                                                           k8s:k8s-app=kube-dns
2447       Disabled           Disabled          44045      k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=kube-system              10.0.1.93    ready
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=coredns
                                                           k8s:io.kubernetes.pod.namespace=kube-system
                                                           k8s:k8s-app=kube-dns
3663       Disabled           Disabled          37483      k8s:app=cni                                                             10.0.1.180   ready
                                                           k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=default
                                                           k8s:io.cilium.k8s.policy.cluster=default
                                                           k8s:io.cilium.k8s.policy.serviceaccount=default
                                                           k8s:io.kubernetes.pod.namespace=default
```