Preserve Original Source Address within Istio

Zhonghu Xu @hzxuzhonghu

About me

Zhonghu Xu：an open source engineer from Huawei Cloud.

- Github：https://github.com/hzxuzhonghu
- Istio steering committee member
- Istio Core Maintainer & Contributor
- Open source enthusiastic, previously Kubernetes active contributor and Volcano maintainer

Agenda

1. Background
2. TCP Original Address Preserve
3. HTTP Original Address Preserve
4. Demo

Content

What is the use case of original address

1. Sticky Session: based on ip hash, traffic from same client is forwarded to the same backend
2. Security Policy: set white/black list
3. Access log & Stats
4. Specific scenarios like SIP Trunking

Common Ways to Preserve Original Src Addr

- **L3**
  - LVS, one connection
  - HAProxy transparent mode, two connections
- **L4**
  - Add IP in TCP Protocol options
  - Proxy Protocol
- **L7**
  - HTTP header “x-forwarded-for”
  - User Protocol

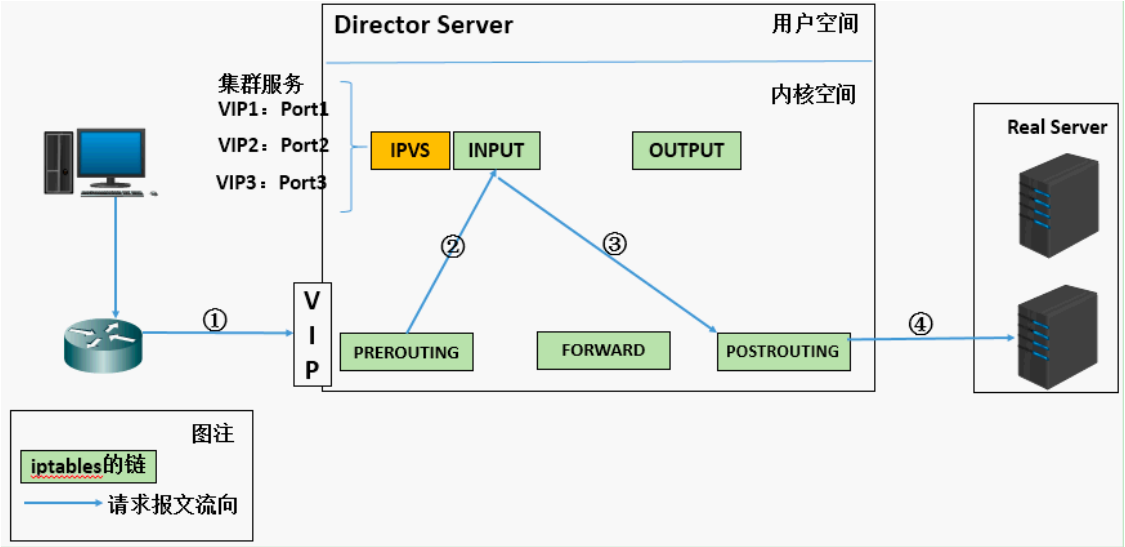
LVS

① user send traffic to LVS

② PREROUTING chain intercept packet and send it to INPUT

③ LVS work on INPUT, modify the packet dest ip + port and forward it to POSTROUTING

④ send out to real server



Note: **Only one connection between user and real server**

HAPROXY- Transparent Transport

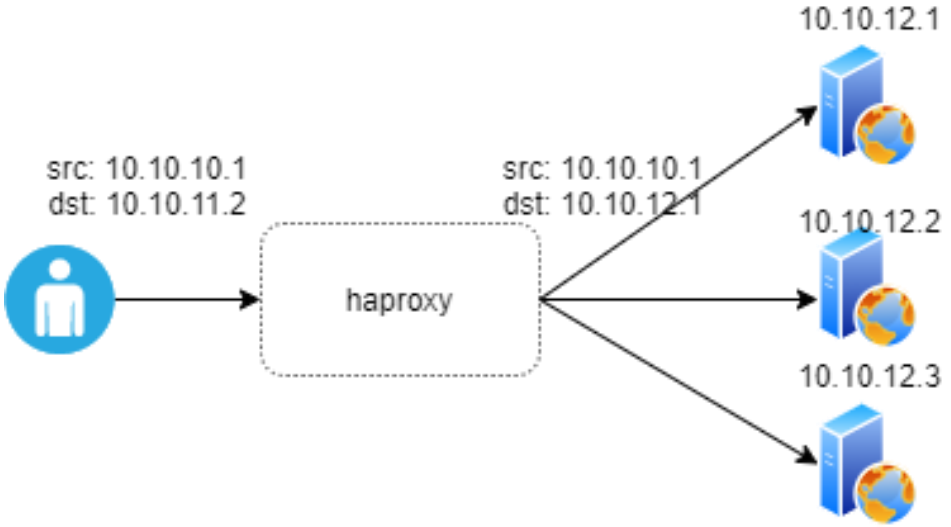
① user send traffic to haproxy

② HAPROXY works on userspace

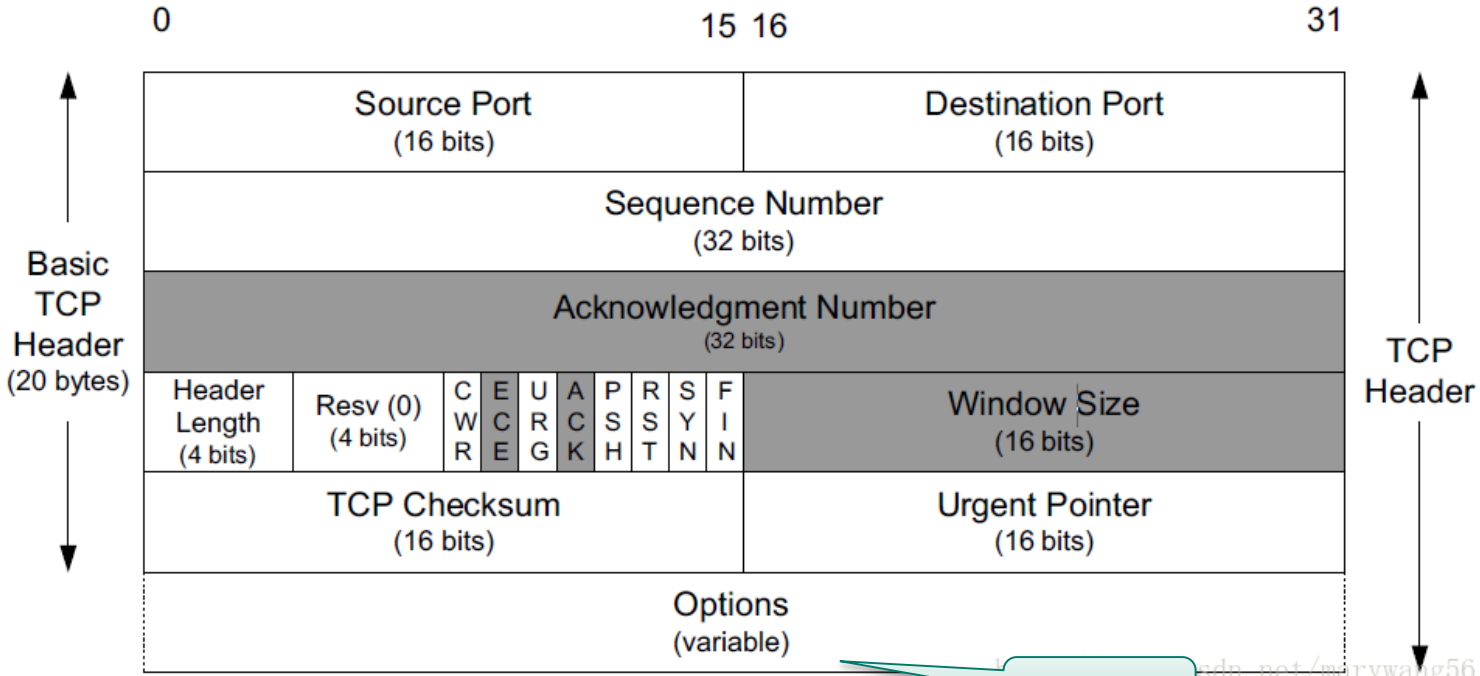
③ Listen on vip + port and accept user connection

④ Loadbalancing: select a endpoint and init a connection to server with original user's address (**IP_TRANSPARENT**)

⑤ Server's response packet is flowing through the same path (**TPROXY + Custom Route**)



TOA



Caveats : **install toa module in kernel**

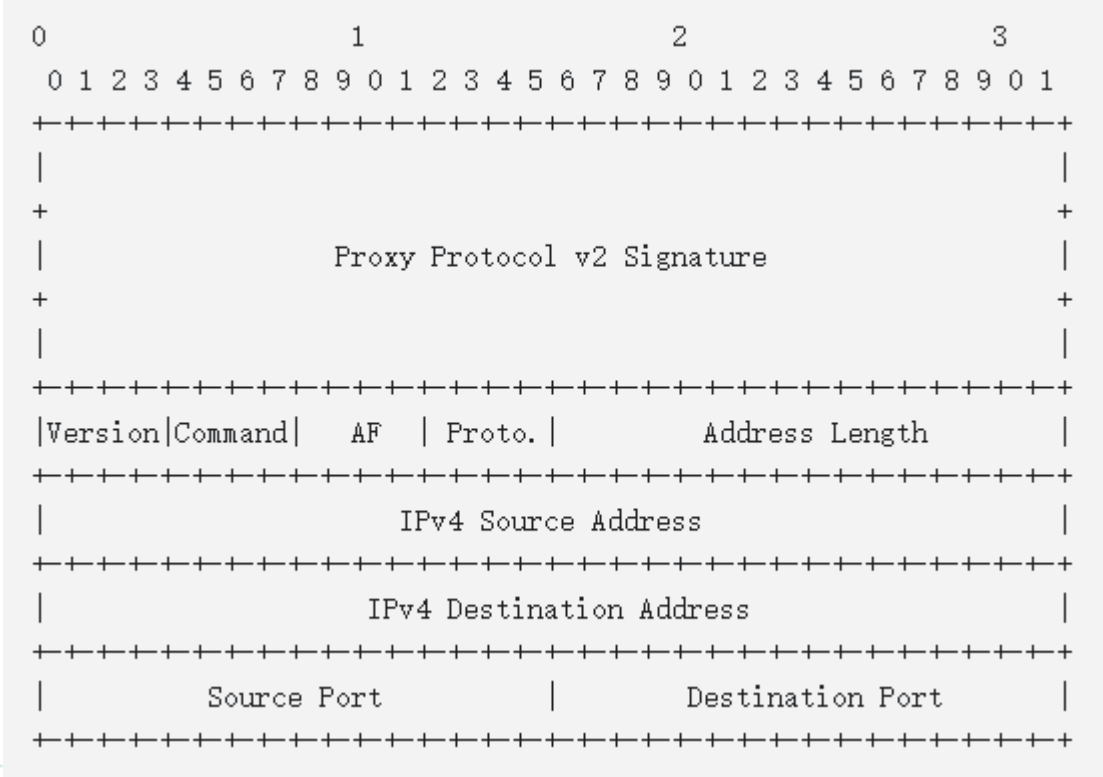
Proxy Protocol

- **Proxy Protocol v1**

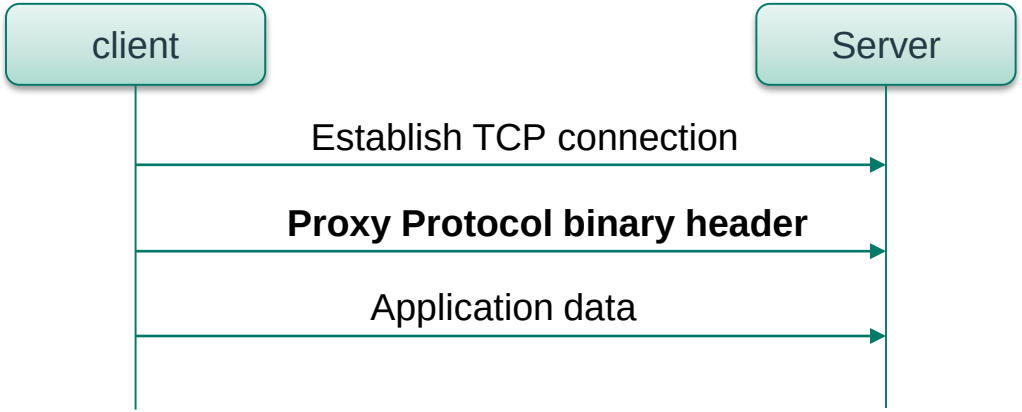
PROXY Protocol prepends every connection with a header reporting the client IP address and port. A PROXY Protocol plain-text header has the format:

PROXY TCP4 192.0.2.0 192.0.2.255 42300 443\r\n

- **Proxy Protocol v2**



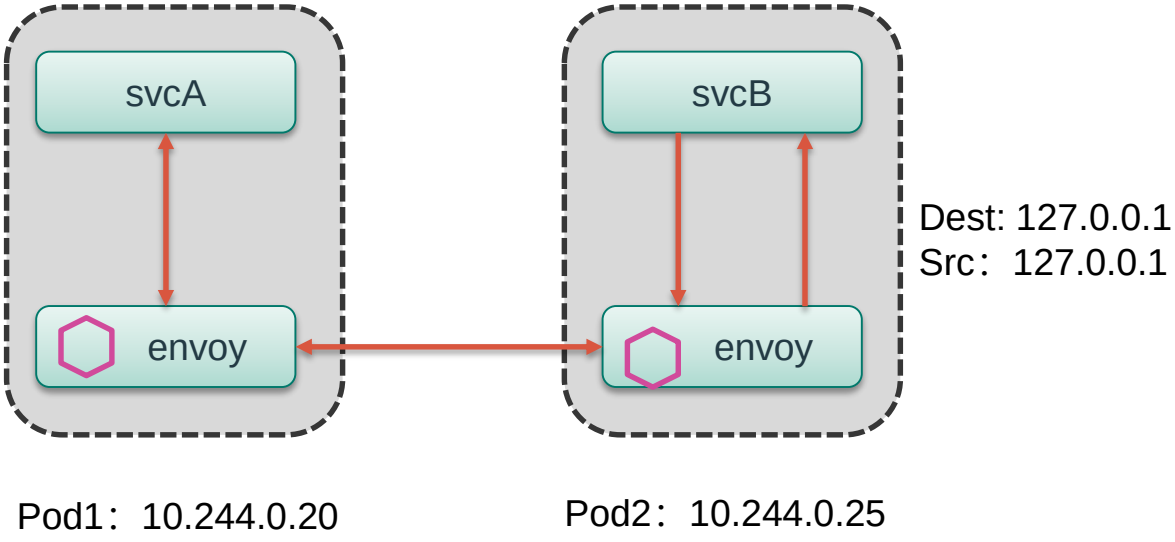
Proxy Protocol



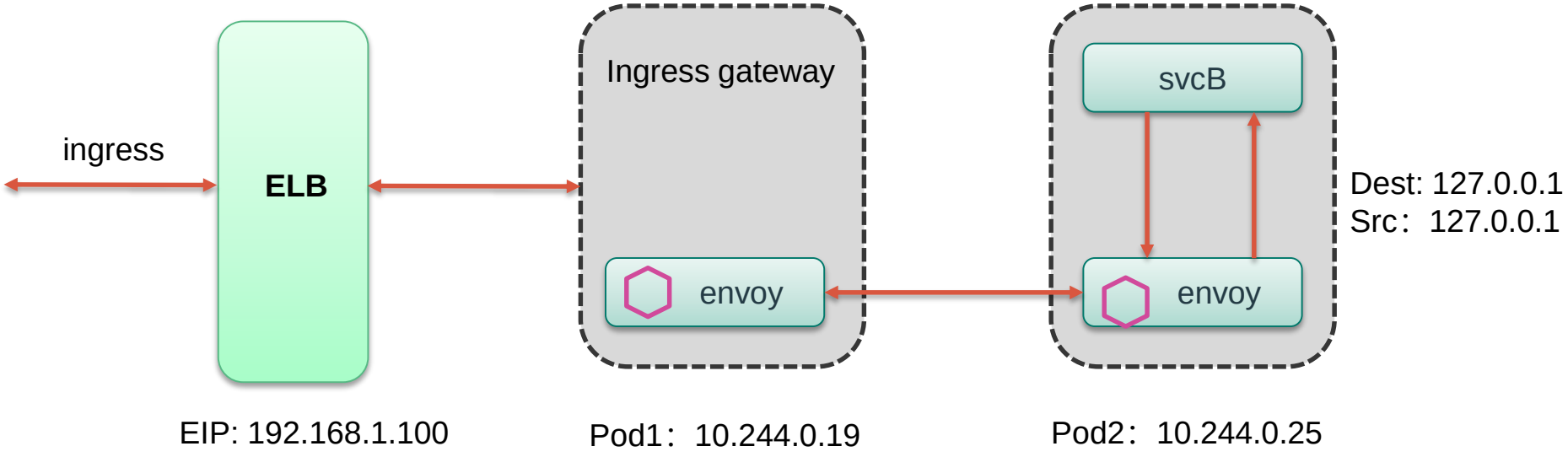
- The client and server side must support proxy protocol simultaneously

- The client here can be load balancers like envoy/haproxy/nginx which have already supported proxy protocol

Istio Traffic Flow – inner cluster



Istio Traffic Flow - ingress



What does envoy provide?

- **Original source filter “envoy.filters.listener.original_src”**

The original source listener filter replicates the downstream remote address of the connection on the upstream side of Envoy. For example, if a downstream connection connects to Envoy with IP address 10.1.2.3, then Envoy will connect to the upstream with source IP 10.1.2.3.

- **Proxy Protocol filter “*envoy.filters.listener.proxy_protocol*”:**

This listener filter adds support for HAProxy Proxy Protocol.

This implementation supports both version 1 and version 2, it automatically determines on a per-connection basis which of the two versions is present.

- **Proxy Protocol Transport Socket**

HTTP XFF

x-forwarded-for (XFF) is a standard proxy header which indicates the IP addresses that a request has flowed through on its way from the client to the server.

Envoy can append the ip address of the nearest client to the XFF

HttpConnectionManager configuration

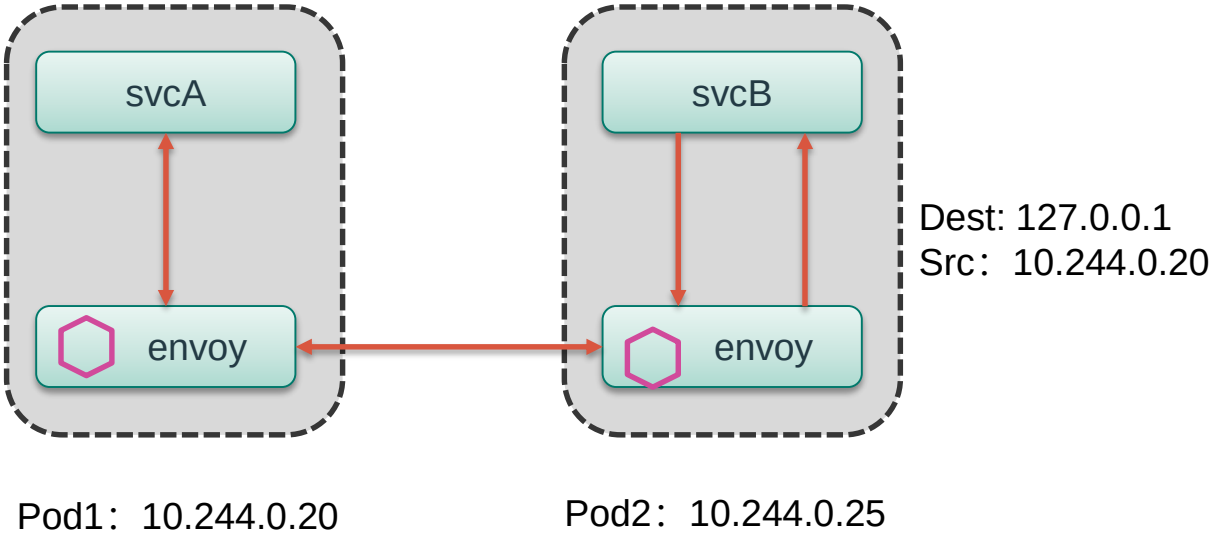
use_remote_address: Envoy will only append to XFF if the use_remote_address HTTP connection manager option is set to true and the skip_xff_append is set false.

xff_num_trusted_hops : If *use_remote_address* is true and *xff_num_trusted_hops* is set to a value *N* that is greater than zero, the trusted client address is the Nth address from the right end of XFF.

Content

1. Background
2. TCP Original Address Preserve
3. HTTP Original Address Preserve
4. Demo

Preserve TCP Original Src Addr - inner



① Setting annotation sidecar.istio.io/interceptionMode: TPROXY, istio will automatically set the original src filter and iptabels rules

Preserve TCP Original Src Addr - inner

① Config original src filter: IP_TRANSPARENT and mark upstream packets to 1337

② Make the response packet redirected back to envoy

-A PREROUTING -p tcp -m mark --mark 0x539 -j CONNMARK --save-mark --nfmask 0xffffffff --ctmask 0xffffffff # mark connection 1337 according to packet sent to application

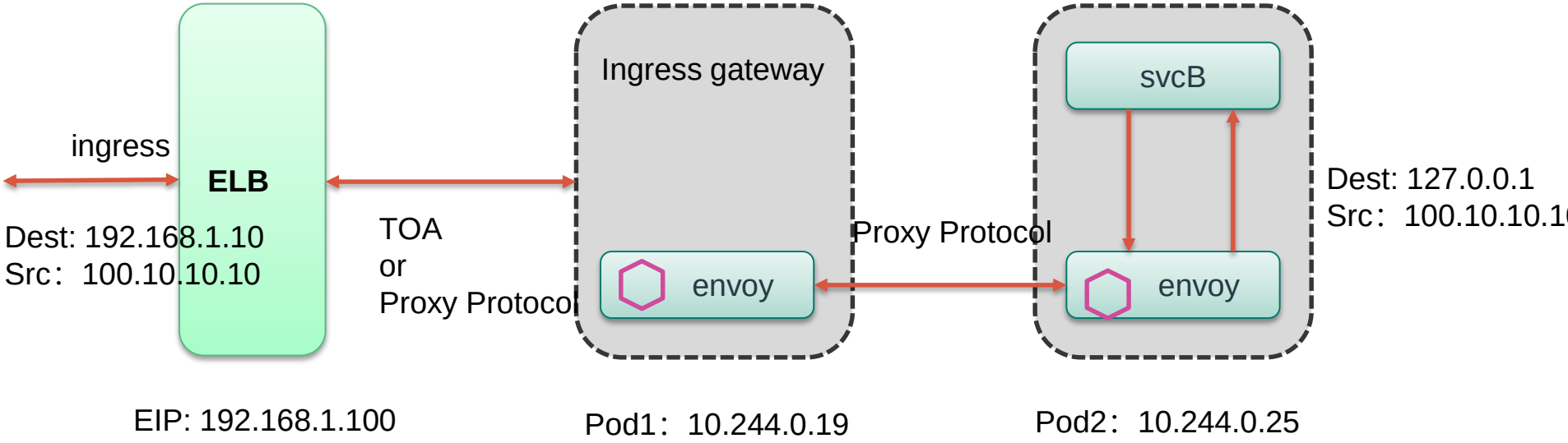
-A OUTPUT -p tcp -m connmark --mark 0x539 -j CONNMARK --restore-mark --nfmask 0xffffffff --ctmask 0xffffffff # packet sent back to envoy will be marked 1337

ip -f inet rule add fwmark 1337 lookup 133

ip -f inet route add local default dev lo table 133

③ echo 1 > /proc/sys/net/ipv4/conf/eth0/route_localnet

Preserve TCP Original Src Addr - ingress



Preserve TCP Original Src Addr - ingress

- If ingress traffic is using TOA

  ① Ingress gateway only need enable Proxy Protocol Transport Socket.

```
clusters:
- name: service1
  connect_timeout: 0.25s
  type: strict_dns
  lb_policy: round_robin
  transport_socket:
    name: envoy.transport_sockets.upstream_proxy_protocol
    typed_config:
      "@type": type.googleapis.com/envoy.extensions.transport_sockets.proxy_protocol.v3.ProxyProtocolUpstreamTransport
      config:
        version: V1
      transport_socket:
        name: envoy.transport_sockets.raw_buffer
  ...
```

Preserve TCP Original Src Addr - ingress

- If ingress traffic is using proxy protocol

  ① Ingress gateway should set “envoy.filters.listener.proxy_protocol".

```
listeners:
- address:
    socket_address:
      address: 0.0.0.0
      port_value: 443
  listener_filters:
  - name: "envoy.filters.listener.tls_inspector"
    typed_config: {}
  # Uncomment if Envoy is behind a load balancer that exposes client IP address using the PROXY protocol.
  # - name: envoy.filters.listener.proxy_protocol
  #   typed_config:
  #     "@type": type.googleapis.com/envoy.extensions.filters.listener.proxy_protocol.v3.ProxyProtocol
```

  ② enable Proxy Protocol Transport Socket in upstream cluster.

Preserve TCP Original Src Addr - ingress

- svcB

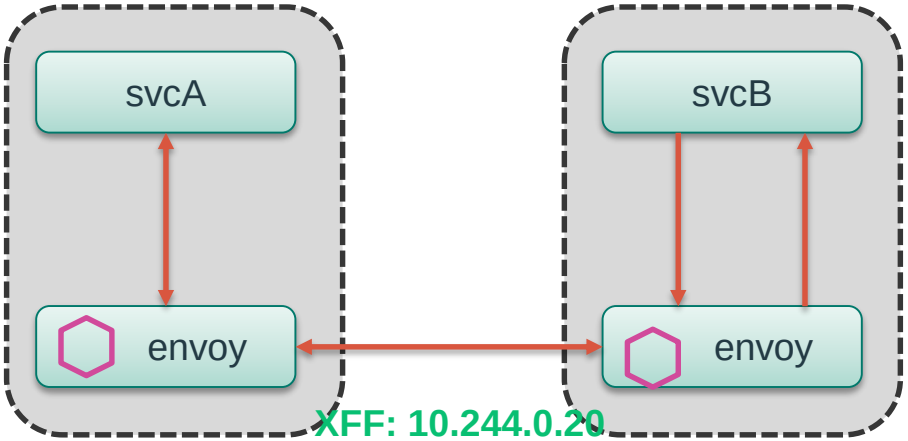
  ① Set “envoy.filters.listener.proxy_protocol“ in inbound listener.

  ② Setting annotation sidecar.istio.io/interceptionMode: TPROXY， this will set all the rules as inner cluster

Content

1. Background
2. TCP Original Address Preserve
3. HTTP Original Address Preserve
4. Demo

Preserve HTTP Original Src Addr - inner



Dest: 127.0.0.1
Src： 127.0.0.1

XFF: 10.244.0.20

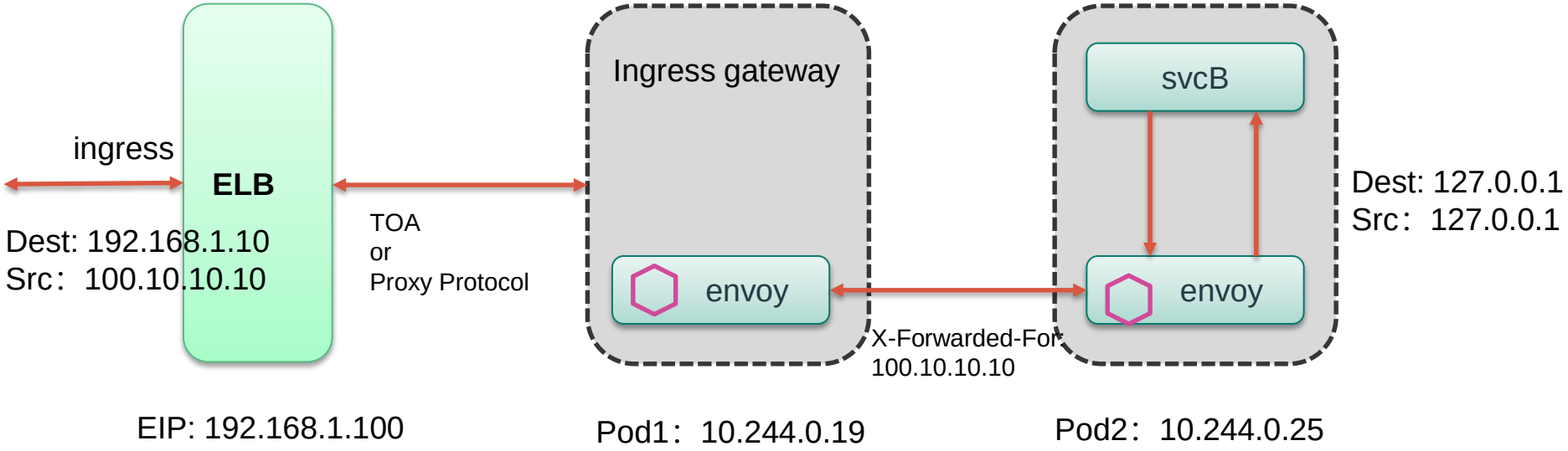
Pod1：10.244.0.20

Pod2：10.244.0.25

① Enable X-Forwarded-For HTTP header in svcA

```
name: envoy.http_connection_manager
typed_config:
  "@type": type.googleapis.com/envoy.config.filter.network.http_connection_manager.v2.HttpConnectionManager
  skip_xff_append: false
  use_remote_address: true
  xff_num_trusted_hops: 1
```

Preserve HTTP Original Src Addr - ingress



Preserve HTTP Original Src Addr - ingress

- If ingress traffic is using TOA, Ingress gateway only need enable X-Forwarded-For header.

```
- applyTo: NETWORK_FILTER
  match:
    listener:
      filterChain:
        filter:
          name: envoy.http_connection_manager
  patch:
    operation: MERGE
    value:
      name: envoy.http_connection_manager
      typed_config:
        "@type": type.googleapis.com/envoy.config.filter.network.http_connection_manager.v2.HttpConnectionManager
        skip_xff_append: false
        use_remote_address: true
        xff_num_trusted_hops: 1
```

Preserve HTTP Original Src Addr - ingress

- If ingress traffic is using proxy protocol

① Ingress gateway should set “envoy.filters.listener.proxy_protocol".

```
listeners:
- address:
    socket_address:
      address: 0.0.0.0
      port_value: 443
  listener_filters:
  - name: "envoy.filters.listener.tls_inspector"
    typed_config: {}
  # Uncomment if Envoy is behind a load balancer that exposes client IP address using the PROXY protocol.
  # - name: envoy.filters.listener.proxy_protocol
  #   typed_config:
  #     "@type": type.googleapis.com/envoy.extensions.filters.listener.proxy_protocol.v3.ProxyProtocol
```

② enable X-Forwarded-For header.

Demo

① Inner cluster HTTP traffic

② External HTTP traffic

③ Inner Cluster TCP traffic

④ External TCP (not supported well)

Thank you!

@hzxuzhonghu
https://github.com/hzxuzhonghu