Use eBPF instead of iptables to accelerate the Istio dataplane

刘齐均 / kebe7jun / DaoCloud

IstioCon

#IstioCon

Welcome to the IstioCon 2022

IstioCon 2022 is the inaugural community conference for the industry's most popular service mesh. IstioCon is a community-led event, showcasing the lessons learned from running Istio in production, hands-on experiences from the Istio community, and featuring maintainers from across the Istio ecosystem. The conference offers a mix of keynotes, technical talks, lightning talks, workshops and roadmap sessions. Fun and games are also included with two social hours to take the load off and mesh with the Istio community, vendors, and maintainers!

Agenda

- Idea
- Motivation
- Approaches
    - How to do traffic interception with eBPF?
    - How to shorten the data-path?
    - How to resolve quadruplet conflicts?
    - How to get the current Pod IP in eBPF?
- Live demo
- RoadMap

Idea

1. There has been some voices in the community about the possibility of using eBPF instead of iptables.

2. There is no project in the open source community that do this well.

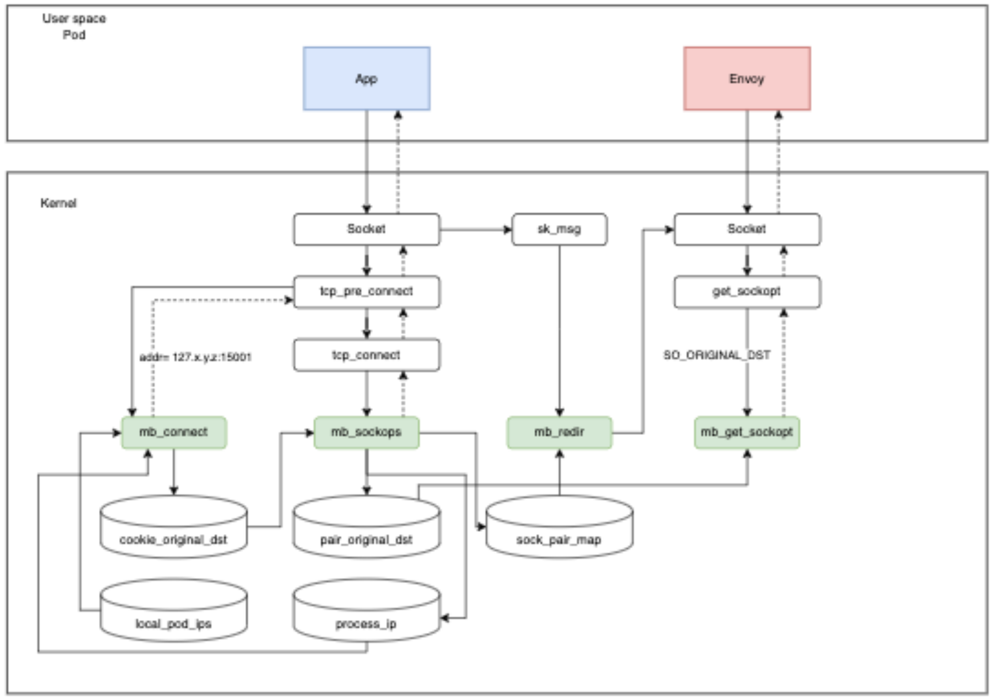
What is our goal?

1. Use eBPF instead of iptables to accelerate the Istio data-plane.
2. Zero Istio modification.
3. Quick and easy way to use for as many users as possible.

How to do it?

How to do traffic interception with eBPF?

- Which kinds of traffic？
  - Sent from app containers(recognized by UID)
- Where to send traffic？
- How？
  - Revise the original destination address to 127.0.0.1:15001 through `connect` program
- How to obtain original destination address？
  - Returned by `get_sockopt` function

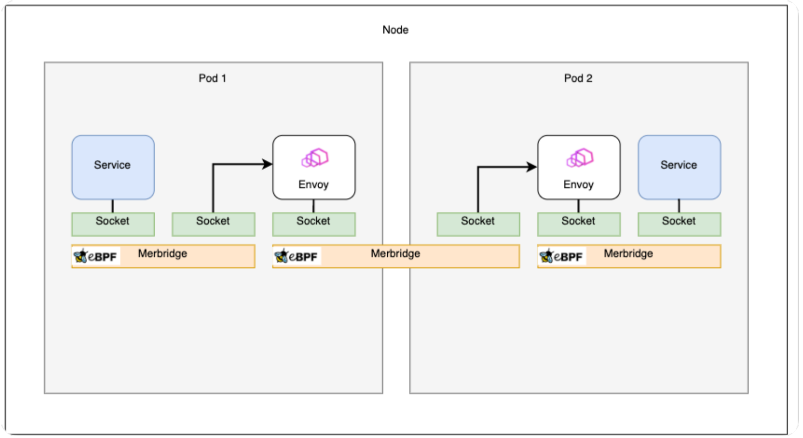


#IstioCon

How to shorten the data-path?

- eBPF provides a function [bpf_msg_redirect_hash] to process policies at the socket level
- After processing, the packet will be redirected to the destination socket referenced from the sock map.

What if quadruples have conflicts in the container scenario?
P1: 127.0.0.1:56789 => 127.0.0.1:15001
P2: 127.0.0.1:56789 => 127.0.0.1:15001
Two addresses are identical in eBPF, so how?

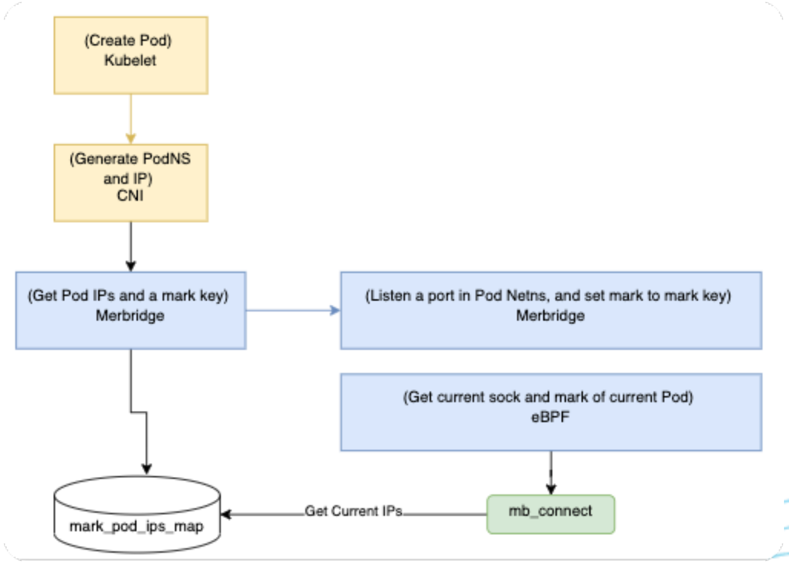


How to resolve quadruplet conflicts?

- To resolve the quadruple conflicts, any of the variables from sip, sport, dip, dport should be changed.
    1. Solution 1: revise dip from 127.0.0.1 to 127.128.x.y, where x, y are increased when new connections are established to avoid the conflicts.
    2. Solution 2: revise sip from 127.0.0.1 to Pod IP.
    3. Solution 3: introduce socket cookie to identify each of the connections
- Any flaws of solutions mentioned above?
    - For Solution 1, Merbridge adopted this in the early stage, however won’t work in the IPv6 network.
    - For Solution 2, eBPF cannot access current pod’s IP address information directly.
    - For Solution 3, only works with high kernel version greater than 5.15.

How to get the current Pod IP in the eBPF?

- Merbridge will use the ability of the CNI plugin to write the IP of the Pod into a Map when the Pod is created. It will monitor a specific port on the Netns of the Pod, and set the mark of the socket as the key of the pod IP in the map.
- When Merbridge's mb_connect program is processing the request, current IP address can be obtained by identifying the mark of the socket from current Netns, and then reading the IP from the map according to the mark.


(Create Pod)
Kubelet
(Generate PodNS and IP)
CNI
(Get Pod IPs and a mark key)
Merbridge
(Listen a port in Pod Netns, and set mark to mark key)
Merbridge
(Get current sock and mark of current Pod)
eBPF
mark_pod_ips_map
Get Current IPs
mb_connect

Live Demo！

Next Step - RoadMap

- Ingress traffic capture with XDP
- IPv6 support
- Merbridge CNI – to support Istio sidecar annotations.
- Cross-node acceleration

Others

- More details:
  - https://istio.io/latest/blog/2022/merbridge/
  - https://merbridge.io/
- Join us:
  - Slack: https://merbridge.slack.com/
  - GitHub: https://github.com/merbridge

Thank you!

@handle
https://merbridge.io

#IstioCon