User Guide

Service Mesh - TCP/IP eBPF Bypass in Istio and Envoy with Intel® Xeon® Scalable Processors

Authors

Luyao Zhong

Xu Yizhou

Ruijing Guo

Ramesh Masavarapu

1 Introduction

Istio implements service mesh by injecting a sidecar proxy for each pod. This side car will redirect all traffic from or to the original pod app to the sidecar, which may lead to degradation of network performance, since there will be additional packets traversing the TCP/IP stack both on client and server side. This paper outlines an approach to bypass the TCP/IP stack between the app and envoy and between envoy deployments on the same host.

Linux has introduced eBPF (extended Berkeley Packet Filter) - a technology that allows safe mini programs to be attached to various low-level hooks in the kernel. Focusing on networking, eBPF can be utilized to communicate between Envoy side cars on the same host through socket rather than traversing the same TCP/IP networking stack multiple times. In a host, we could have two pods (each having a microservice + Envoy side car proxy) that are communicating through eBPF sockets rather than multiple kernel TCP/IP networking stacks. This is very similar to having two processes communicate to each other using a UNIX domain socket while applications remain unchanged.

Traversing the TCP/IP networking stack introduces latencies. This solution aims to bypass TCP/IP stack on the same host to accelerate Istio/Envoy implementation of service mesh and reduce latencies. The additional benefit is that this solution benefits Kubernetes Networking without service mesh because it bypasses the Linux Kernel Networking Stack for communication between two microservices that reside on the same host.

The solution is available on 3rd Gen Intel® Xeon® Scalable processors and later.

This document is part of the Network Transformation Experience Kits.

Table of Contents

1 Introduction 1
1.1 Terminology 3
1.2 Reference Documentation 3
2 Overview 3
3 System Requirements 4
4 Build Image 4
5 Deployment Details 4
5.1 Deploy TCP/IP Bypass via Docker 5
5.2 Deploy TCP/IP Bypass in Kubernetes Cluster 5
6 Results 6
7 Summary 6

Figures

Figure 1. Bypass TCP/IP Stack using eBPF 3
Figure 2. Deployment of eBPF 4

Tables

Table 1. Terminology 3
Table 2. Reference Documents 3
Table 3: System Requirements 4

Document Revision History

Revision	Date	Description
001	December 2022	Initial release.
002	January 2023	Revised for public distribution on Intel Network Builders.

1.1 Terminology

Table 1. Terminology

Abbreviation	Description
eBPF	extended Berkeley Packet Filter
TCP/IP	Transmission Control Protocol/Internet Protocol

1.2 Reference Documentation

Table 2. Reference Documents

Reference	Source
Service Mesh – Istio and Envoy Optimizations for Intel® Xeon® Scalable Processors Solution Brief	https://networkbuilders.intel.com/solutionslibrary/service-mesh-istio-envoy-optimizations-intel-xeon-sp-solution-brief
Service Mesh - Crypto Accelerations in Istio and Envoy with Intel® Xeon® Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/service-mesh-crypto-accelerations-istio-envoy-intel-xeon-sp-user-guide
Service Mesh - Envoy Regular Expression Matching Acceleration with Hyperscan User Guide	https://networkbuilders.intel.com/solutionslibrary/service-envoy-regular-expression-matching-acceleration-hyperscan-user-guide
Service Mesh – mTLS Key Management in Istio and Envoy for Intel® Xeon® Scalable Processors User Guide	https://networkbuilders.intel.com/solutionslibrary/service-mesh-mtls-key-mgmt-istio-envoy-intel-xeon-sp-user-guide

2 Overview

This solution aims to bypass TCP/IP stack on the same host to accelerate service mesh. If two pods are located on two different nodes, the envoy-to-envoy still goes through the TCP/IP stack.

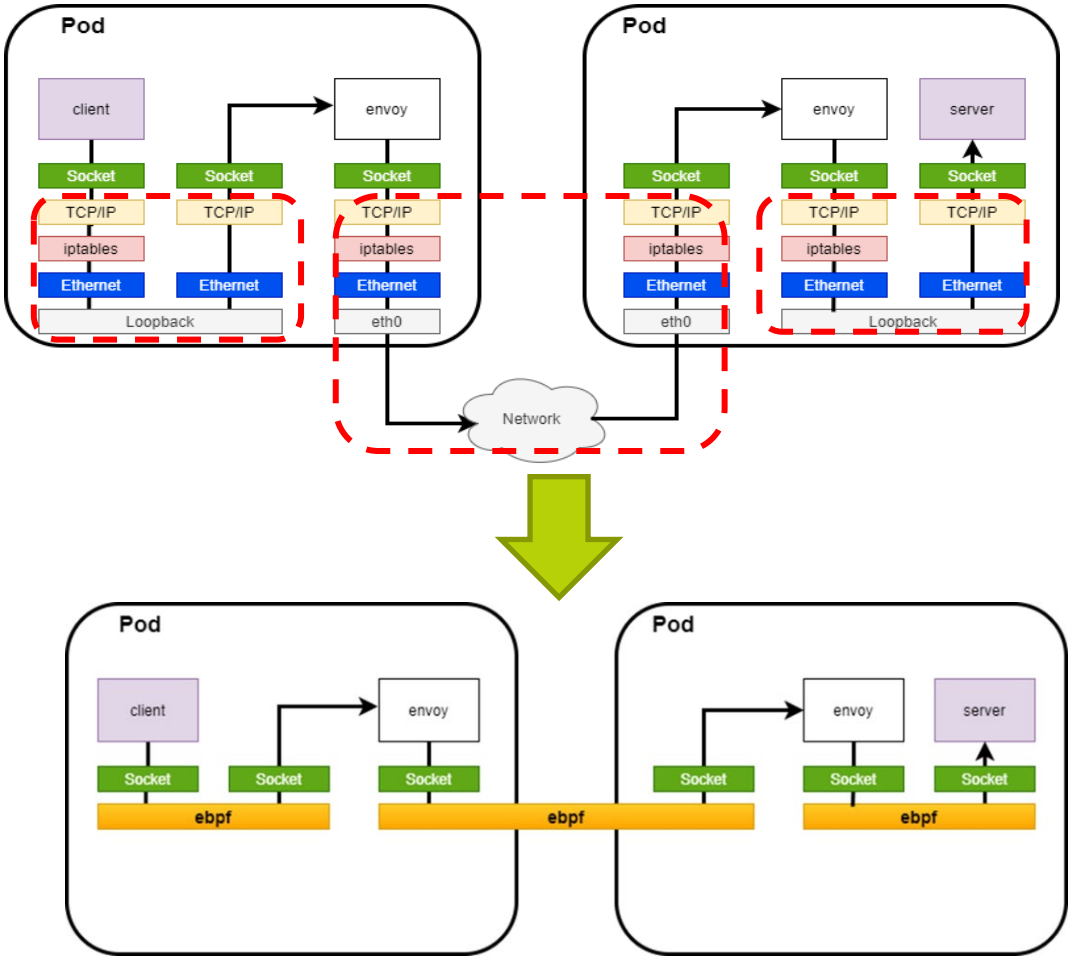

Figure 1. Bypass TCP/IP Stack using eBPF

This solution is totally independent, which:

- Does not require changes to the Linux kernel
- Does not require changes to Istio and Envoy (>= v1.10)
- Does not require changes to the CNI plugin

3 System Requirements

The system requirements for 3rd Gen Intel® Xeon® Scalable processor and 4th Gen Intel® Xeon® Scalable processor are given below:

Table 3: System Requirements

	3rd Gen Intel® Xeon® Scalable processor	4th Gen Intel® Xeon® Scalable processor
CPU Cores per socket	36	56
Total Sockets	2	2
Intel Turbo Boost	Enabled	Enabled
Operating System	Ubuntu 20.04	Ubuntu 20.04
Linux Kernel	5.4.0-74.generic or greater	5.4.0-74.generic or greater
Kubernetes	1.24.4	1.24.4
CNI	Calico 3.24.3	Calico 3.24.3

4 Build Image

If you want to build the image based on latest code, you can download source code from the "main" branch:

$ git clone https://github.com/intel/istio-tcpip-bypass.git

Then, build Docker image:

$ docker build --network=host -t bpf_bypass_tcpip.

5 Deployment Details

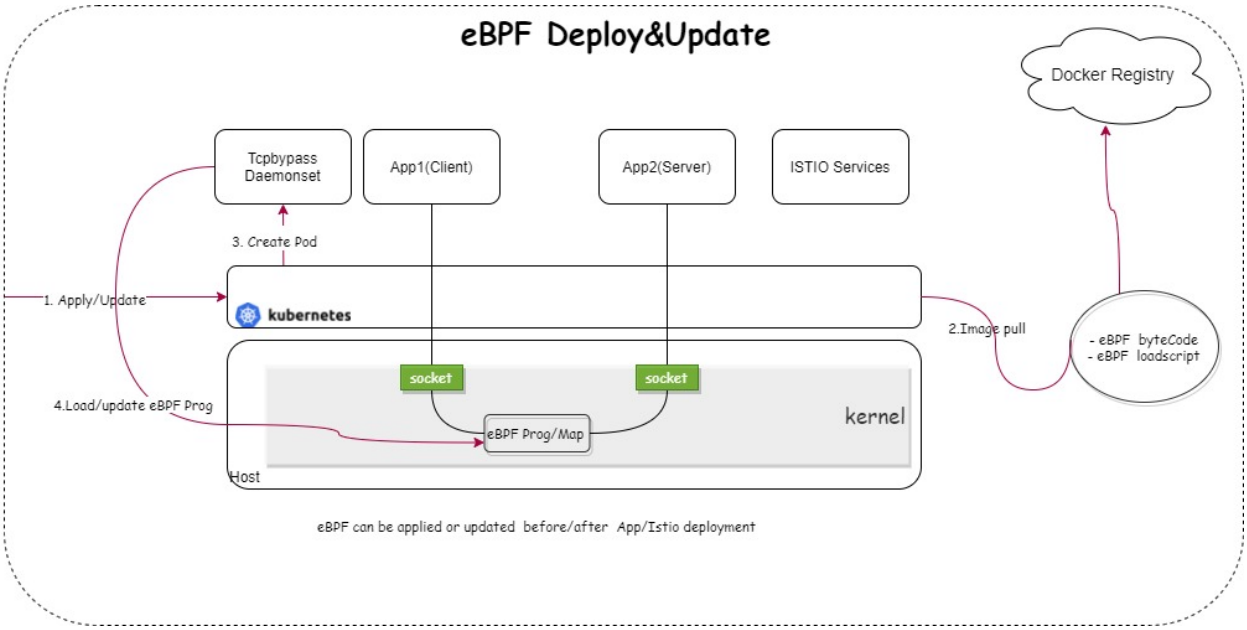

Figure 2. Deployment of eBPF

5.1 Deploy TCP/IP Bypass via Docker

1. Deploy the docker image.

```
$ docker run --mount type=bind,source=/sys/fs,target=/sys/fs,bind-propagation=rshared --net=host --privileged --name tcpip-bypass bpf_bypass_tcpip
```

5.2 Deploy TCP/IP Bypass in Kubernetes Cluster

Deploy the solution in the Kubernetes Cluster.

```
$ kubectl apply -f - <<EOF
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: bypass-tcpip
  namespace: kube-system
  labels:
    k8s-app: bypass-tcpip
spec:
  selector:
    matchLabels:
      name: bypass-tcpip
  template:
    metadata:
      labels:
        name: bypass-tcpip
    spec:
      tolerations:
      # this toleration is to have the daemonset runnable on master nodes
      # remove it if your masters can't run pods
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: bypass-tcpip
        image: intel/istio-tcpip-bypass:latest
        imagePullPolicy: IfNotPresent
        securityContext:
            privileged: true
        volumeMounts:
        - name: sysfs
          mountPath: /sys/fs
          mountPropagation: Bidirectional
      volumes:
      - name: sysfs
        hostPath:
          path: /sys/fs
EOF
```

6 Results

Deploying the TCP/IP eBPF Bypass solution results in lower latencies between microservices that deploy Istio/Envoy as a Service Mesh Environment on the 3rd Gen Intel® Xeon® Scalable processors and 4th Gen Intel® Xeon® Scalable processors. The additional benefit is that this solution benefits Kubernetes Networking without Service Mesh because it bypasses the Linux Kernel Networking Stack for communication between two microservices that reside on the same host.

7 Summary

TCP/IP eBPF Bypass solution helps in reducing latency between microservices that deploy Istio/Envoy as a service mesh deployment. The solution is available at: https://github.com/intel/istio-tcpip-bypass.

0123/DN/WIT/PDF 761925-002US