

# Packet mark in a Cloud Native world

Joe Stringer

Cilium.io

Introduction

# Google

| the internet is held together with                 | Ŷ      |
|--|--------|
| the internet is held together with duct tape       | Remove |
| the internet is held together with bubble gum      | Remove |
| the internet is held together with baling wire     | Remove |
| the internet is held together with popsicle sticks | Remove |
| the internet is held together with pixie dust      | Remove |
| the internet is held together with prayers         | Remove |
| the internet is held together with skb->mark       | Remove |

Introduction

#### Overview

1 Background

2 Use cases

**3** Observations & Challenges

Background

# Mark of the 🖑

- fw\_mark
- ct\_mark
- skb\_mark
- SO\_MARK
- xfrm\_mark
- pkt\_mark

struct sk\_buff {
 ...
 \_\_u32 mark;
 ...
}

Background

So what does the mark represent?

Nothing..

#### So what does the mark represent?

- Nothing..
- Anything!

#### So what does the mark represent?

- Nothing..
- Anything!
- 🔳 MAGIC. 🤸

#### Background





y

Today I made github.com/fwmark/registry, an informal registry (IANA style) of netfilter packet mark bits and values that are in use by software in the wild. The hope is that by making it googleable, new software will pick nonconflicting things.



https://twitter.com/dave\_universetf/status/1285752332135788544





Background

#### Cloud Native networking plugins



# Methodology

- 1 Look at CNCF landscape<sup>1</sup>
- 2 Find the project on GitHub
- 3 Search for \$mark\_name
- 4 ???
- 5 Knowledge!

<sup>1</sup> https://landscape.cncf.io/category=cloud-native-network&format=card-mode&grouping=category

#### Use cases

#### Network policy

- 1 bit, two variations:
  - 1 bit -> drop  $^2$
  - $\blacksquare 1 \text{ bit } -> \text{ allow}$
- Store complex path through rules into mark
- Typically netfilter -> netfilter



<sup>&</sup>lt;sup>2</sup> Kubernetes default

#### Transparent encryption

- 2+ bits
  - 1 bit encrypt, 1 bit decrypt
  - Variation: key index
- { eBPF, netfilter } -> xfrm



#### Virtual IP services

- 1+ bits, request DNAT
  - 1 bit: route towards bridge for DNAT
  - 30 bits representing hashed 3-tuple
- { eBPF, netfilter } -> netfilter
- $\blacksquare$  OVS -> routing -> OVS



# IP masquerade

- 1+ bits, request SNAT
  - Variation: 1 bit, Skip SNAT
  - Variation: 32 bits for source address selection
- Connection may not originate on the node
- {eBPF, OVS, netfilter} -> netfilter
- $\blacksquare \ \mathsf{eBPF} \ -> \mathsf{stack} \ -> \mathsf{eBPF}$



# Multi-homing

- 1 bit, two variations:
  - Reply via primary device
    - Default: Pod communicates via secondary device
    - Inbound connections must reply via primary device
    - Store & restore in connmark
  - Route via management interface
- $\blacksquare \ \{ \text{ socket, netfilter} \ \} \ -> \ routing$



# Application identity

#### Variable bits

- 4 bit pattern: "local" traffic
- 16+ bits: Carry Identity to destination
  - Policy routing
  - Portmap plugin
- { eBPF, netfilter } -> routing -> eBPF



# Service proxy

- $\blacksquare$  1+ bits depending on context
  - 1 bit, route locally
  - 16 bit tproxy port towards proxy
  - 16+ bit Identity from proxy
- eBPF -> { netfilter, routing }
- netfilter -> routing
- socket -> { eBPF, netfilter },



**Observations & Challenges** 

#### Marking your territory

- Bitwise usage
  - Simpler interoperability
- Full-mark
  - More values to work with
  - Most usage doesn't make use of this

# A tiny bit of overload

- Use every feature: 100+ bits
  - ...but there's only 32 bits to play with?
- Mitigation: Encode meaning in bit range
  - Use [0x0000..0x000F] rather than bits in 0xFFFF
- Mitigation: Overload bits on different paths
  - Ingress / Egress
  - Make semantics dependent on packet fields

### Sharing is caring

- Driven by common deployment scenarios
- The clearer responsibility assignment you have, the better
- Not free (in effort or in complexity)

**Observations & Challenges** 

# One does not simply understand skb mark

- Required reading: network stack diagram
- Distinct bits do not guarantee integration
  - skb, conn matches may steer packets
- Fun: replies disappear
- Proxies: Double the connections, double the fun



#### Less is more

- "If only I had more bits..."
- Consolidate subsystem usage
- Extend generic mark space?
- Formalize some use cases?

## Summary

- Powerful mechanism for cross-subsystem programming
- Uncertainty when bits are OK to use
- There are more uses than there are bits

#### Cilium

- https://cilium.io
- https://cilium.io/slack
- https://github.com/cilium/cilium
- https://twitter.com/ciliumproject

#### Mark registry

• https://github.com/fwmark/registry

