Packet mark in a Cloud Native world

Joe Stringer

Cilium.io
<table>
<thead>
<tr>
<th>Suggestion</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>the internet is held together with</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with duct tape</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with bubble gum</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with baling wire</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with popsicle sticks</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with pixie dust</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with prayers</td>
<td>Remove</td>
</tr>
<tr>
<td>the internet is held together with skb-&gt;mark</td>
<td>Remove</td>
</tr>
</tbody>
</table>
Overview

1. Background

2. Use cases

3. Observations & Challenges
Mark of the

- fw_mark
- ct_mark
- skb_mark
- SO_MARK
- xfrm_mark
- pkt_mark

```c
struct sk_buff {
    ...
    __u32 mark;
    ...
}
```
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

https://twitter.com/dave_universef/status/1285752332135788544

August 24, 2020 Packet mark in a Cloud Native world
eBPF-based Networking, Observability, and Security

Networking
- Highly Scalable Kubernetes CNI
- Kube-proxy Load Balancer Replacement
- Multi-cluster Connectivity

Observability
- Identity-aware Network Visibility
- Network Metrics + Troubleshooting
- API-aware Network Observability

Security
- Advanced Network Policy
- Security Forensics + Audit
- Transparent Encryption
Cloud Native networking plugins
Methodology

1. Look at CNCF landscape

2. Find the project on GitHub

3. Search for $mark_name

4. ???

5. Knowledge!

---

1 https://landscape.cncf.io/category=cloud-native-network&format=card-mode&grouping=category
Use cases
Network policy

- 1 bit, two variations:
  - 1 bit -> drop \(^2\)
  - 1 bit -> allow

- Store complex path through rules into mark

- Typically netfilter -> netfilter

\(^2\) Kubernetes default
Use cases

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

- 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
- eBPF -> stack -> 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

- \{ socket, netfilter \} -> routing
Use cases

Application identity

- Variable bits
  - 4 bit pattern: “local” traffic
  - 16+ bits: Carry Identity to destination
    - Policy routing
    - Portmap plugin

- \{ eBPF, netfilter \} -> routing -> eBPF
Use cases

Service proxy

- 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