# hswaw - Bugless #53

## Access to k0 pod network from routing fabric is borked

09/20/2021 10:48 PM - q3k

| | | |
|---|---|---|
| **Status:** | Assigned | |
| **Priority:** | Normal | |
| **Assignee:** | implr | |
| **Category:** | hscloud | |

**Description**

For example, from boston:

```
$ curl 10.10.25.14:9092 # matrix metrics
```

will sometimes work and sometimes get stuck.

10.10.25.0/26 is ECMP'd across all k0 hosts:

```
dcsw01.hswaw.net#show ip route 10.10.25.0/26
Codes: C - connected, S - static, K - kernel,
       O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1,
       E2 - OSPF external type 2, N1 - OSPF NSSA external type 1,
       N2 - OSPF NSSA external type2, B I - iBGP, B E - eBGP,
       R - RIP, I - ISIS, A B - BGP Aggregate, A O - OSPF Summary,
       NG - Nexthop Group Static Route

 B E    10.10.25.0/26 [200/0] via 185.236.240.35, Vlan2001
                              via 185.236.240.36, Vlan2001
                              via 185.236.240.39, Vlan2001
                              via 185.236.240.40, Vlan2001
```

However, it's a pod IP, so that's only really handled by one node - in this case, dcr01s24 / 185.236.240.40. And it seems like it only works when it gets ECMPd directly to that node, but not otherwise.

But even still, it should be properly bounced off if it hits other nodes, what's going on?

### History

#### #1 - 09/20/2021 10:55 PM - q3k

Here's an example of when it hits 10.10.25.14 (on dcr01s24) through bc01n01:

SYN through bc01n01:

```
bc01n01 $ tcpdump -i eno1 -vv -n -e host 10.10.25.14 and tcp port 9092
00:46:19.295852 00:1c:73:11:8a:83 > 00:23:ae:fe:83:c4, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 63, i
d 61506, offset 0, flags [DF], proto TCP (6), length 60)
    185.236.240.38.39384 > 10.10.25.14.9092: Flags [S], cksum 0xf440 (correct), seq 1319813603, win 64240, opt
ions [mss 1460,sackOK,TS val 3893286313 ecr 0,nop,wscale 7], length 0
```

SYN and SYN/ACK on dcr01s24

```
dcr01s24 $ tcpdump -i enp130s0f0 -vv -n -e host 10.10.25.14 and port 9092
00:46:15.039772 90:1b:0e:08:12:b8 > 90:1b:0e:31:bb:6a, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 62, i
d 2005, offset 0, flags [DF], proto TCP (6), length 60)
    185.236.240.38.39380 > 10.10.25.14.9092: Flags [S], cksum 0x1453 (correct), seq 2397532729, win 64240, opt
ions [mss 1460,sackOK,TS val 3893282056 ecr 0,nop,wscale 7], length 0
00:46:15.039917 90:1b:0e:31:bb:6a > 00:23:ae:fe:45:8c, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 63, i
d 0, offset 0, flags [DF], proto TCP (6), length 60)
    10.10.25.14.9092 > 185.236.240.38.39380: Flags [S.], cksum 0xcd59 (incorrect -> 0x365b), seq 1341792797, a
ck 2397532730, win 65236, options [mss 1400,sackOK,TS val 434785853 ecr 3893282056,nop,wscale 7], length 0
```

And SYN, SYN/ACK, ACK on boston (different flow):

```
00:50:36.884468 00:23:ae:fe:45:8c > 00:1c:73:11:8a:83, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 64, i
d 19559, offset 0, flags [DF], proto TCP (6), length 60)
    185.236.240.38.39386 > 10.10.25.14.9092: Flags [S], cksum 0xcd59 (incorrect -> 0x7951), seq 1150572718, wi
```

```
n 64240, options [mss 1460,sackOK,TS val 3893543902 ecr 0,nop,wscale 7], length 0
00:50:36.884823 90:1b:0e:31:bb:6a > 00:23:ae:fe:45:8c, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 63, i
d 0, offset 0, flags [DF], proto TCP (6), length 60)
    10.10.25.14.9092 > 185.236.240.38.39386: Flags [S.], cksum 0x54a5 (correct), seq 2657360781, ack 115057271
9, win 65236, options [mss 1400,sackOK,TS val 435047699 ecr 3893543902,nop,wscale 7], length 0
00:50:36.884867 00:23:ae:fe:45:8c > 00:1c:73:11:8a:83, ethertype IPv4 (0x0800), length 66: (tos 0x0, ttl 64, i
d 19560, offset 0, flags [DF], proto TCP (6), length 52)
    185.236.240.38.39386 > 10.10.25.14.9092: Flags [.], cksum 0xcd51 (incorrect -> 0x8014), seq 1, ack 1, win
502, options [nop,nop,TS val 3893543902 ecr 435047699], length 0
```

So that looks fine so far - SYN from boston goes through intermediary host, SYN/ACK from other side, and boston sends an ACK. But the first sign of trouble is if we look further down at dcr01s24 logs:

```
00:46:15.039917 90:1b:0e:31:bb:6a > 00:23:ae:fe:45:8c, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 63, i
d 0, offset 0, flags [DF], proto TCP (6), length 60)
    10.10.25.14.9092 > 185.236.240.38.39380: Flags [S.], cksum 0xcd59 (incorrect -> 0x365b), seq 1341792797, a
ck 2397532730, win 65236, options [mss 1400,sackOK,TS val 434785853 ecr 3893282056,nop,wscale 7], length 0
00:46:16.095067 90:1b:0e:31:bb:6a > 00:23:ae:fe:45:8c, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 63, i
d 0, offset 0, flags [DF], proto TCP (6), length 60)
    10.10.25.14.9092 > 185.236.240.38.39380: Flags [S.], cksum 0xcd59 (incorrect -> 0x323c), seq 1341792797, a
ck 2397532730, win 65236, options [mss 1400,sackOK,TS val 434786908 ecr 3893282056,nop,wscale 7], length 0
```

that's a SYN/ACK retransmit! And then even further down:

```
00:46:17.013760 00:23:ae:fe:83:20 > 90:1b:0e:31:bb:6a, ethertype IPv4 (0x0800), length 74: (tos 0x0, ttl 62, i
d 58262, offset 0, flags [DF], proto TCP (6), length 60)
    185.236.240.38.39382 > 10.10.25.14.9092: Flags [S], cksum 0x0ae5 (correct), seq 3264954427, win 64240, opt
ions [mss 1460,sackOK,TS val 3893284030 ecr 0,nop,wscale 7], length 0
```

another SYN received from boston for a different flow.

So it seems like dcr01s24 never gets the ACK from boston, and retransmits SYN/ACKs, while boston attempts another connection.

That's odd.

#### #2 - 09/20/2021 10:59 PM - q3k

Can this be because the SYN/ACK (dcr01s24 -> boston) is a direct server return (bypassing bc01n01), and that means that bc01n01 is dropping the ACK (also going through it, because ECMP is stable across the same TCP flow 5-tuple) as it can't follow that flow anymore? Do we have some iptables FORWARD rules that would reject unknown TCP flow packets?

#### #3 - 09/20/2021 11:03 PM - q3k

```
Chain KUBE-FORWARD (1 references)
 pkts bytes target     prot opt in     out     source               destination
   32  3344 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0            ctstate INVALID
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* kubernetes forward
ing rules */ mark match 0x4000/0x4000
  242 97612 ACCEPT     all  --  *      *       10.10.16.0/20        0.0.0.0/0            /* kubernetes forward
ing conntrack pod source rule */ ctstate RELATED,ESTABLISHED
  255 13330 ACCEPT     all  --  *      *       0.0.0.0/0            10.10.16.0/20        /* kubernetes forward
ing conntrack pod destination rule */ ctstate RELATED,ESTABLISHED
```

Hmmm.... And that DROP target counter certainly is increasing if I spam curl on boston...

#### #4 - 09/20/2021 11:05 PM - q3k

Seems like setting a sysfs tunable should be able to help us with this asymmetric routing issue:

https://github.com/kubernetes/kubernetes/issues/94861#issuecomment-796779433

I'll manually flip the following on our machines:

```
sysctl net.netfilter.nf_conntrack_tcp_be_liberal=1
```

And if that helps, we'll deploy it properly.

#### #5 - 09/20/2021 11:07 PM - q3k

Flipped all machines:

```
$ for m in bc01n{01,02} dcr01s{22,24}; do ssh root@$m.hswaw.net sysctl -a | grep conntrack | grep liberal; don
e
net.netfilter.nf_conntrack_tcp_be_liberal = 1
net.netfilter.nf_conntrack_tcp_be_liberal = 1
```

```
net.netfilter.nf_conntrack_tcp_be_liberal = 1
net.netfilter.nf_conntrack_tcp_be_liberal = 1
```

But that doesn't seem to have helped...

#### #6 - 09/20/2021 11:21 PM - q3k

Yeah, I don't think that's gonna help.

We have to either hack kube-proxy to remove that rule (although removing it might cause us to trigger https://github.com/kubernetes/kubernetes/issues/74839), or to stop all nodes from announcing all pod networks from every machine.

I'm now leaning towards the second, as I think that's the right solution (why would all machines announce all pod networks? it's probably a bug in our calico hacks that I didn't see) - but it will require mangling some more calico bird rule templates...

#### #7 - 09/21/2021 09:44 AM - q3k

*- Status changed from New to Assigned*

*- Assignee set to implr*

Punting this over to implr, as he's holding the backlog of calico things to be fixed or aware of wrt. upgrading it.

#### #8 - 07/04/2022 01:08 PM - q3k

*- Category set to hscloud*