Flying higher: hardware offloading with BIRD

Asbjørn Sloth Tønnesen

Fiberby ApS (Fibercity)

February 3rd, 2024
Who are Fibercity

- We operate AS42541
- Based in Copenhagen, Denmark
- Based on FTTB (Fiber to the Building)
- 1/1 Gbps connection for 17€/month
- 35k residential households connected
- Peak utilization: 140 Gbps / 15 Mpps (in+out)
Simple ISP network

- PNI’s
- IXP’s
- Transit
- DFZ router
- L2 switch
- L3 switch
- Forward-deployed streaming servers
- L3 switch
- internal routing cloud
- End-users
AS42541 routing timeline

2010
- Cisco 6500 (10G)

2015
- Brocade MLXe-4 (40G)
AS42541 routing timeline (cont.)

2020
- 2U server running Linux + BIRD v2

2021
- NIC upgraded to 100G (Mellanox/Nvidia ConnectX-6 Dx)
Router with only one active physical port
- Typically uses IEEE 802.1Q VLAN tags to reach multiple L2 networks.
- L3 router ports are expensive
- L2 switches are cheap
OVS Offload Using ASAP² Direct

- Open vSwitch (OVS)
- eSwitch
- SRIOV
- flow offload
- TC flower API
TC & flower primer

- Linux Traffic Control (TC)
- Chains, and priorities
- Actions: drop, trap, goto, redirect
- Hardware offload:
  - skip_sw / skip_hw
  - Vendor agnostic-ish
TC chains and prios

Packet fate: drop trap redirect

Rule action can goto another chain
Routing with TC flower

The anatomy of routing a packet

- Change VLAN tag
- Change source and destination MAC address
- Decrement TTL / hoplimit
- Update checksum (IPv4 only)
- Push it back out again
TC flower by example

```bash
tc filter add dev "$dev" ingress chain 4 pref 1 \
  protocol 802.1Q flower skip_sw \n  vlan_ethtype_ipv4 \n  action vlan modify id "$vlan_id" \n  pipe action pedit ex \n  munge eth dst set "$new_dst_mac" \n  munge eth src set "$new_src_mac" \n  munge ip ttl dec \n  pipe csum ip4h \n  pipe action mirred egress redirect dev "$dev"
```
TC flower by example

tc filter show dev enp2s0f0np0 ingress chain 4

1. filter protocol 802.1Q pref 1 flower
2. filter protocol 802.1Q pref 1 flower handle 0x1
   - vlan_ethtype ip
   - eth_type ipv4
   - skip_sw
3. in_hw in_hw_count 1
   - action order 1: vlan modify id 620 protocol 802.1Q pipe
   - action order 2: pedit action pipe keys 5
   - action order 3: mirred (Egress Redirect to device enp2s0f0np0) stolen
4. index 1 ref 1 bind 1
5. used_hw_stats delayed
tc filter show dev enp2s0f0np0 ingress chain 4

action order 1: vlan modify id 620 protocol 802.1Q pipe
  index 1 ref 1 bind 1
  used_hw_stats delayed

action order 2: pedit action pipe keys 5
  index 1 ref 1 bind 1
  key #0 at eth+0: val xxxxxxxxxx mask 00000000
  key #1 at eth+4: val xxxx0000 mask 0000ffff
  key #2 at eth+4: val 0000xxxx mask ffff0000
  key #3 at eth+8: val xxxxxxxx mask 00000000
  key #4 at ipv4+8: add ff000000 mask 00ffff00
  used_hw_stats delayed

action order 3: mirred (Egress Redirect to device enp2s0f0np0) stolen
TC flower by example (cont.)

tc -s filter show dev enp2s0f0np0 ingress chain 4

1. action order 3: mirred (Egress Redirect to device enp2s0f0np0) 
   stolen
2. index 1 ref 1 bind 1 installed 1241229 sec used 0 sec
3. Action statistics:
4. Sent 2828989939137069 bytes 3904698201 pkt 
   (dropped 0, overlimits 0 requeues 0)
5. Sent software 0 bytes 0 pkt
6. Sent hardware 2828989939137069 bytes 3904698201 pkt
7. backlog 0b 0p requeues 0
8. used_hw_stats delayed
Static hardware offload

Inbound traffic is simple

- Always known to be online next hop
- Prefixes don’t change
- Next-hop is a L3 switch.
### Chaining it together

**Chain 0**
- Send IPv4 to chain 1
- Send IPv6 to chain 2

**Chain 1/2**
- Rule 1: If TTL is expiring trap packet
- Rule 10-: Match some linknets, and trap em
- Rule 100-: Match inbound destinations, and goto chain 4/6

**Chain 4/6**
- Forward packet
Offloaded performance

Scalability of skip_sw rules

<n> non-matching rules + one final matching forward rule

pktgen  HW forwarded
flower-route

- Small daemon to synchronize routes to hardware
- Runs an event loop, and have two Netlink sockets
- BIRD exports a subset of routes to a dedicated routing table
- Extracts links, neighbours, routes and TC ruleset from the kernel
- Maintains TC ruleset in kernel
- Defensively coded, only making minimal changes
- Licensed as GPLv2+
Block diagram

- Links
- Neighbours
- Aux. routing table
- BIRD
- BGP
- flower-routed
- TC ruleset

- Static hardware offload
- flower-route
- Intro
- Wrapup
Bird config

- An extra kernel protocol per AF to an extra kernel table.
- Pipe protocols with filter picking routes to offload.
Future work

- More flexible configuration:
  - Reverse path filtering (BCP38)
  - Offload to directly connected hosts
- MTU differences
- ECMP support
- 200G LAG
Alternative applications

Figure 2: Thinclient router setup

25G router @ ~15W

- Dell Wyse 3030LT (~3W, peak: 5W)
- M.2 (E-key)
- -> mini-PCIe
- -> PCIe riser
- -> Mellanox ConnectX-5 2x25G (~7W w/ DAC)
- and a fan
Patches welcome https://github.com/fiberby-dk/flower-route