netconf

Roopa Prabhu, 9feb2016

lwt, mpls, netlink api review

lwt

- Two kinds of users today:
  - using redirection without a tunnel netdevice (mpls, ila)
  - using dst_metadata and a single tunnel netdevice (flow based tunnels: vxlan, geneve)

lwt redirection optimizations: too early, too late

- Two redirections available today (would more redirections be useful ?):
  - input redirection
  - output redirection
- Redirection too early:
    - In some cases, output redirection could wait for ip fragmentation to finish before redirecting to tunnel output (eg., mpls)
- Redirection too late:
    - input redirection where it is today, hurts early demux (eg., ILA)

lwt redirection optimizations: too early, too late (Contd)

- Need for more strategically placed redirections/lwt-hooks in the future:
  - Example: an xmit redirect for mpls after ip fragmentation
    - Redirect in ip_finish_output2
    - https://github.com/CumulusNetworks/net-next/commit/0cec8aa813af6904191c273ee07830875b3eb8d4
  - Possibly piggy back on netfilter hooks ?

lwt: mtu headroom in lwt_state (for mpls ip tunnels)

```
struct lwtunnel_state {
    __u16       type;
    __u16       flags;
    atomic_t      refcnt;
    int         (*orig_output)(struct net *net, struct sock *sk,
                              struct sk_buff *skb);
    int         (*orig_input)(struct sk_buff *);
    int         len;
    __u16        headroom;
    __u8        data[0];
```

struct lwtunnel_state -> headroom is initialized by the light weight tunnel driver

```
ip_skb_dst_mtu {
        ....
        mtu -= dst->lwtstate->headroom;
        ....
}
```

mpls

mpls futures

- stats (yes!, yet another stat for jamal :)
- mpls egress pop and lookup
- mpls VPN
- mpls ping and traceroute
- mpls hardware offload

Netlink API review

keeping netlink api extensible

- Encourage use of extensible nested attributes
- Most common errors today are due to copy pasting code from other subsystems
- kernel does not return error on attributes it does not care (the thing we discussed about yesterday!)
- Once you put iproute2 code out there, people use that as an example. As long as kernel does not error out, and it works .. the usage of the api lives ...until we need to extend it (next slide)

Dump message header format inconsistent with new/del:

RTM_NEWNEIGH - struct ndmsg

RTM_DELNEIGH - struct ndmsg

RTM_GETNEIGH - struct ifinfomsg

RTM_NEWMDB - struct br_port_msg

RTM_DELMDB - struct br_port_msg

RTM_GETMDB - struct ifinfomsg

The reason this happened over time is because:

- libnetlink.c published an rtnl_wilddump_request api which implicitly sent 'struct ifinfomsg' for all dumps whether the kernel supported it or not
- And people started using the api regardless of type of dump or type of family (basically cut-copy-paste error)

iproute2: check attribute type before dereferencing

```
for (i = RTA_DATA(attr); RTA_OK(i, rem); i = RTA_NEXT(i, rem)) {
                e = RTA_DATA(i);
                print_mdb_entry(f, ifindex, e, n);
        }
}
```

ie ..don’t assume an attribute type

netlink api review

- Would having a netlink api guideline document help ?

Other misc areas

other misc areas (you will see patches soon..)

- bridge:
  - per vlan stats
  - igmp stats
  - more per vlan things
- iproute2: complete support for bridge and bond netlink attributes
- vxlan (changelink, react to netdev events..,)

Thank you!