Switchdev/rocker status + FIB offload, netconf 2015

Want to get to FIB offload, but first background...

2014 netconf (August)

Jiri/Scott had rocker driver/device glued to sw_flow in openvswitch.ko

- Basic L2 bridging stuff working
- Jamal/others don't like it
    - (mostly because it has the word "flow' in it)
    - Jamal wants L2/L3 offload via netlink
- OVS guys don't want the hw interface at that level

Ok, fine, let's do L2/L3 offload…

Jiri/Scott release L2 offload support + rocker driver

- Modified ndo_fdb_xxx ops to be vlan-aware for static FBD entries added to hw
- Added ndo op to push port STP state down to hw
- Added notification for hw FDB learn/forget events back to bridge driver
- Added some tweaks to setlink/getlink for new knobs
    - learning_sync on/off (to sync port's FDB with bridge's)
    - learning on/off (at hw port level)

Yes! Included in 3.18. Now we have L2 offload to multiple switch instances on multiple vlan bridges. L2 mgmt is existing netlink. L2 STP ctrl remains in software.

More tweaks refinements for 3.19/3.20 from Roopa and others

- rocker driver/device stats
- NETIF_F_HW_SWITCH_OFFLOAD port flag added
- tweaks to set/get/dellink

What's missing for L2 offload?

- (kernel/driver) should use bridge PVID for picking untagged internal VLAN ID
- (driver) Need to add LAG bonding support
    - Need to know port membership
    - Need to know port active status
- (kernel/driver) Duplicate pkts on bcast/mcast flooding
- (kernel/driver) Need to revise FDB ageing strategy to move ageing function to hw
- multicast (mdb)
- more?

Nice surprises:

- Florian Fainelli working to move DSA to switchdev. Just about 100% switchdev applicable.
- John Fastabend uses rocker for prototyping new flow API
- Pablo Neira Ayuso using rocker to prototype nf hw offload for ACLs
- (Shhh) Intel preparing switchdev-based kernel driver for net-gen switch

Not so nice surprises:

- Still trying to get rocker device upstream into Qemu
    - so, so close

L3 offloading:

Initial patches based on rocker are IPv4 only, and no ECMP support. Crazy good performance for scaled route add/del tests.

Ready to send v2 patches

- v1 didn't handle route change correctly
    - v2 now does route change atomically down to device

Remaining work:

- (driver/device) Add ECMP support
- (kernel/driver/device) Add ECMP hash algo selector policy
- Add IPv6 support
- L3 multicast routing support

Implementation notes:

- Adds two new ndo ops to add/del FIB entry
    - add op has modify semantics for existing entries
    - ops are hooked deep in the fib_trie.c code
        - basically hooks added where NEWROUTE/DELROUTE echos are generated
    - Passes fib_info and other internals to driver directly.
- Driver must resolve nexthops to neigh MAC
    - driver listens to ARP neigh table updates and keeps copy
    - if route's nexthops aren't in local copy, driver sends ARP requests
- Synchronous call path from user-space netlink process to program hw and return status
    - For add op, use simple scheme for adding/modifying route to sw only or sw+hw, and handle failures inline.

Pseudo code:

```
# ip route add 12.0.0.1/32 nexthop via 11.0.0.1 dev eth1
        RTM_NEWROUTE
                fib_table_insert()
                        validate request
                        allocate fib alias
                        err = call ndo op to add fib entry to hw
                        if err and err != -EOPNOTSUPP
                                return err
                        install in kernel FIB
```

Driver returns:

err = 0	route programmed into hw
err = -EOPNOTSUPP	route not supported on hw, install in sw only
err = <some other err>	route failed on hw add, don't install in sw either