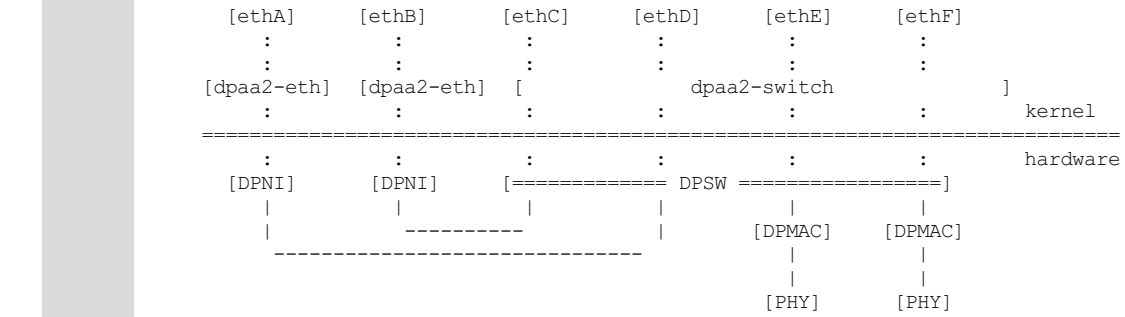
DPAA2 Switch driver

The DPAA2 Switch driver probes on the Datapath Switch (DPSW) object which can be instantiated on the following DPAA2 SoCs and their variants: LS2088A and LX2160A.

The driver uses the switch device driver model and exposes each switch port as a network interface, which can be included in a bridge or used as a standalone interface. Traffic switched between ports is offloaded into the hardware.

The DPSW can have ports connected to DPNIs or to DPMACs for external access.



Creating an Ethernet Switch

The dpaa2-switch driver probes on DPSW devices found on the fsl-mc bus. These devices can be either created statically through the boot time configuration file - DataPath Layout (DPL) - or at runtime using the DPAA2 object APIs (incorporated already into the restool userspace tool).

At the moment, the dpaa2-switch driver imposes the following restrictions on the DPSW object that it will probe:

- The minimum number of FDBs should be at least equal to the number of switch interfaces. This is necessary so that separation of switch ports can be done, ie when not under a bridge, each switch port will have its own FDB.

  ```
  fsl_dpaa2_switch dpsw.0: The number of FDBs is lower than the number of ports, cannot probe
  ```

- Both the broadcast and flooding configuration should be per FDB. This enables the driver to restrict the broadcast and flooding domains of each FDB depending on the switch ports that are sharing it (aka are under the same bridge).

  ```
  fsl_dpaa2_switch dpsw.0: Flooding domain is not per FDB, cannot probe
  fsl_dpaa2_switch dpsw.0: Broadcast domain is not per FDB, cannot probe
  ```

- The control interface of the switch should not be disabled (DPSW_OPT_CTRL_IF_DIS not passed as a create time option). Without the control interface, the driver is not capable to provide proper Rx/Tx traffic support on the switch port netdevices.

  ```
  fsl_dpaa2_switch dpsw.0: Control Interface is disabled, cannot probe
  ```

Besides the configuration of the actual DPSW object, the dpaa2-switch driver will need the following DPAA2 objects:

- 1 DPMCP - A Management Command Portal object is needed for any interraction with the MC firmware.
- 1 DPBP - A Buffer Pool is used for seeding buffers intended for the Rx path on the control interface.
- Access to at least one DPIO object (Software Portal) is needed for any enqueue/dequeue operation to be performed on the control interface queues. The DPIO object will be shared, no need for a private one.

Switching features

The driver supports the configuration of L2 forwarding rules in hardware for port bridging as well as standalone usage of the independent switch interfaces.

The hardware is not configurable with respect to VLAN awareness, thus any DPAA2 switch port should be used only in usecases with a VLAN aware bridge:

```
$ ip link add dev br0 type bridge vlan_filtering 1

$ ip link add dev br1 type bridge
$ ip link set dev ethX master br1
Error: fsl_dpaa2_switch: Cannot join a VLAN-unaware bridge
```

Topology and loop detection through STP is supported when `stp_state 1` is used at bridge create

```
$ ip link add dev br0 type bridge vlan_filtering 1 stp_state 1
```

L2 FDB manipulation (add/delete/dump) is supported.

HW FDB learning can be configured on each switch port independently through bridge commands. When the HW learning is

disabled, a fast age procedure will be run and any previously learnt addresses will be removed.

```
$ bridge link set dev ethX learning off
$ bridge link set dev ethX learning on
```

Restricting the unknown unicast and multicast flooding domain is supported, but not independently of each other:

```
$ ip link set dev ethX type bridge_slave flood off mcast_flood off
$ ip link set dev ethX type bridge_slave flood off mcast_flood on
Error: fsl_dpaa2_switch: Cannot configure multicast flooding independently of unicast.
```

Broadcast flooding on a switch port can be disabled/enabled through the brport sysfs:

```
$ echo 0 > /sys/bus/fsl-mc/devices/dpsw.Y/net/ethX/brport/broadcast_flood
```

Offloads

Routing actions (redirect, trap, drop)

The DPAA2 switch is able to offload flow-based redirection of packets making use of ACL tables. Shared filter blocks are supported by sharing a single ACL table between multiple ports.

The following flow keys are supported:

- Ethernet: dst_mac/src_mac
- IPv4: dst_ip/src_ip/ip_proto/tos
- VLAN: vlan_id/vlan_prio/vlan_tpid/vlan_dei
- L4: dst_port/src_port

Also, the matchall filter can be used to redirect the entire traffic received on a port.

As per flow actions, the following are supported:

- drop
- mirred egress redirect
- trap

Each ACL entry (filter) can be setup with only one of the listed actions.

Example 1: send frames received on eth4 with a SA of 00:01:02:03:04:05 to the CPU:

```
$ tc qdisc add dev eth4 clsact
$ tc filter add dev eth4 ingress flower src_mac 00:01:02:03:04:05 skip_sw action trap
```

Example 2: drop frames received on eth4 with VID 100 and PCP of 3:

```
$ tc filter add dev eth4 ingress protocol 802.1q flower skip_sw vlan_id 100 vlan_prio 3 action drop
```

Example 3: redirect all frames received on eth4 to eth1:

```
$ tc filter add dev eth4 ingress matchall action mirred egress redirect dev eth1
```

Example 4: Use a single shared filter block on both eth5 and eth6:

```
$ tc qdisc add dev eth5 ingress_block 1 clsact
$ tc qdisc add dev eth6 ingress_block 1 clsact
$ tc filter add block 1 ingress flower dst_mac 00:01:02:03:04:04 skip_sw \
        action trap
$ tc filter add block 1 ingress protocol ipv4 flower src_ip 192.168.1.1 skip_sw \
        action mirred egress redirect dev eth3
```

Mirroring

The DPAA2 switch supports only per port mirroring and per VLAN mirroring. Adding mirroring filters in shared blocks is also supported.

When using the tc-flower classifier with the 802.1q protocol, only the "vlan_id" key will be accepted. Mirroring based on any other fields from the 802.1q protocol will be rejected:

```
$ tc qdisc add dev eth8 ingress_block 1 clsact
$ tc filter add block 1 ingress protocol 802.1q flower skip_sw vlan_prio 3 action mirred egress mirror dev
Error: fsl_dpaa2_switch: Only matching on VLAN ID supported.
We have an error talking to the kernel
```

If a mirroring VLAN filter is requested on a port, the VLAN must to be installed on the switch port in question either using "bridge" or by creating a VLAN upper device if the switch port is used as a standalone interface:

```
$ tc qdisc add dev eth8 ingress_block 1 clsact
$ tc filter add block 1 ingress protocol 802.1q flower skip_sw vlan_id 200 action mirred egress mirror dev
Error: VLAN must be installed on the switch port.
We have an error talking to the kernel

$ bridge vlan add vid 200 dev eth8
$ tc filter add block 1 ingress protocol 802.1q flower skip_sw vlan_id 200 action mirred egress mirror dev
```

```
$ ip link add link eth8 name eth8.200 type vlan id 200
$ tc filter add block 1 ingress protocol 802.1q flower skip_sw vlan_id 200 action mirred egress mirror dev
```

Also, it should be noted that the mirrored traffic will be subject to the same egress restrictions as any other traffic. This means that when a mirrored packet will reach the mirror port, if the VLAN found in the packet is not installed on the port it will get dropped.

The DPAA2 switch supports only a single mirroring destination, thus multiple mirror rules can be installed but their "to" port has to be the same:

```
$ tc filter add block 1 ingress protocol 802.1q flower skip_sw vlan_id 200 action mirred egress mirror dev
$ tc filter add block 1 ingress protocol 802.1q flower skip_sw vlan_id 100 action mirred egress mirror dev
Error: fsl_dpaa2_switch: Multiple mirror ports not supported.
We have an error talking to the kernel
```