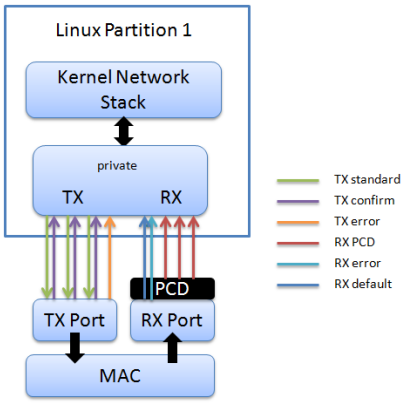
Shared-MAC and MAC-less Implementation in DPAA Linux Kernel Driver

A shared-MAC device is one that can be used from two Linux and/or USDPAA partitions. Shared-MAC net device can be used in two scenarios, two or more Linux separate partitions under control of hypervisor(topaz), one Linux and one USDPAA running in the same partition.

A MAC-less device is also called a virtual controller in Linux Ethernet Driver, it doesn't have a physical FMAN port, but that is transparent to Linux Kernel and user space applications, only the DPAA-Ethernet driver is aware of the difference.

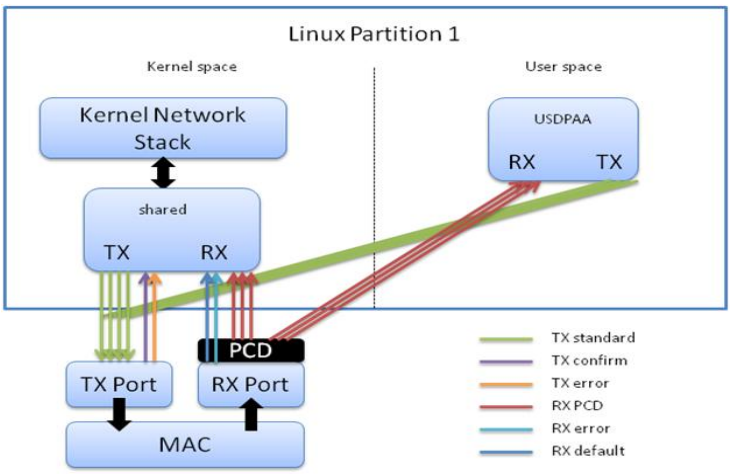
1. DPAA Ethernet Driver Types

The private Ethernet port is as the following, Linux private Ethernet driver is common Linux Ethernet diver, it is related to "fsl,dpa-ethernet" node in the device tree, dpaa_eth_priv_probe() function of dpaa_eth.c.

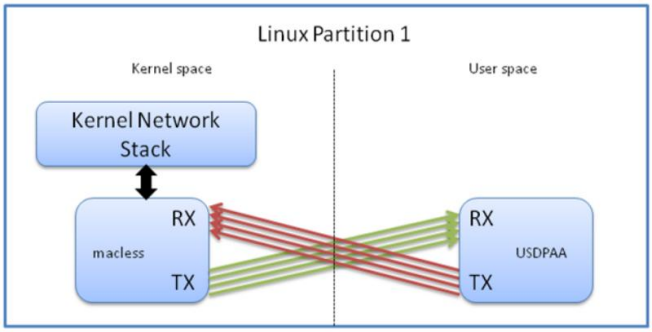


The shared-MAC driver has the fsl,dpa-ethernet-shared string as compatible string in the device tree, it related to dpaa_eth_shared_probe function of the file dpaa_eth_shared.c.

USDPAA and Linux stack communicating through a single MAC device, using Shared driver inside a single Linux partition.



The Macless DPAA Ethernet driver has fsl,dpa-ethernet-macless as compatible string in the device tree, related to dpaa_eth_macless_probe function of source code dpaa_eth_macless.c. For some applications, USDPAA needs the benefits and flexibility of Linux networking capabilities. To connect to the Linux Network Stack, it uses a Macless DPAA Ethernet Driver. The scenario is as the following.



2. BMan Driver for shared-MAC and MAC-less port

For Linux private Ethernet port, the buffer pool device tree is not needed, in the private Ethernet driver, a single buffer pool is dynamically created and seeded, the number and size of buffer pool is decided internally private driver. Shared-MAC or MAC-less Ethernet driver is different with private Ethernet driver, the frames are transmitted between different memory space, the buffer pool needs to be explicitly specified.

BMAN buffer pool node is defined as the following in the device tree.

bp7: buffer-pool@7 {
compatible = "fsl,b4860-bpool", "fsl,bpool";
fsl,bpid = <7>;
fsl,bpool-ethernet-cfg = <0 256 0 192 0 0>;
fsl,bpool-thresholds = <0x400 0xc00 0x0 0x0>;
};
Property bpid specifies the hardware index of the buffer pool.
Property fsl,bpoolethernet-cfg is parsed solely by DPAA Ethernet driver that has a reference to this buffer pool, it has the following meaning.
fsl,bpool-ethernet-cfg = <count size base_address>;
count - represents the number of buffers from the buffer pool;
size – *buffer size*
base_address - physical address of the buffer pool. Because two Linux partition have different memory spaces, this physical address will be mapped in both partitions.
The example above declares a buffer pool with buffer pool ID 7, and describes a pool with 256 192-byte buffers, occupying the memory region from 0x40000000 to 0x40000000 + 256*192.
fsl,bpool-thresholds – this property specify the threshold of entry and exit the depletion state for both the software portals and hardware portals.
Shared-MAC Ethernet port is defined as the following in the device tree.
ethernet@9 {
compatible = "fsl,b4860-dpa-ethernet-shared", "fsl,dpa-ethernet-shared";
fsl,bman-buffer-pools = <&bp17>;
fsl,qman-frame-queues-rx = <0x5e 1 0x5f 1 0x2000 3>;
fsl,qman-frame-queues-tx = <0 1 0 1 0x3000 8>;
};
bp17: buffer-pool@17 {
compatible = "fsl,b4860-bpool", "fsl,bpool";
fsl,bpid = <17>;

fsl,bpool-ethernet-cfg = <0 2048 0 1728 0 0>;
fsl,bpool-thresholds = <0x100 0x300 0x0 0x0>;
};

Linux DPAA driver dpa_bp_probe searches for compatible property "fsl,bman-buffer-pools", in the scenario shared-MAC between Linux stack and USDPAA, USDPAA allocates the buffer pool used in the communication, the above device tree nodes will also parsed by USDPAA. Because the communication occurs inside a single Linux partition, there is no need for the buffer pool to advertise its physical address, hence the 0 address representing an invalid physical address. Because the buffer allocated by USDPAA, these are in the virtual address space of USDPAA. The mapping of the USDPAA's user-space memory space to kernel space is done through kernel kmap/kunmap primitives.

The DPAA driver dpa_bp_create will create the new buffer pool with bpid specified in dts node, then seeds new buffer pool. In Freescale SDK USDPAA and shared-MAC scenario, USDPAA application will allocates and seed the buffer pools.

DPAA driver function dpaa_eth_init_ports initializes the buffer pool to FMAN Rx port, during Packet Rx process, FMAN will automatically acquire new buffers from the buffer pool for the new incoming frames.

Linux DPAA MAC-less Ethernet Driver is similar as Shared-MAC, dpa_bp_probe search for compatible property "fsl,bman-buffer-pools" and dpa_bp_create creates the buffer pool, except that "dpaa_eth_init_ports" is not needed, because there is no "MAC" existing in MAC-less port.

3. QMan Driver for shared-MAC and MAC-less port

In QMan, there are 3 types of Rx/Tx FQ, error, default and PCD Rx FQ, error, confirm and Tx FQ.

The Rx/Tx FQID property for Linux Ethernet port is specified in driver, it also can be statically defined in the above shared-MAC ethernet@9 device node.
"fsl,qman-frame-queues-rx" property specifies the Rx "Error FQID, default FQID and PCD FDID".
"fsl,qman-frame-queues-tx" property specified the TX "Err FQID, confirm FQID & Tx FQID".

The function dpa_fq_probe_mac probes the Rx/Tx FQID of the share-MAC port. In the scenarios of MAC-less between partitions, Rx queues in the first partition are the same as Tx queues in the second partition. There are 8 queues in the example, this is necessary because Tx code will expect at least one queue per core.

First partition:
dpa-ethernet@10 {
compatible = "fsl,p4080-dpa-ethernet", "fsl,dpa-ethernet-macless";
fsl,qman-frame-queues-rx = ***<0x4000 8>;***
fsl,qman-frame-queues-tx = <0x4008 8>;
local-mac-address = [02 00 c0 a8 6f fe];
};

Second partition:
dpa-ethernet@10 {
compatible = "fsl,p4080-dpa-ethernet", "fsl,dpa-ethernet-macless";
fsl,qman-frame-queues-rx = <0x4008 8>;
fsl,qman-frame-queues-tx = ***<0x4000 8>;***
local-mac-address = [02 00 c0 a8 79 fe];
};

A shared MAC interface is similar to MAC-less interface, except one of the interface has a MAC, the Tx queues in the secondary partition are the same as the Tx queues in the main partition. This is because the convention for queue initialization is that only MAC-having interface initialize both the Rx & Tx queues. MAC-less interfaces only initialize their Rx queues.

First partition:

```
dpa-ethernet@1 {
    compatible = "fsl,p4080-dpa-ethernet", "fsl,dpa-ethernet-shared";
    fsl,qman-frame-queues-rx = <0x230 1 0x231 1 0x210 3>;
    fsl,qman-frame-queues-tx = <0x234 1 0x238 1 0x200 8>;
};
```

Second partition:

```
dpa-ethernet@20 {
    compatible = "fsl,p4080-dpa-ethernet", "fsl,dpa-ethernet-macless";
    fsl,qman-frame-queues-rx = <0x220 3>;
    fsl,qman-frame-queues-tx = <0x200 8>;
    delete-prop = "fsl,fman-mac";
    local-mac-address = [02 00 c0 a8 a1 fe];
};
```

DPAA driver dpa_get_channel and dpaa_eth_add_channel, get the pool channel ID and add it to each CPU/Portal's SDQCR register.
The function dpa_fq_setup and dpa_fq_init create and initialize the Tx/Rx FQ. For the Shared-MAC scenario, the main partition create and initializes the Rx/Tx FQs. For the secondary partition, the MAC-less Tx FQ will only create but not initialize, the MAC-less Rx FQ will be created and initialize normally as the PCD FQs. For MAC-less scenario, Rx FQs will be created and initialized as PCD queues, Tx queues will only be created but not initialized.

4. Running Shared-MAC between USDPAA and Linux

USDPAA shared-MAC is implemented by the application lpm ipfwd.

On Linux side, the shared-MAC port can communicate with host by ping.

On USDPAA side, the shared-MAC port can function as other USDPAA ports to forward packets.

The shared-MAC port allocated as the following.

On P2041/P3041/P5020: fm1-10g
On P4080/P5040: fm2-10g
On B4860: fm1-mac10
On T4240QDS/RDB: fm2-mac10
On T2080QDS/RDB: fm2-mac2
On T1040RDB: fm1-gb4

If the shared-MAC dtb is not compiled by default, it could be compiled as the following.

```
# cd <source_path>/linux/
# scripts/dtc/dtc -I dts -O dtb -o <platform>-usdpaa-shared-interfaces.dtb -b 0 -p 1024
arch/powerpc/boot/dts/<platform> -usdpaa-shared-interfaces.dts
```

Run shared-MAC application

1. Launch LPM ipfwd application.
   # cd /usr/etc
   Set FMan PCD for each port:
   # fmc -c <Serdes Config> -p <Policy file> -a
   Policy file:
   usdpaa_policy_hash_shared_mac_ipv4.xml
   Launch LPM ipfwd application
   # lpm_ipfwd_app -c <Serdes Config> -p <Policy file> -d 0x10000000 -b 1600:0:1600

2. Display the available Ethernet ports in USDPAA and add route and ARP entries.

   Connect the board with ssh
   Display available ethernet ports in USDPAA by:
   # lpm_ipfwd_app -E -a
   Note: This will print Port_Num for all enabled ports

   Assign IP address to Shared-Mac.
   # lpm_ipfwd_config -P pid -F -a <IPSeg>.1 -i <Port_Num>
   Note, IP segment varies on different platforms:
   For P2041/P3041/P5020/T1040: it's 192.168.88.0/24
   For P4080/P5040/B4860/T4240/T2080: it's 192.168.44.0/24

   Add route entries and ARP entries by:
   #lpm_ipfwd_config -P <pid> -G -s <IPSeg>.3 -m 00:e0:0c:00:d7:05 -r true
   #lpm_ipfwd_config -P <pid> -B -d <IPSeg>.3 -g <IPSeg>.3 -n 32 -c 1
   (dest ip will be stored in 5 tables which causes longest lookup)
   (pid is the Process Identifier of lpm_ipfwd_app)

3. Set ip address to the shared-mac port on the linux side.
   # ifconfig <shared-mac port> <IPSeg>.4

4. Run traffic on the shared-MAC and can see traffic splitting amongst kernel and USDPAA.
   Create Raw Stream
   'Dest Mac', the mac of shared-mac port on board
   'Source IP', <IPSeg>.2
   'Dest IP', <IPSeg>.3
   Ping linux side ip addr(<IPSeg>.4)
   The packets can be forwarded and ping succeeds.