Dedicated Channel and Pool Channel Used in Linux Kernel and USDPAA

1. Basic Concept of QMAN Channels

When the frames on a FQ are ready to be processed, the FQ is enqueued onto a work queue(WQ). WQ are organized into channels. A channel is a fixed, hardware-defined association of 8 work queues, also though of “priority work queues”. There are two types of WQ channels defined in QMAN:

Dedicated channels, which are always serviced by a single entity. QMAN software portals allow QMAN to be used by logically separated units of software. Each software portal has its own dedicated channel(of 8 work queues), that only it may dequeue from.

Pool channels, which are serviced by pool of like entities, such as a pool of processor cores. There are 15 “pool channels” from which any software portal can dequeue, this is typically used for load-balancing or load spreading.

2. Dedicated Channel Used in Flow Order Preservation Scenario

The DPAA helps address packet order issues that may occur as result of running an application in a multiple processor environment. There are several ways to leverage the DPAA to handle follow order in a system, the simplest technique for preserving order is to route the ingress traffic of an individual flow to a particular core. The single-core software requires using a dedicated channel to the processors. The FMAN PCD can be configured to either directly match a flow to a core or to use the hashing to provide traffic spreading that offer a permanent flow-to-core affinity.

The FMAN can be configured to extract data from a field or fields within the data frame, build a key from that, and then hash the resultant key into a small number. An example is to define a flow as an IPv4 source + IPv4 destination address, both files together constitute 64 bits, so there are 264 possible combinations, the FMAN then uses a hash algorithm to compress this into a manageable number of bits. Because the hash algorithm is consistent, packets from a particular flow always go to the same FQ, then this frame queue can be assigned to the specific core using a dedicated channel. The following figure 1 describes using hash to assign one FQ to a dedicated channel assigned to the specific core in the flow order preservation scenario.

3. Pool Channel Used in Order Preservation with Hold Active Scheduling

Some users may need the pool channel approach where multiple cores may pool together to service a specific set of flows. When the application doesn’t require flows to be processed in order, the pool channel approach allows the easiest method for balancing the processing load. When order is required, the software must maintain order.

Order is preserved as long as two or more cores never process frames from the same flow at the same time. This can also be accomplished by using hold active scheduling along with discrete consumption acknowledgment (DCA) mode associated with the DQRR. Although flow affinity may change for an FQ with hold active scheduling when the FQ is emptied, if the new work (from frames received after the FQ is emptied) is held off until all previous work completes, then the flow will not be processed by multiple cores simultaneously, thereby preserving order.

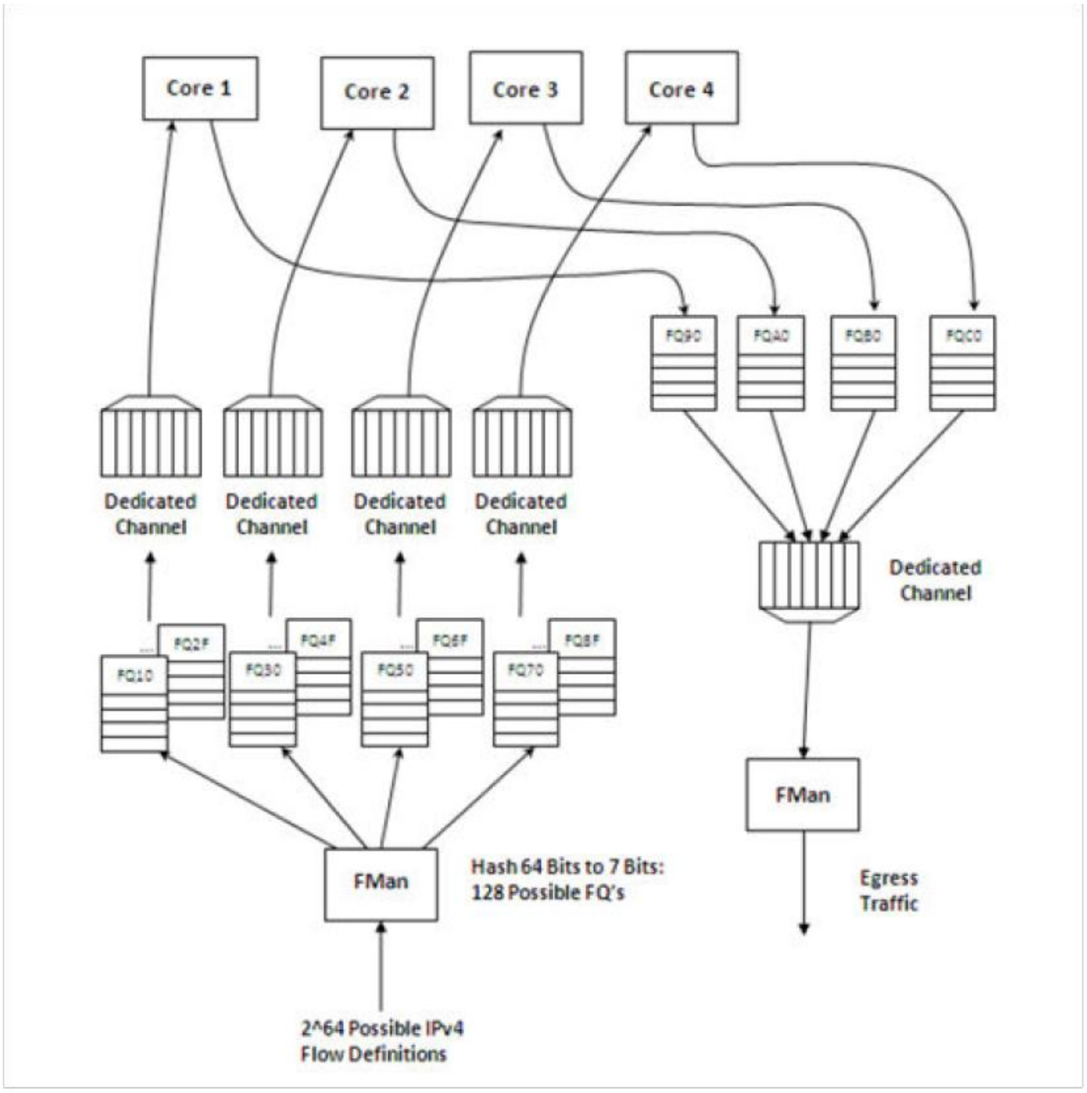


Figure 1 Using dedicated channel and hash in order preserved effective

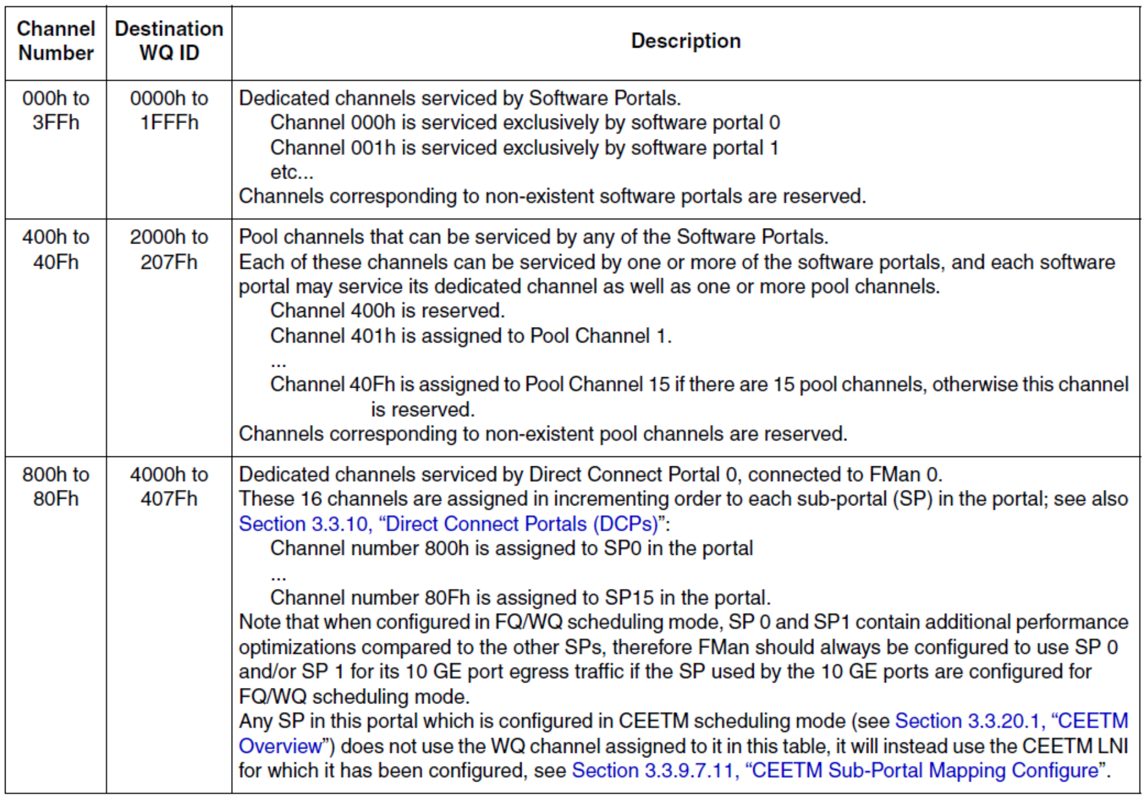
4. Work Queue Channel Assignment

In general, a WQ channel is dedicated to a specific portal, with the exception of the pool channels, which can be serviced by any of the software portals.

Each FQ in the system is configured (when the FQ is initialized or rescheduled) to be enqueued onto one of these WQs when its enqueue eligibility criteria are met.

The following Table describes a part of T4240 WQ channel assignments in QMAN.

Table 1 T4240 Work Queue Channel Assignments in QMAN



Channel Number	Destination WQ ID	Description
000h to 3FFh	0000h to 1FFFh	Dedicated channels serviced by Software Portals. Channel 000h is serviced exclusively by software portal 0 Channel 001h is serviced exclusively by software portal 1 etc... Channels corresponding to non-existent software portals are reserved.
400h to 40Fh	2000h to 207Fh	Pool channels that can be serviced by any of the Software Portals. Each of these channels can be serviced by one or more of the software portals, and each software portal may service its dedicated channel as well as one or more pool channels. Channel 400h is reserved. Channel 401h is assigned to Pool Channel 1. ... Channel 40Fh is assigned to Pool Channel 15 if there are 15 pool channels, otherwise this channel is reserved. Channels corresponding to non-existent pool channels are reserved.
800h to 80Fh	4000h to 407Fh	Dedicated channels serviced by Direct Connect Portal 0, connected to FMan 0. These 16 channels are assigned in incrementing order to each sub-portal (SP) in the portal; see also Section 3.3.10, "Direct Connect Portals (DCPs)": Channel number 800h is assigned to SP0 in the portal ... Channel number 80Fh is assigned to SP15 in the portal. Note that when configured in FQ/WQ scheduling mode, SP 0 and SP1 contain additional performance optimizations compared to the other SPs, therefore FMan should always be configured to use SP 0 and/or SP 1 for its 10 GE port egress traffic if the SP used by the 10 GE ports are configured for FQ/WQ scheduling mode. Any SP in this portal which is configured in CEETM scheduling mode (see Section 3.3.20.1, "CEETM Overview") does not use the WQ channel assigned to it in this table, it will instead use the CEETM LNI for which it has been configured, see Section 3.3.9.7.11, "CEETM Sub-Portal Mapping Configure".

5. Dedicated and Pool Channels Usage in Linux Kernel

Each interface used by default one pool channel across all software portals and also the dedicated channels of each CPU.

In the device tree, the property "fsl,qman-channel-id" is used to address the dedicated channel ID associated with the QMAN portal.

```
qportal1: qman-portal@4000 {
            cell-index = <0x1>;
            compatible = "fsl,qman-portal";
            reg = <0x4000 0x4000 0x1001000 0x1000>;
            interrupts = <106 0x2 0 0>;
            fsl,qman-channel-id = <0x1>;
    };
```

"Pool channel range" nodes have been added to specify pool channels that are available of dynamic allocation.

```
qman-pools@0 {
```

```
            compatible = "fsl,pool-channel-range";

            fsl,pool-channel-range = <0x401 0xf>;

        };
```

In Linux Kernel, PCD Frame queues in Linux Kernel use dedicated channels. The default and error FQs are assigned to the pool channel.

Please refer to dpa_fq_setup function in drivers/net/ethernet/freescale/dpa/dpaa_eth_common.c of Linux Kernel.

```
void dpa_fq_setup(struct dpa_priv_s *priv, const struct dpa_fq_cbs_t *fq_cbs,

                struct fm_port *tx_port)

{

   … …

        /* Prepare for PCD FQs init */

        for_each_cpu(cpu, affine_cpus)

                portals[num_portals++] = qman_affine_channel(cpu);

        if (num_portals == 0)

                dev_err(priv->net_dev->dev.parent,

                        "No Qman software (affine) channels found");

        pcd_fqid = (priv->mac_dev) ?

                DPAA_ETH_PCD_FQ_BASE(priv->mac_dev->res->start) : 0;


        /* Initialize each FQ in the list */

     list_for_each_entry(fq, &priv->dpa_fq_list, list) {

                switch (fq->fq_type) {

                case FQ_TYPE_RX_DEFAULT:

                        BUG_ON(!priv->mac_dev);

                        dpa_setup_ingress(priv, fq, &fq_cbs->rx_defq);

                        break;

                case FQ_TYPE_RX_ERROR:

                        BUG_ON(!priv->mac_dev);

                        dpa_setup_ingress(priv, fq, &fq_cbs->rx_errq);

                        break;

                case FQ_TYPE_RX_PCD:
```

```
            /* For MACless we can't have dynamic Rx queues */
            BUG_ON(!priv->mac_dev && !fq->fqid);
            dpa_setup_ingress(priv, fq, &fq_cbs->rx_defq);
            if (!fq->fqid)
                fq->fqid = pcd_fqid++;
            fq->channel = portals[portal_cnt];
            portal_cnt = (portal_cnt + 1) % num_portals;
            break;
        case FQ_TYPE_TX:
            dpa_setup_egress(priv, fq, tx_port,
                        &fq_cbs->egress_ern);
… …
}
```

6. Using Dedicated Channel in USDPAA

In USDPAA, pool channel mode is used by default. The network interface obtained via usdpaa_netcfg_acquire(), can be used to identify the QMan pool-channels that the network interfaces' Rx FQs are scheduled to. Using this information, the application can determine the pool-channels it wishes the initialized portal to dequeue from.

This section will discuss more about how to modify PPAC and USDPAA QMAN driver source code to implement using dedicated channel in USDPAA applications.

As described previously, dedicated channels are assigned to software portals by hardware, so in USDPAA qman driver, if a processor(core) is allocated a qman portal, it is binding to a specific dedicated channel automatically.

In order to use dedicated channel in USDPAA, the source code could be modified as the following.

Modify function "fsl_qman_portal_init" in drivers/qbman/qman_driver.c as the following:

```
int16_t dedicate_channel[24] = {0};
static int __init fsl_qman_portal_init(uint32_t index, int is_shared)
{
  ...
    pcfg.public_cfg.channel = map.channel;
    pcfg.public_cfg.pools = map.pools;
    pcfg.public_cfg.index = map.index;
```

```
        dedicate_channel[pcfg.public_cfg.cpu] = map.channel;
#if 1
        printf("Assigning dedicate channel 0x%x, pool channel 0x%x to thread/core %d\n",
                        0x0000ffff & map.channel, map.pools, pcfg.public_cfg.cpu);
#endif
   ...
}
```

Modify PPAC source code apps/ppac/main.c as the following:

Add function get_rxdc and use it in the function ppac_fq_pcd_init later.

```
extern uint16_t dedicate_channel[24];
int id = 0;
static inline u16 get_rxdc(uint32_t fqid)
{
        int core;
        uint16_t ch;
        //core = fqid % ncpus/2 + id * ncpus/2;
        core = fqid % ncpus;
        ch = dedicate_channel[core];
        printf("fqid = 0x%x, core = 0x%x, ch = 0x%x\n", fqid, core, ch);
        return ch;
}
```

Modify the function ppac_fq_pcd_init:

```
void ppac_fq_pcd_init(struct qman_fq *fq, u32 fqid,
                      u16 channel,
                      const struct qm_fqd_stashing *stashing,
                      int prefer_in_cache)
{
   ...

   if (dedicate)
```

```
        channel = get_rxdc(fqid);
        ret = qman_reserve_fqid(fqid);
        BUG_ON(ret);
        ret = qman_create_fq(fqid, QMAN_FQ_FLAG_NO_ENQUEUE, fq);
        BUG_ON(ret);
    ...
}
```