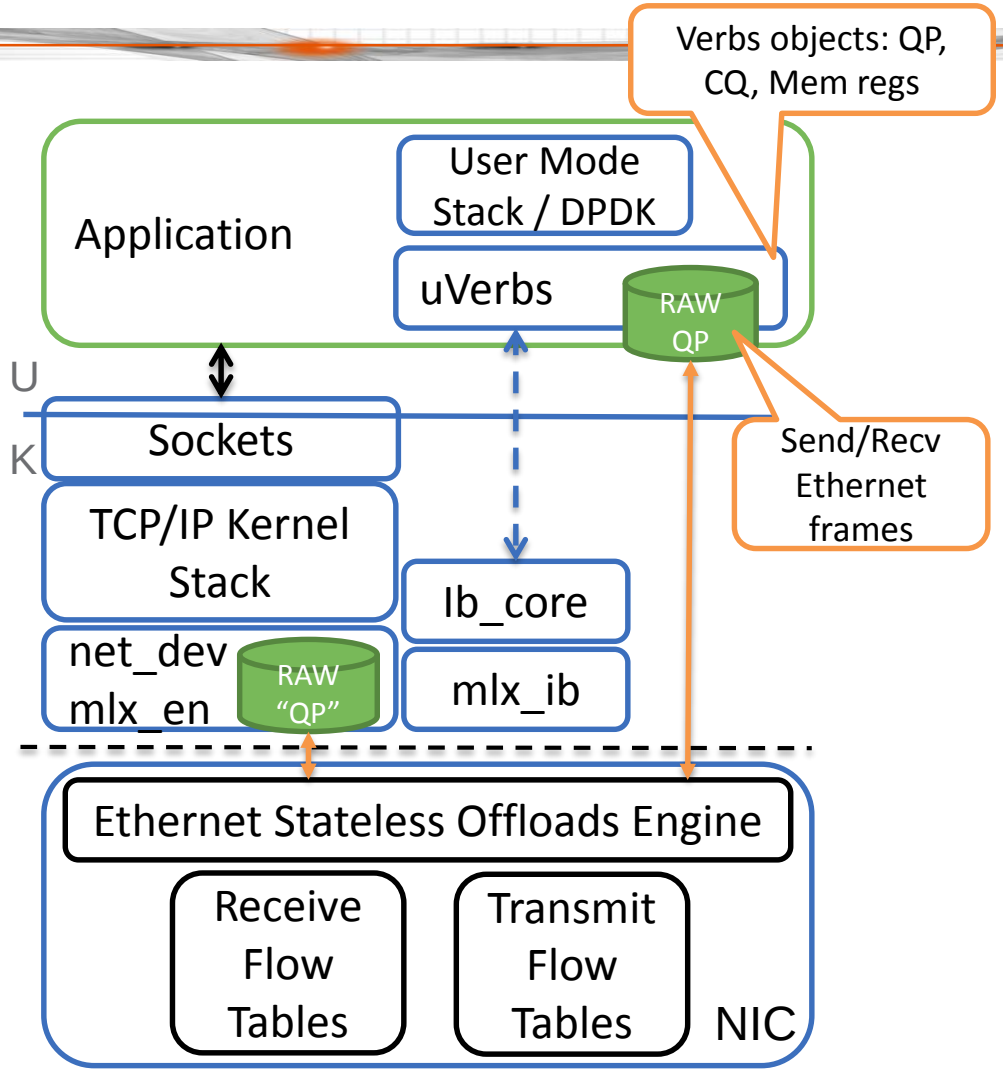
OPENFABRICS ALLIANCE

OVFWG – RSS Verbs

May 2016

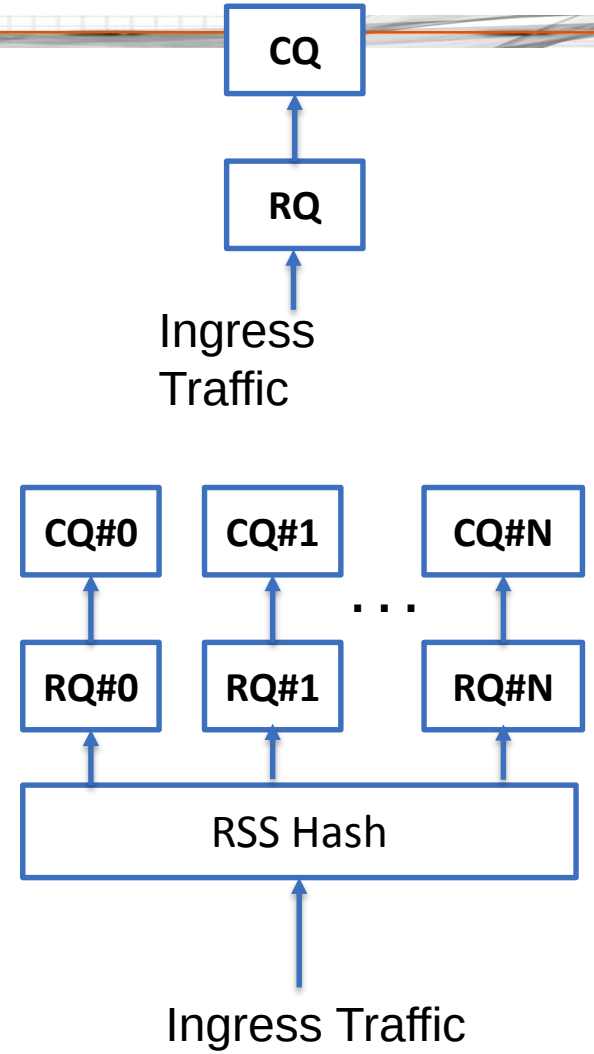
Current status – The RAW ETH QP

- Ibv_qp type: RAW_ETH
- Use mature verbs objects
  - QP, CQ, MR
- Pair of send and receive queues
  - Send queue to transmit raw packets - No implicit headers
  - Receive queue is steered according to flows classification
- Stateless Offloads Engine
  - Currently csum offload is supported
  - And Interrupt moderation (CQ moderation)
- Require privileged user
  - CAP_NET_RAW



Introduction

- Receive Side Scaling (RSS) technology enables spreading incoming traffic to multiple receive queues
- Each receive queue is associated with a completion queue
- Completion Queues (CQ) are bound to a CPU core
  - CQ is associated with interrupt vector and thus with CPU
    - For polling, user may run polling for each CQ from associated CPU
  - In NUMA systems, CQ may be allocated on close memory to associated CPU
- Spreading the receive queues to different CPU cores allows spreading receive workload of incoming traffic



Flow Overview

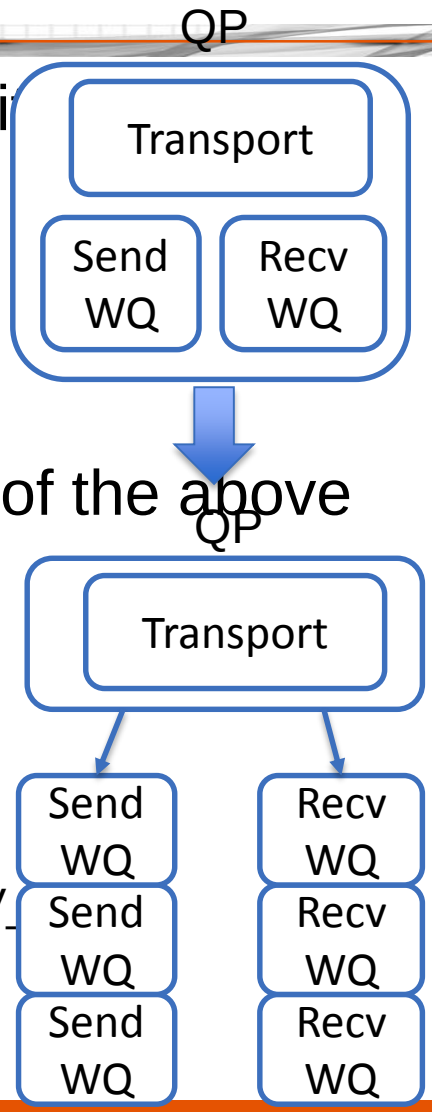
Classify first, distribute after

- Begin with classification
  - Using Steering (ibv_create_flow()) classify incoming traffic
  - Classification rules may be any of the packet L2/3/4 header attributes
    - e.g. TCP/UDP only traffic, IPv4 only traffic, ..
  - Classification result is transport object - QP
- Continue with spreading
  - Transport object (QPs) are responsible for spreading to the receive queues
  - QPs carry RSS spreading rules and receive queue indirection table
- RQs are associated with CQ
  - CQs are associated with CPU core

- Different traffic types can be subject to different spreading

Work Queue (WQ)

- Typically QPs (Queued Pairs) are created wi elements
  - Transmit and receive Transport
  - Receive Queue
    - Exception is QPs which are associated with SRQ
  - Send Queue
- Extend verbs to support separate allocation of the above 3 elements
  - Transport – ibv_qp with no RQ or SQ
    - Ibv_qp_type of IBV_QPT_RAW_ETH
      - Next will be UD QP type
    - New QP attribute: ibv_rx_hash_conf
  - Work Queue – ibv_wq
    - Can be of 2 types: IBV_RQ – Receive Queue and IBV_
    - We'll start with IBV_RQ definition



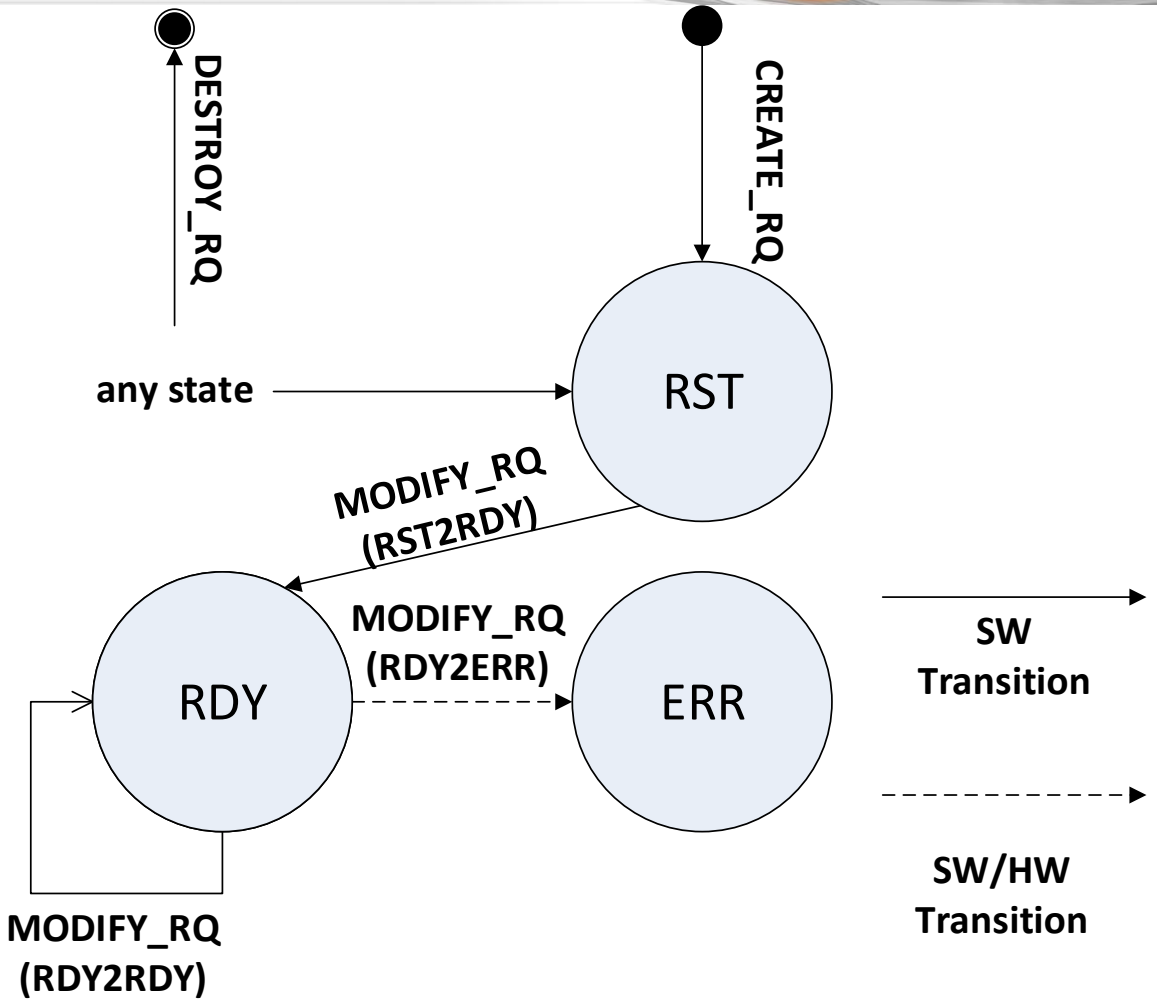
Work Queue (WQ) – Cont.

- **New object: Work Queue - ibv_wq**
- **Managed through following new calls:**
  - ibv_wq *ibv_create_wq(ibv_wq_init_attr)
  - ibv_modify_wq(ibv_wq , ibv_wq_attr)
  - ibv_destory_wq(ibv_wq)
  - ibv_post_wq_recv(ibv_wq, ibv_recv_wr)
- **Work Queues (ibv_wq) are associated with Completion Queue (ibv_cq)**
  - Multiple Work Queues may be mapped to same Completion Queue (many to one)

```
struct ibv_wq {
      struct ibv_context  *context;
      void                *wq_context;
      uint32_t            handle;
      struct    ibv_pd    *pd;
      struct    ibv_cq    *cq;
      /* SRQ handle if WQ is to be /
           associated with an SRQ, /
           otherwise NULL */
      struct    ibv_srq   *srq;
      uint32_t            wq_num;
      enum ibv_wq_state   state;
      enum ibv_wq_type    wq_type;
      uint32_t       comp_mask;
};
```

- Work Queues of type Receive Queue (IBV_RQ) may share receive pull
  - By associating many Work Queues to same Shared Receive Queue (the existing verbs ibv_srq object)
- QP (ibv_qp) can be created without internal Send and Receive Queues and associated with external Work Queue (ibv_wq)
- QP can be associated with multiple Work Queues of type Receive Queue
  - Through Receive Queue Indirection Table object

WQ of Type RQ – State Diagram



Receive Work Queue Indirection Table

- New object: Receive Work Queue Indirection Table – ibv_rwq_ind_table
- Managed through following new calls:
  - ibv_wq_ind_tbl *ibv_create_rwq_ind_table(ibv_rwq_ind_table_init_attr)
  - ibv_modify_rwq_ind_table(ibv_rwq_ind_table)
  - ibv_query_rwq_ind_table(ibv_rwq_ind_tbl, ibv_rwq_ind_table_attr)
  - ibv_destroy_rwq_ind_table(ibv_rwq_ind_tbl)
- QPs may be associated with an RQ Indirection Table
- Multiple QPs may be associated with same RQ Indirection Table

```
struct ibv_rwq_ind_table {
     struct ibv_context  *context;
     uint32_t            handle;
     int             ind_tbl_num;
     uint32_t        comp_mask;
};

/*
 * Receive Work Queue Indirection Table
attributes
*/
struct ibv_rwq_ind_table_init_attr {
     uint32_t        log_rwq_ind_tbl_size;
     struct ibv_wq   **rwq_ind_tbl;
     uint32_t        comp_mask;
};

/*
 * Receive Work Queue Indirection Table
attributes
*/
struct ibv_rwq_ind_table_attr {
     uint32_t        attr_mask;
     uint32_t        log_rwq_ind_tbl_size;
     struct ibv_wq   **rwq_ind_tbl;
     uint32_t        comp_mask;
};
```

Transport Object (QP)

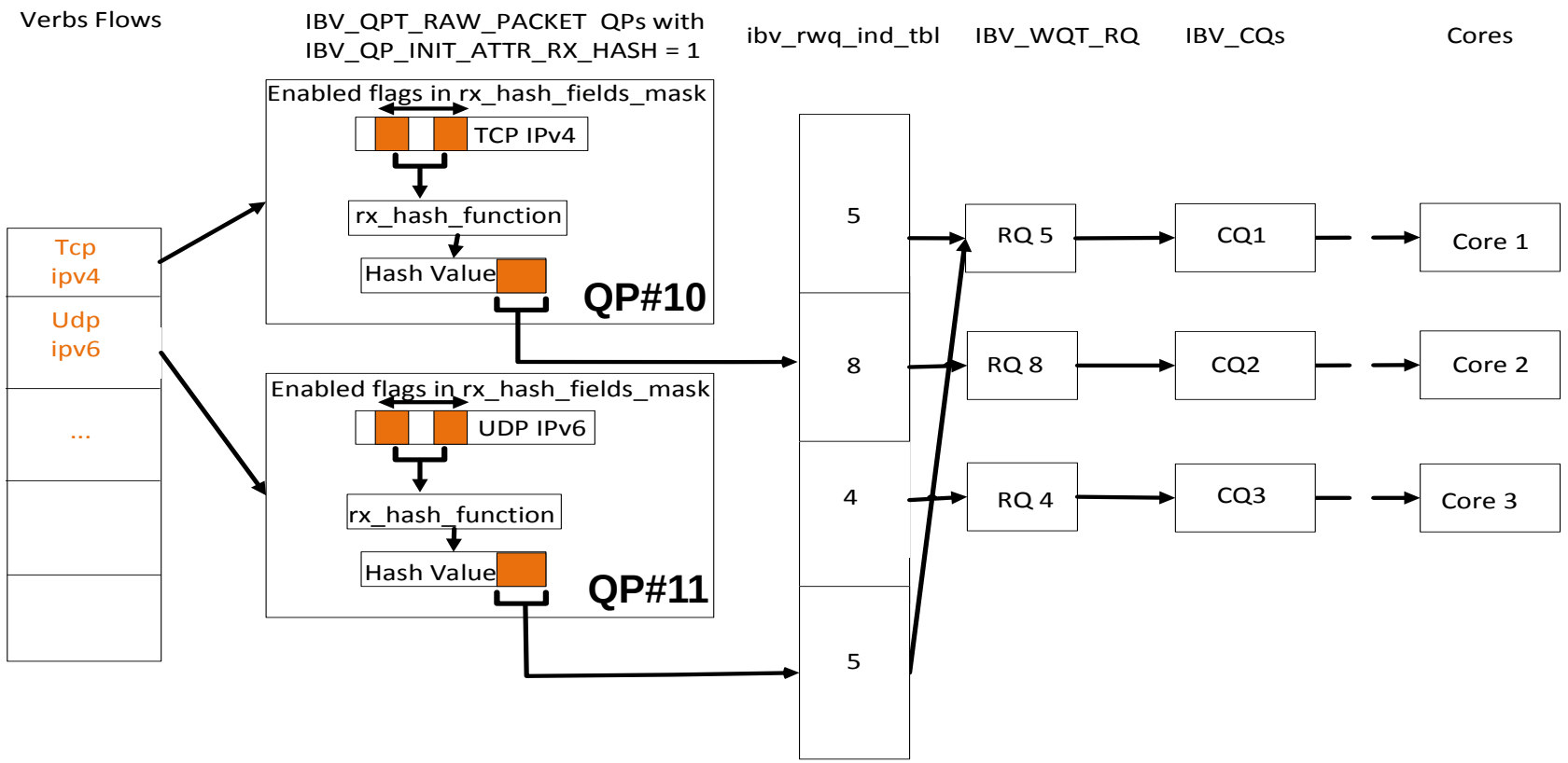
- "RSS" QP
  - QP attributes (ibv_qp_attr) now include RSS hash configuration attributes (ibv_rx_hash_conf)
  - QP is Stateless
  - QP's Send and Receive WQs parameters are invalid - QP has no internal work queues
  - Use ibv_post_wq_recv instead of ibv_post_recv
  - QP is connected to RQ Indirection Table
- On Receive, traffic is steered to the QP according to existing steering API
  - Ibv_create_flow()
- Following, matching RQ is chosen according to QPs hash calculation

```
struct ibv_rx_hash_conf {
      /* enum ibv_rx_hash_fnction */
      uint8_t    rx_hash_function;
      /* valid only for Toeplitz */
      uint8_t   *rx_hash_key;
      /* enum ibv_rx_hash_fields */
      uint64_t   rx_hash_fields_mask;
      struct ibv_rwq_ind_table   *rwq_ind_tbl;
};
/*
 RX Hash Function.
*/
enum ibv_rx_hash_function_flags {
      IBV_RX_HASH_FUNC_TOEPLTIZ   = 1 << 0,
      IBV_RX_HASH_FUNC_XOR        = 1 << 1
};
/*
 Field represented by the flag will be
 used in RSS Hash calculation.
*/
enum ibv_rx_hash_fields {
      IBV_RX_HASH_SRC_IPV4        = 1 << 0,
      IBV_RX_HASH_DST_IPV4        = 1 << 1,
      IBV_RX_HASH_SRC_IPV6        = 1 << 2,
      IBV_RX_HASH_DST_IPV6        = 1 << 3,
      IBV_RX_HASH_SRC_PORT_TCP    = 1 << 4,
      IBV_RX_HASH_DST_PORT_TCP    = 1 << 5,
      IBV_RX_HASH_SRC_PORT_UDP    = 1 << 6,
      IBV_RX_HASH_DST_PORT_UDP    = 1 << 7
};
```

Flow Diagram

Verbs Steering Classifies the traffic

IBV_QPT_RAW_PACKET QPs distributes traffic type between RQs/Cores



Next

- IPoIB UD QP type
  - “RSS” UD QP is connected to RQ Indirection Table
  - RSS UD QP to continue to manage UD transport attributes: pkey, qkey checks…
  - Single wire QPN for all getting to all the QPs Receive Queues
- Transmit Side Scaling (TSS)
  - As in RSS, QP is stateless, Send and Receive work queues attributes are invalide
  - Use ibv_post_wq_send instead of ibv_post_send
  - For IPoIB UD QP:
    - Manage UD transport properties: pkey, qkey…
    - Use single source QPN in DETH wire protocol header for all Send WQ which is the “TSS” UD QP
  - The same QP may be used for both “RSS” and “TSS” operations

Thank You