13th ANNUAL WORKSHOP 2017

IPOIB ACCELERATION

Tzahi Oved, Rony Efraim

Mellanox Technologies

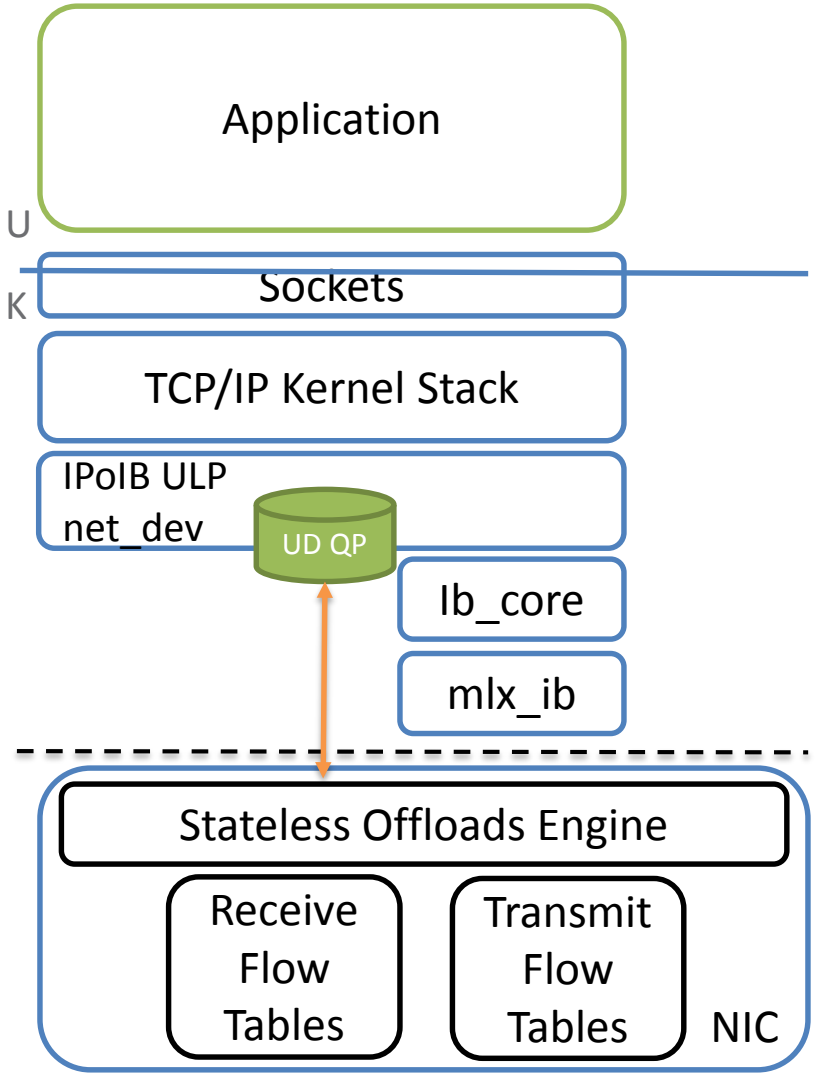
[March, 2017]

AGENDA

- IPoIB as ULP
- Acceleration Goals
- Architecture
- Why vendor driver
- Packet Flows
- Kernel verbs API update – initial proposal
- Kernel verbs API update – current proposal
- ULP event handling
- Performance
- Summary

IPOIB AS ULP

- **Encapsulate IP frames over IB transport**
- **IPoIB net_device is implemented as Upper Layer Protocol (ULP)**
- **Underlying API is the Kernel Verbs API**
- **Supports both unconnected (UD) and Connected (RC) modes**
- **Use SA, CM and MAD services**
  - Path query
  - Multicast membership (Join/Leave)
  - Communication Manager for IPoIB RC
- **Only CSUM, LSO offloads are currently supported**
- **Any feature must be supported through kernel Verbs API**
  - Stateless offload
  - Device management and diagnostics (ethtool)

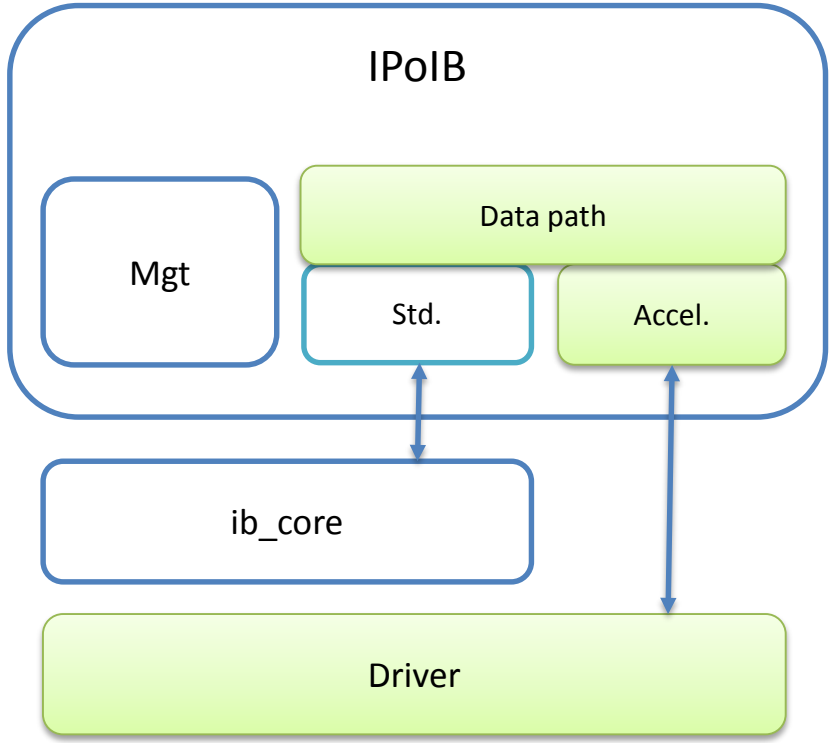


ACCELERATION GOALS

- **Allow vendors to optimize IPoIB data path**
  - Stateless offloads
    - LRO, aRFS, RSS, TSS, ..
    - Multi-queue support – all queues to share same transport
  - Interrupt moderation
  - Tunneling offloads
  - Multi partitions optimizations
    - Share send/recv Work Queues across multiple pkeys
  - Vendor specific optimizations
    - Work queue processing
- **net_device management**
  - ethtool_ops
- **Avoid bloating kAPI**
- **Leverage IPoIB ULP code as much as possible**
- **Support UD IPoIB mode only**
  - No plans to support IPoIB Connected Mode (RC)
- **Generic ULP code – support “Legacy” mode**
  - No accelerated mode support
  - ***Same ULP fully supports non accelerated providers***

ARCHITECTURE

- **Separate functionality into management and data paths**
- **Management requirements (IPoIB)**
  - Interface registration
  - Multicast management
  - Event processing
    - LID change, SM LID change
  - Logical link state
  - Path resolution and path cache management
  - Address resolution (IB ARP)
- **Data path requirements (Driver)**
  - Multi-queue support
  - Buffering
  - Receiving packets
  - Sending packets



WHY VENDOR DRIVER

- **Allow HW specific optimizations for data path**
  - Not bound by Verbs API semantics
- **HW agnostic API requires HW agnostic call parameters**
  - Requires extra conversion from HW structures (WQE, CQE) to API intermediate structures and then to application specific structures
- **HW agnostic API requires HW specific provider callbacks**
  - Using function pointers to register provider code costs pointer dereference
  - Using functions requires instruction memory pre-emption
- **Minimize parameters size**
  - Using general WQE (ibv_send_wr, ibv_recv_wr), CQE (ibv_wc) access calls results is large and redundant data structures allocation and reference
  - Need to use only necessary data and as near as possible, possible in the same cache line
- **Instruction memory cache utilization**
  - Generic code results in large functions which require more cache lines utilization
- **Compile time code optimizations**
  - Using function calls for HW structures access blocks the compiler to perform code optimizations for driver+HW access code

PACKET FLOWS

- **TX**
  - ULP
    - Resolve L2 address (IB ARP)
    - Resolve address path and pass AH attributes
  - Vendor Driver
    - Select TX queue
    - Build send WQE out of the send skb and post_send
    - Transmit CQE processing
    - Free skb
- **RX**
  - Vendor Driver
    - Receive CQE processing
    - Receive WQ processing
      - NAPI
      - Call netif_receive_skb
    - skb allocation and post_recv

IPOIB ULP MULTICAST FLOW

- **Join IPoIB “Broadcast” MGID**
  - On ULP initialization join default multicast MGID per IPoIB device pkey
  - Result will include the IPoIB qkey to be used with IPoIB QP
- **On MC send**
  - ULP to join corresponding MC group towards the SM through Multicast Join request
  - Use Send Only member attribute
  - Follow with Attach Multicast local request to attach the corresponding MGID to IPoIB QP
  - According to aging scheme call Multicast Leave
- **On MC receive**
  - Net_device->mc_list is updated with new MC address
  - ULP to join corresponding MC group towards the SM through Multicast Join request
  - Use full member attribute
  - On revoke of the MC address from mc_list call corresponging Multicast Leave
- **Join/Leave once per port**

KERNEL VERBS API UPDATE – INITIAL PROPOSAL

- **Extend Kernel Verbs to allow query of struct ipoib_ops**
  - Dependent on per vendor support
  - Allows each vendor to export it's own calls
- **Struct ipoib_ops to expose only selected vendor accelerated calls**

```
struct ipoib_ops {
     struct net_device * (*create_netdev)(struct ib_device *hca, const char *name);
     int (*update_netdev_settings) (struct net_device *dev);
     int (*ib_dev_init)(struct net_device *dev, uint *qp_num);

     void (*ib_dev_open) (struct net_device *dev);
     void (*ib_dev_stop) (struct net_device *dev);

     int (*set_qkey)(struct net_device *dev)
     int (*attach_mcast)(struct net_device *dev , union ib_gid *gid, u16 lid, int
     set_qkey);
     int (*dettach_mcast)(struct net_device *dev , union ib_gid *gid, u16 lid);
     int (*send) (struct net_device *dev, struct sk_buff *skb,
                  struct ib_ah *address, u32 qpn);

     void (*ib_dev_cleanup)(struct net_device *dev);
};
```

KERNEL VERBS API UPDATE – CURRENT PROPOSAL

- **Kernel verbs to support `alloc_rdma_netdev()` call to get vendor initialized struct net_device**
- **Each vendor can implement it's own optimized net_dev operations (net_dev ndo)**
  - Dependent on per vendor support
  - Allows each vendor to export it's own callbacks
- **rdma_netdev to include ipoib multicast management calls**

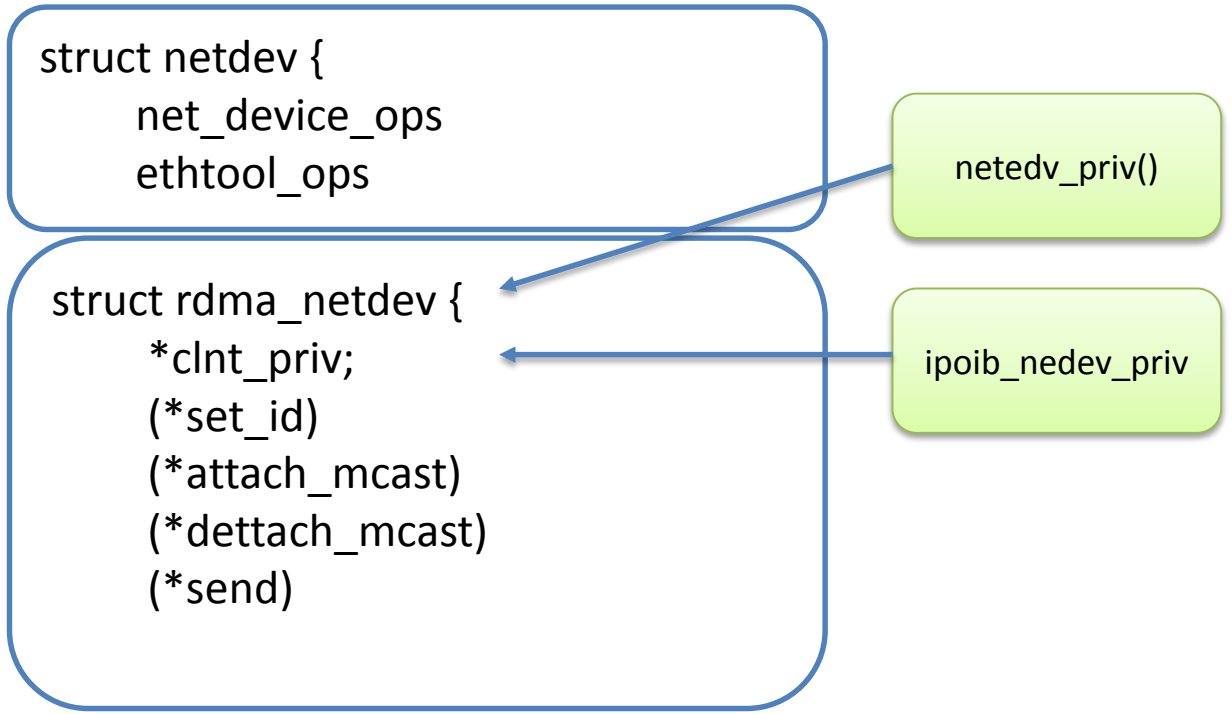
```
New struct ib_device call:
    struct net_device *(*alloc_rdma_netdev)(
                        struct ib_device *device,
                        u8 port_num,
                        enum rdma_netdev_t type,
                        const char *name,
                        unsigned char name_assign_type,
                        void (*setup)(struct net_device *));
    void (*free_rdma_netdev)(struct net_device *netdev);

struct rdma_netdev {
    void *clnt_priv;

    /* control functions */
    void (*set_id)(struct net_device *netdev, int id);
    int (*attach_mcast)(struct net_device *dev , union ib_gid
                        *gid, u16 lid, int set_qkey, u32 qkey);
    int (*dettach_mcast)(struct net_device *dev , union ib_gid
                         *gid, u16 lid);
    int (*send) (struct net_device *dev, struct sk_buff *skb,
                 struct ib_ah *address, u32 qpn);
};
```

IPOIB ULP INITIALIZATION FLOW

- **On IPoIB ULP initialization call alloc_rdma_netdev()**
  - Verify ib_device provider support for enhanced mode
  - Extract enhanced mode provider calls
    - Extract rdma_netdev from netdev_priv()
- **ULP may update net_dev ndo and other parameters/function in struct net_device**
- **Use returned struct net_device to register standard OS network I/F**
  - register_netdevice()



EVENT HANDLING

Event	ULP	Driver
Interface up/down	V	
Physical port up/down	V	
Carrier up/down	V	
Client Reregister	V	
PKey change	V	
MTU change	V	
Tx timeout	V	
Select queue		V
Tx queue management (netif_wake/stop_queue, polling)		V
Rx queue management (NAPI, polling, buffer allocation)		V

PERFORMANCE

- **Standard IPoIB**
  - BW max at 36Gbps
  - CPU bound
- **Accelerated IPoIB BW**
  - Saturate line rate of 100G EDR link
    - With >=4 streams
  - Scales with number of cores

SUMMARY

- **Using single call extension to ib_device to extract vendor specific enhanced calls**
- **Reuse the good old struct net_device to allow most flexibility for vendor calls**
- **Net additions to struct net_device can transparently be overloaded with new ib_device providers**
- **Keep ULP code generic - allow backward compatibility of the same ULP to older provider**
  - Legacy support for “non enhanced” providers
- **Allow optimized data path while reuse and share existing control path**

13th ANNUAL WORKSHOP 2017

THANK YOU

Tzahi Oved & Rony Efraim

COMPANY Mellanox

[LOGO HERE]