SoftRoCE Update

Liran Liss
Mellanox Technologies

Agenda

- Introduction
- Initial development
- New(!) RXE community project
- Upcoming tasks
  - Address community feedback
  - Lossless networking
  - RoCEv2 UDP model
  - Optimize buffering and 0-copy
  - Minimize execution contexts
- Status
- Summary

Introduction

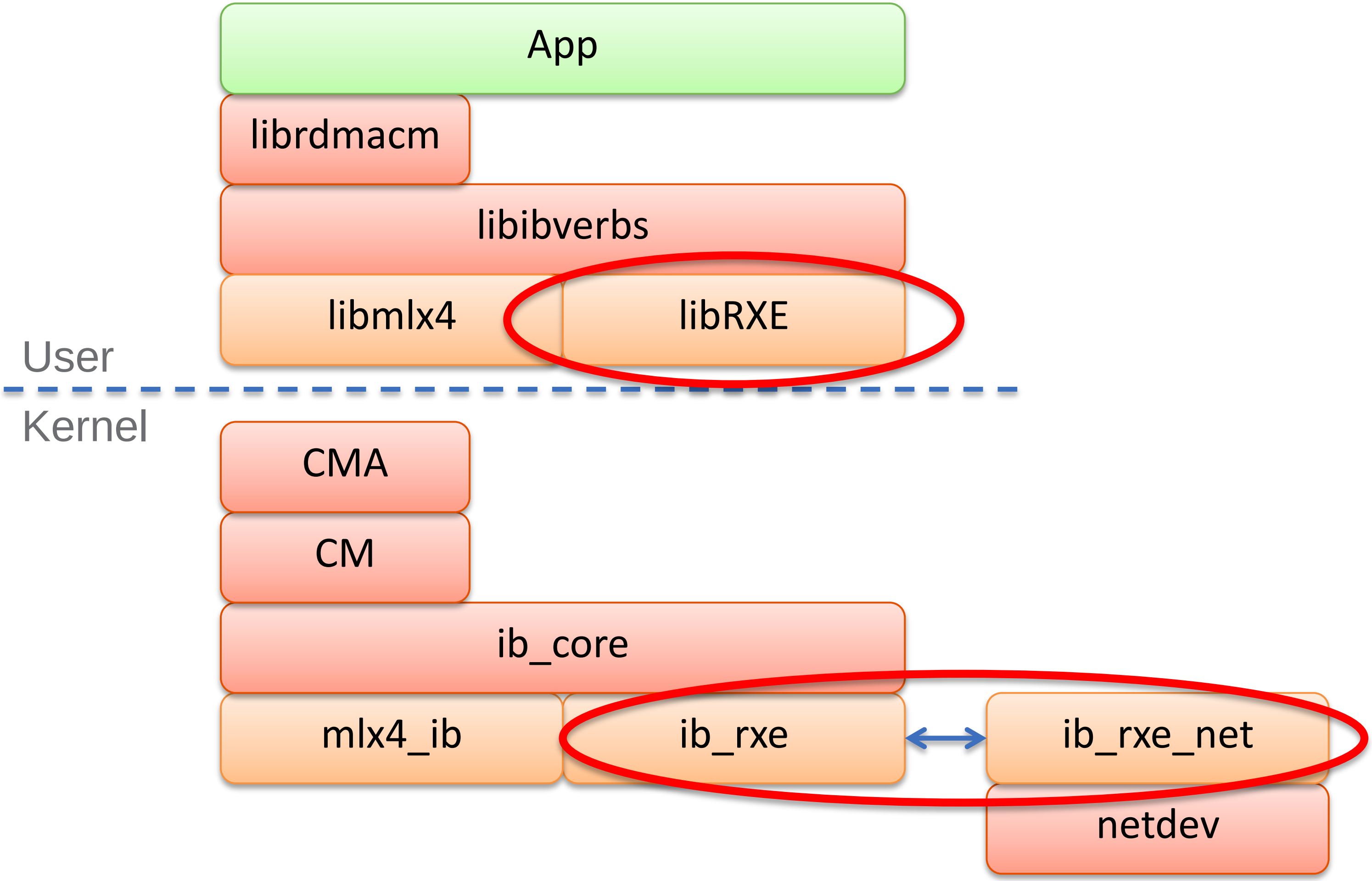
- SoftRoCE is a software RoCE device
- Leverages the same efficiency characteristics of the RoCE transport
  - Transaction-oriented wire-protocol
  - Decouples congestion control from reliability
  - Asynchronous IO directly to application buffers
- A device instance may be bound to any NIC
  - Lossless network required (global pause or PFC)
  - ETS recommended
- Use cases
  - Pervasive RDMA (e.g., running RoCE on my laptop)
  - Asymmetric deployments
    - E.g., multiple SW clients accessing a high-performance HW server

Initial Development

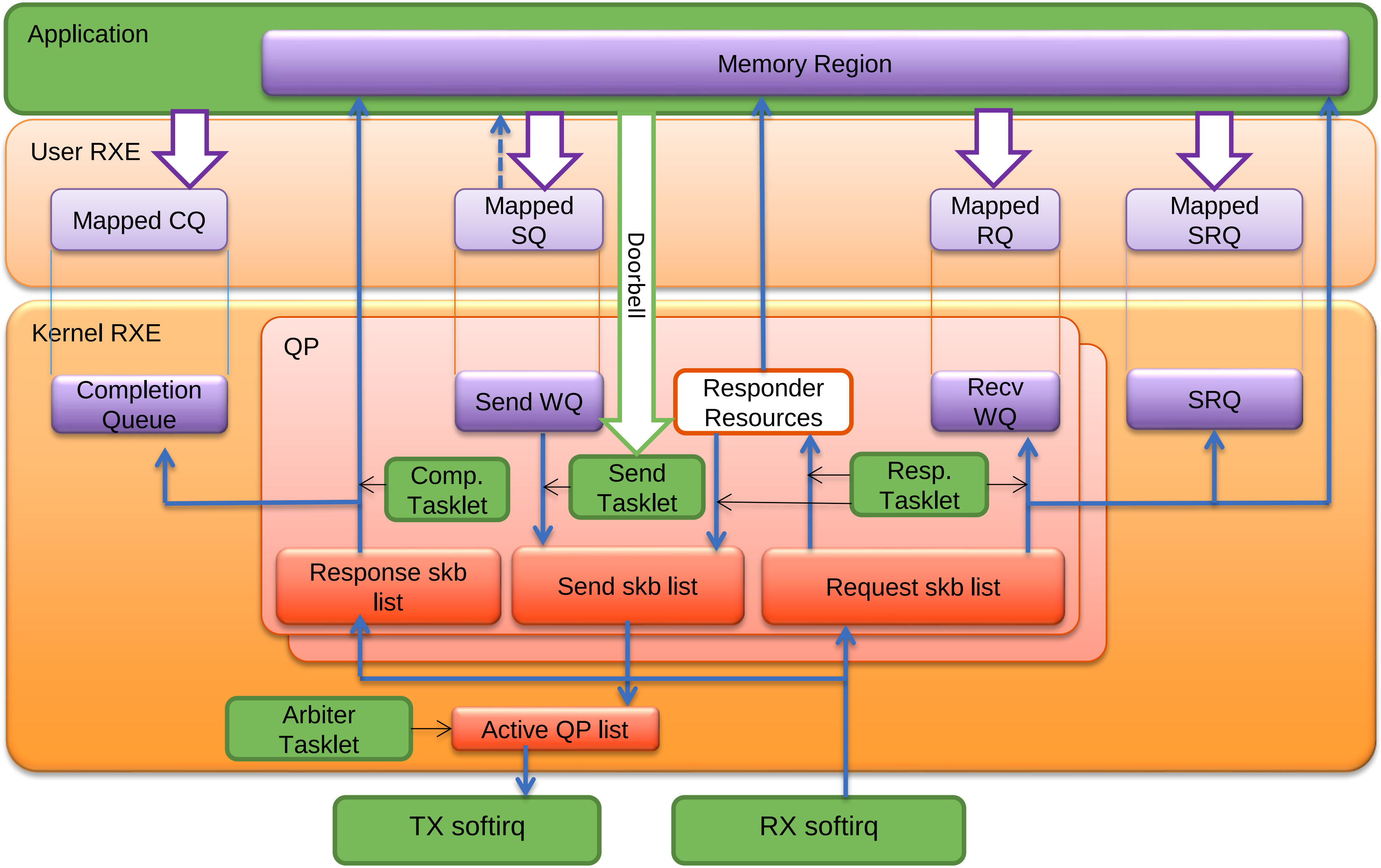
- Seminal driver (RXE) from System Fabric Works
  - Complete
    - PDs, QPs, CQs, SRQs
    - RC, UD, and UC transports
  - User-space and kernel drivers

- Solid design principles
  - Kernel transport
    - Support both kernel ULPs and applications
    - Asynchronous progress
  - Kernel-allocate / user-mapped queues
    - Complete user queue processing
    - System call used only for ringing doorbell
  - Pinned memory regions
    - Data may be copied into / out-of buffers within non-sleepable contexts

RXE Driver Stack



RXE Data Path



Renewing Development

- New RXE community project started
  - https://github.com/SoftRoCE
  - Kernel + user-driver library
  - Contributors from SFW, Mellanox, and IBM

- Goals
  - Get SoftRoCE ready from prime time
    - Rigorous testing and validation in mixed environments
  - Linux upstream submission
  - Continue development
    - New features
    - Improve performance

Upcoming Tasks

- Complete initial submission
  - Address issues raised by initial submission rounds
    - Simplify device management and driver stack
    - Code cleanups (e.g., CRC, module parameters)
    - Remove redundant compilation-dependent code paths
    - Intrusive netdev-queuing heuristics
  - Lossless networking
  - RoCEv2 UDP model
    - Build upon recent ib_core GID management patches

- Enhancements
  - Minimize execution contexts
  - Optimize buffering and 0-copy
  - Doorbell system-call avoidance
  - Improve arbitration

Lossless Networking

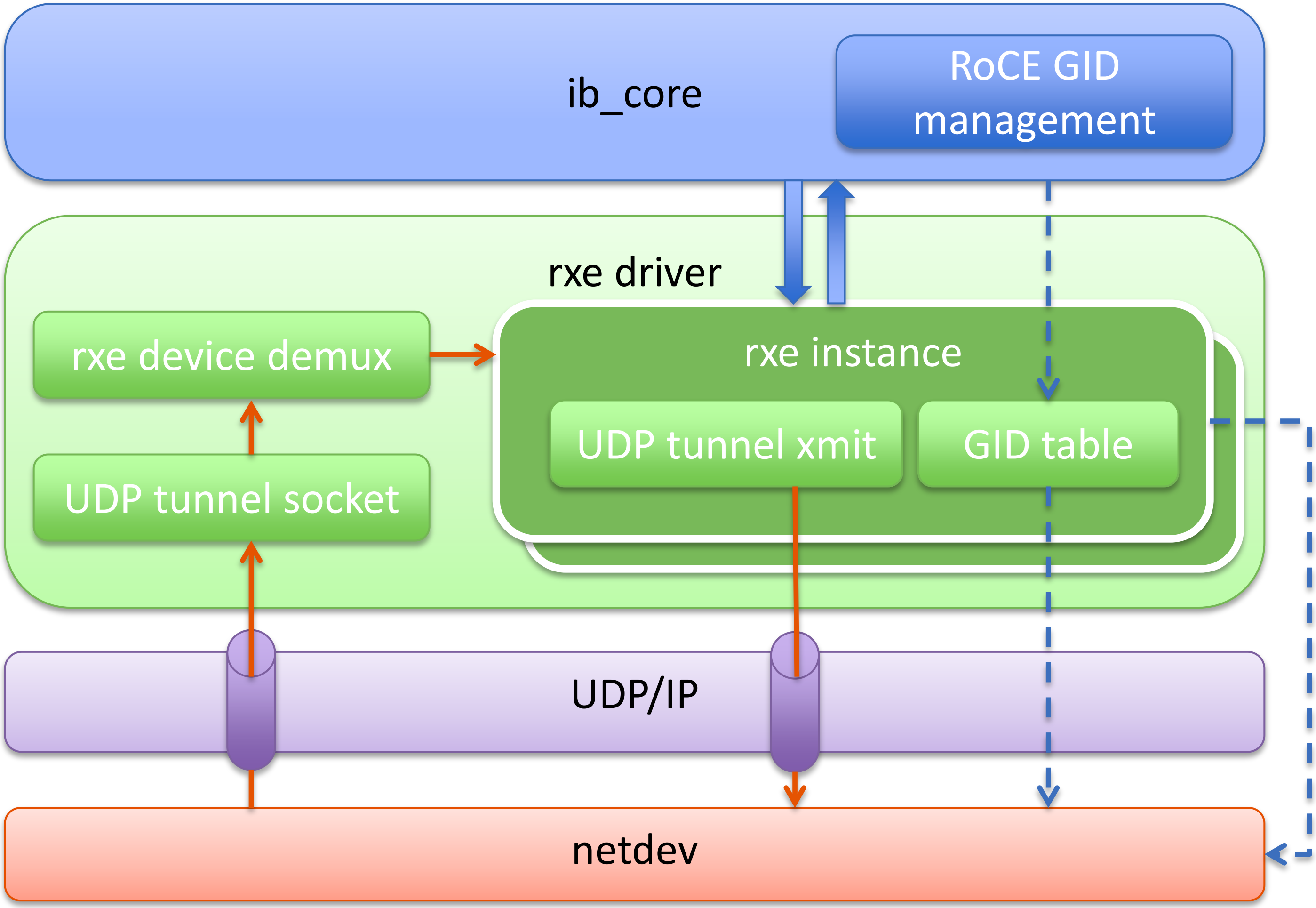
- Transmission
  - Respond to “queue full” indication from dev_queue_xmit()
    - Re-queue packet for later transmission
  - Resume transmission later
    - Start with timer backoffs (e.g., FCoE)
    - Future work: use skb lifetime accounting to trigger progress
- Reception
  - Typically not a problem
  - Introduce new netdev capability to pause network upon empty queues
- Future work: lossless virtual networking
  - virt_io, bridge, OVS

RoCEv2 UDP Model

Criteria \ Model	UDP application	UDP tunneling
RDMA namespace	Single OS-wide	Per-device
Relation to HW RoCE devices	Incompatible	Compatible
Lossless network support	Hard	Easy
Performance	Medium	High
Address management	Linux networking	RDMA core
Packet generation	Standard UPD socket	UDP tunneling APIs

- Preferred approach: UDP-tunneling
  - Seamless HW/SW RoCE interoperability
  - High performance – the ultimate goal
  - Reuse existing kernel UDP-tunneling practices (e.g., VXLAN driver)

UDP-Tunnel Architecture



UDP-tunnel Data Path

- Tx
  - udp_tunnel_xmit_skb() / upd_tunnel6_xmit_skb()
  - Qdisc full status reported by return value

- Rx
  - setup_udp_tunnel_sock() / udp_tunnel_sock_release()
    - Establishes listening socket
      - Bound to IP_ADDR_ANY
    - Receiving netdev determined according to skb->dev
  - No context switching
    - Callback passed to setup_udp_tunnel_sock()

Reducing Execution Contexts

- Remove ***all*** execution contexts
  - Per QP tasklets
    - Sender
    - Responder
    - Completer
  - Per device tasklets
    - Arbiter
- Progress done within
  - Doorbell system call
  - NAPI processing
  - Qdisc start indication
    - Implicitly by skb destructors
    - Explicitly by new Qdisc callbacks
  - Similar to TCP fast-path…

- Handle arbitration by queuing
  - QP selection, Sender vs. responder, etc,
  - Similar to Qdisc…

Optimize Buffering

- 0-copy for ***all*** transmitted packets
  - Leverage the fact that all MRs are pinned
  - Applies to
    - Send / RDMA-W packets
    - RDMA-R response packets
  - Generate completions when skbs are destroyed
    - In contrast to current Qdisc->enqueue()
  - Note: we still go over all the data for CRC calculation

- Efficient 1-copy for ***all*** received packets
  - Remove ***completely*** any skb re-queuing
  - Scatter data to memory directly from NAPI context
    - Just like TCP fastpath
    - Directly related to HW NIC back-pressure

Status

- RXE successfully passes regression tests
- Initial lossless backpressure testing in the face of congestion look promising
- Tasks for next submission round nearly complete

Summary

- SoftRoCE is an important building block in the RDMA eco-system
- Provides an extremely high-performance SW transport
  - For both user-space and kernel applications
- New community project has been started
  - https://github.com/SoftRoCE
- Call for action
  - Help us make SoftRoCE great!

Thank You

OPENFABRICS
ALLIANCE

10TH ANNUAL
INTERNATIONAL
DEVELOPER
WORKSHOP