RDMA Programming

- #include <rdma/rdma_cma.h>
- #include <infiniband/verbs.h>

RDMA Workflow

- Connection Setup
- Memory Registration
- Posting Work Requests (WRs)
- Completion Handling

Connection Setup

- Connection Manager (CM): Manages connection setup and teardown

  Open a channel to report asynchronous connection events - rdma_create_event_channel()

  Create connection identifier (like socket) - rdma_create_id(channel, &cm_id, context, RDMA_PS_TCP)

- Client Resolve address and route: Map IP/port → RDMA device and discover route

  Resolve address - rdma_resolve_addr(cm_id, src, dst, timeout)

  Process CM event (RDMA_CM_EVENT_ADDR_RESOLVED) - rdma_get_cm_event(echannel, cm_event), rdma_ack_cm_event(cm_event)

  Resolve route - rdma_resolve_route(cm_id, timeout)

  Process CM event (RDMA_CM_EVENT_ROUTE_RESOLVED)

- Server bind address - rdma_bind_addr(cm_id, server_addr)
- Server listen - rdma_listen(cm_server_id, num)
- Wait and process CM event RDMA_CM_EVNET_CONNECT_REQUEST

  This gives client id (like TCP conn) from cm_event→id
- Ack CM event

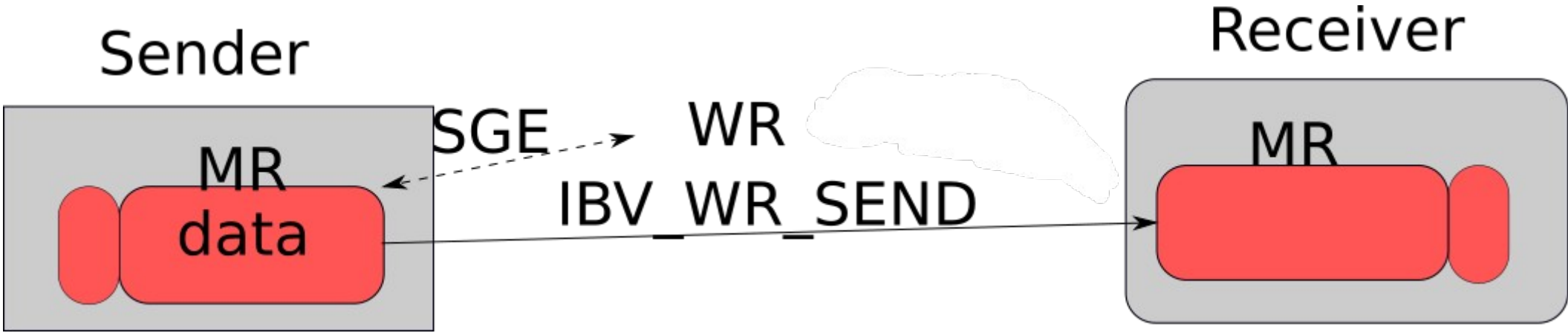
- Allocate Protection Domain (PD): memory region protections for access control
  - ibv_alloc_pd(cm_id→verbs)
- Create Completioin Queue (CQ): where completion events (CQEs) are reported after WRs finish. Use a completion channel to get notifications.
  - ibv_create_comp_channel(cm_id->verbs)
  - ibv_create_cq(cm_id->verbs, CQ_CAPACITY, context, comp_channel, 0)
  - ibv_req_notify_cq(cq, 0) //tell CQ that we want notification

- Create Queue Pair (QP): Two queues per endpoint: Send Queue (SQ) and Receive Queue (RQ). The core send/receive structure
  - Set ibv_qp_init_attr
  - rdma_create_qp(cm_id, pd, &qp_init_attr)

Connection

- Set parameters – rdma_conn_param
- On server side accept - rdma_accept(cm_id, &conn_param)
- On client side connect - rdma_connect(cm_id, &conn_param)

Memory Region (MR)
Work Request (WR)
Scatter-Gather Entry (SGE)



Memory Registration

- A registered memory buffer the RNIC can read/write
  - Allocate buffer – malloc()/mmap()
  - Register MR: Tell RNIC it can access this memory - ibv_reg_mr(pd, buf, len, flags)

- SGEs tells the RNIC which buffer in memory the upcoming Work Request (WR) should use.

  struct ibv_sge sge;

  sge.addr = (uintptr_t)mr->addr; // address of buffer

  sge.length = mr->length; // size of buffer

  sge.lkey = mr->lkey; // local access key

- Can have multiple SGEs to describe disjoint buffers

```
struct ibv_send_wr send_wr = {0};
struct ibv_send_wr *bad_wr = NULL;
```

send_wr.sg_list = &sge; // pointer to array of SGEs

send_wr.num_sge = 1; // number of SGEs

send_wr.opcode = IBV_WR_SEND; // or IBV_WR_RDMA_WRITE, IBV_WR_RDMA_READ, etc.

send_wr.send_flags = IBV_SEND_SIGNALED; // ask for CQE

ibv_post_send(qp, &send_wr, &bad_wr);

```
struct ibv_recv_wr recv_wr = {0}, *bad_recv = NULL;

recv_wr.sg_list = &sge;

recv_wr.num_sge = 1;

ret = ibv_post_recv(qp, &recv_wr, &bad_recv);
```


CQE
SEND
RECV
RECV
RECV
CQE
C
Q
C
Q


C
Q
RECV
RECV
C
Q

- CQE – when you know the WR completes
  - Get notification on CQ channel: there’s work completion (WC) in CQ – ibv_get_cq_event(comp_channel,&cq_ptr, &context);
  - Request for more notification - ibv_req_notify_cq(cq_ptr, 0)
  - Poll WC – ibv_poll_cq(cq_ptr, wc_num, wc)
  - Ack CQ events – ibv_ack_cq_events(cq_ptr, 1)

wr_id

- When having many WRs, which WR is a CQE for?
  - Use wr_id
  - Set recv_wr.wr_id
  - Check wc.wr_id when receiving CQE later

Notes

- When SEND, make sure there’s at least one posted RECV WR on the other end
- The Receive Queue is FIFO: The first RECV WR you post will be the first one consumed by the next incoming SEND message.
- You can post a RECV again after using it.

Write/Read remote memory directly

- wr.opcode: IBV_WR_RDMA_WRITE/IBV_WR_RDMA_READ
- Needs remote raddr + rkey (send by receiver)

```
wr.wr.rdma.remote_addr   = remote_base + off;
wr.wr.rdma.rkey          = remote_rkey;
```

Disconnect

- Client disconnect - rdma_disconnect(cm_id)
- Server wait for disconnect CM event (and ack it) – RDMA_CM_EVENT_DISCONNECTED
- Client also wait for disconnect CM event

Clean up

- Destroy QP – rdma_destroy_qp(cm_id)
- Destroy ID – rdma_destroy_id(cm_id)
- Destroy CQ – ibv_destroy_cq(cq)
- Destroy completion channel – ibv_destroy_comp_channel(channel)
- Destroy other event channel – rdma_destroy_event_channel(channel)
- Destroy PD - ibv_dealloc_pd(pd)

Usefull Resources

- https://man7.org/linux/man-pages/index.html

Lab 3

- Sender (provided framework, your job), Receiver (provided)
- Goal: send messages, send file

Lab 3 Workflow

- Sender setup and initiates connection
- Sender sends file length + hash to receiver
- Receiver sends (addr, len, rkey) of a MR back
- Sender writes the file to that MR, sends a MSG_DONE when completes
- Receiver compares hash and sends back good/bad
- Sender sends ID
- Receiver sends back secret key