RDMA Programming

Course Description: The goal of this programming course is to provide developers with the knowledge and experience they need to be able to implement parallel programming using Remote Direct Memory Access (RDMA) verbs for reliable connections (RC). The course provides the following materials to the developers to help them rapidly understand and create kernel modules that utilize RDMA kernel verbs. The modules will be usable over any RDMA transport such as InfiniBand (IB), iWARP, RoCE or RoCEv2.

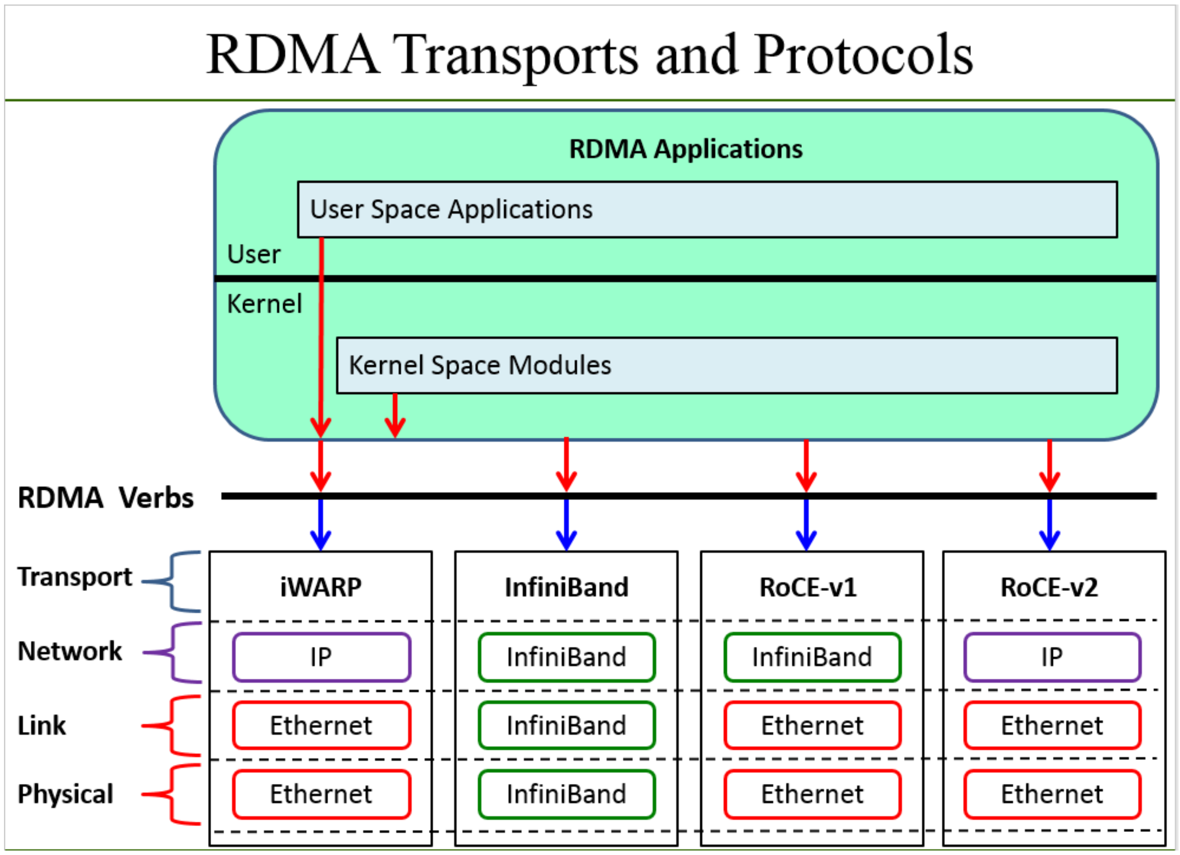
1) Diagrams to illustrate the overall RDMA architecture, transports & protocols and protocol layers.
2) Hundreds of slides in PDF format which explain all of the critical concepts and how to implement them.
3) Many example programs which illustrate all the concepts being presented in the slides. These examples can be used as fully functional modules to develop and implement your company's projects.

The example programs are structured so that developers learn to update and expand them to solve various problems being discussed in the RDMA training session.

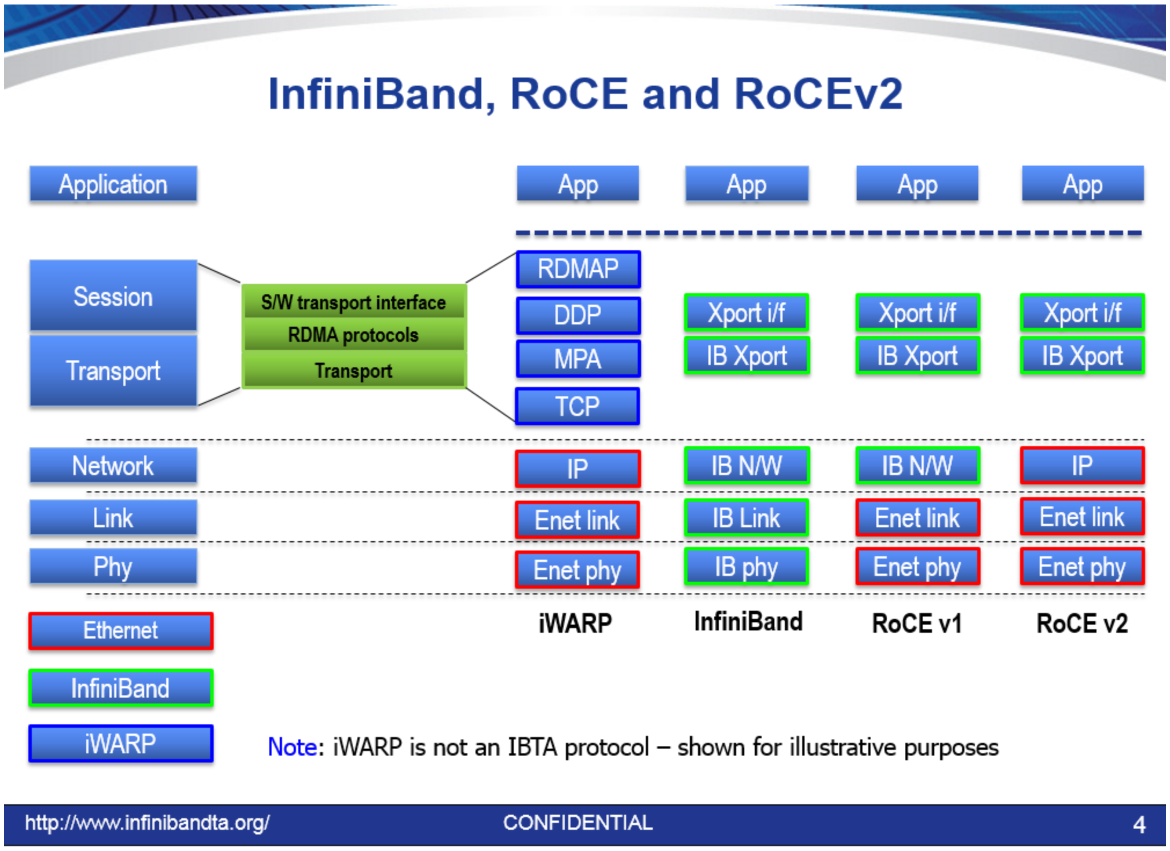
Course pre-requisites
1) Knowledge of C and Linux programming
2) Knowledge of kernel modules
3) User-level programming course

The following are some of the topics discussed in the course:
1) RDMA Overview
    a) Definitions and benefits
    b) Transport Services
    c) Transport Semantics
    d) RDMA verbs and various APIs
2) RDMA Transports and Protocols
    a) An explanation of the various RDMA stacks and protocols
3) Kernel Module
    a) What it is
    b) Components of a module
    c) Functions a module needs
    d) Building a module
    e) Loading/unloading a module
    f) The module file system
    g) Module dependencies, and building a module with dependencies
    h) Interfacing with a module
    i) */proc* filesystem and how it relates to modules
4) Kernel RDMA Verbs
    a) Accessing standard objects
    b) Error reporting conventions
    c) Client-server model
        i) How client, client-listener, server-agent operate and how they differ
    d) Kernel space verbs vs user space verbs

e) Protection domains
f) Completion queues
g) Queue pairs
h) Transferring messages
i) Posting send work requests
j) Getting send work completions
k) Posting receive work requests
l) Getting receive work completions
m) Explanation of why receives must be posted before sends
n) Work queues
o) Read/Write API


RDMA Transports and Protocols

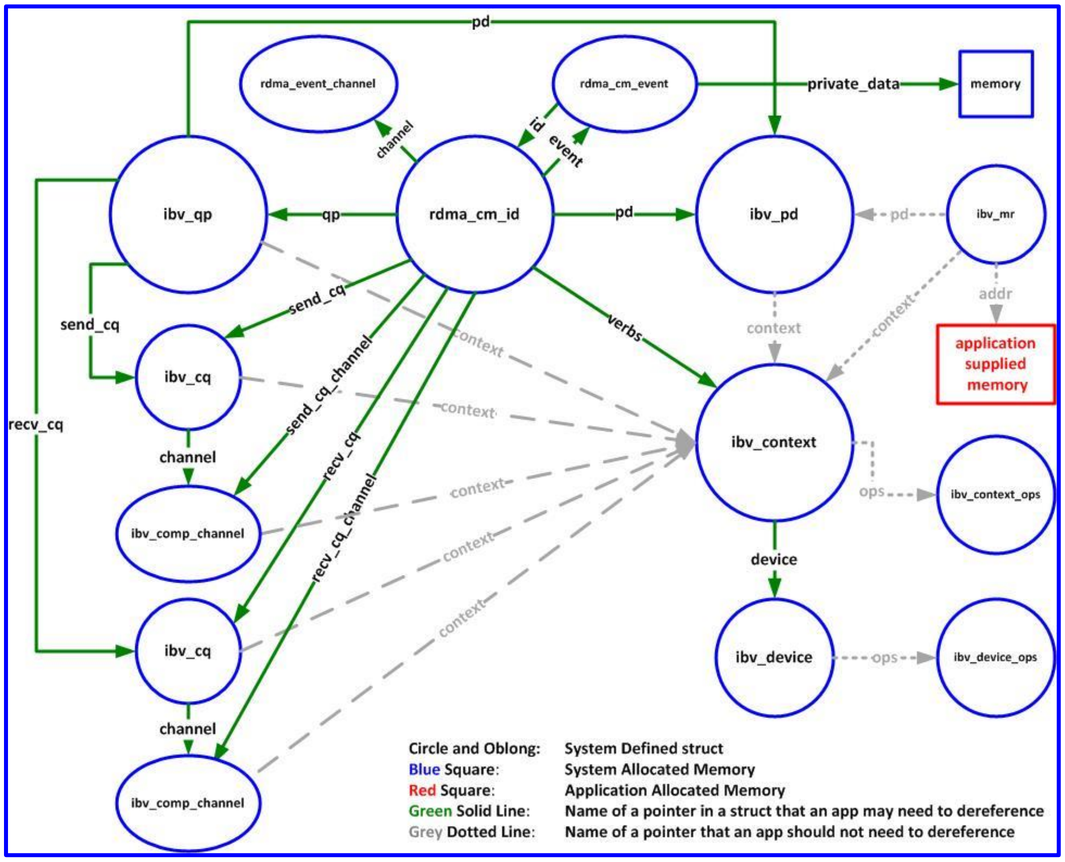
Protocol Layers


InfiniBand, RoCE and RoCEv2

Note: iWARP is not an IBTA protocol – shown for illustrative purposes

http://www.infinibandta.org/ CONFIDENTIAL 4

Architectural Overview


pd
rdma_event_channel
rdma_cm_event
private_data
memory
channel
id
event
ibv_qp
qp
rdma_cm_id
pd
ibv_pd
pd
ibv_mr
addr
send_cq
send_cq
context
verbs
context
context
application supplied memory
ibv_cq
send_cq_channel
context
recv_cq
ibv_context
channel
recv_cq
ops
ibv_context_ops
context
ibv_comp_channel
recv_cq_channel
context
device
context
ibv_cq
ibv_device
ops
ibv_device_ops
channel
ibv_comp_channel
Circle and Oblong: System Defined struct
Blue Square: System Allocated Memory
Red Square: Application Allocated Memory
Green Solid Line: Name of a pointer in a struct that an app may need to dereference
Grey Dotted Line: Name of a pointer that an app should not need to dereference