RDMA Programming

Course Description: The goal of this programming course is to provide experienced sockets programmers with the knowledge and experience they need to be able to implement parallel programming using Remote Direct Memory Access (RDMA) software. The course provides the following materials to the developers to help them rapidly understand and implement RDMA applications.

1) Diagrams to illustrate the overall RDMA architecture, transports & protocols and application flow.
2) Hundreds of slides in PDF format which explain all of the critical concepts and how to implement them.
3) 28 Client Server applications which illustrate all the concepts being presented in the slides. These applications can be used as fully functional applications to develop and implement your company's projects.

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

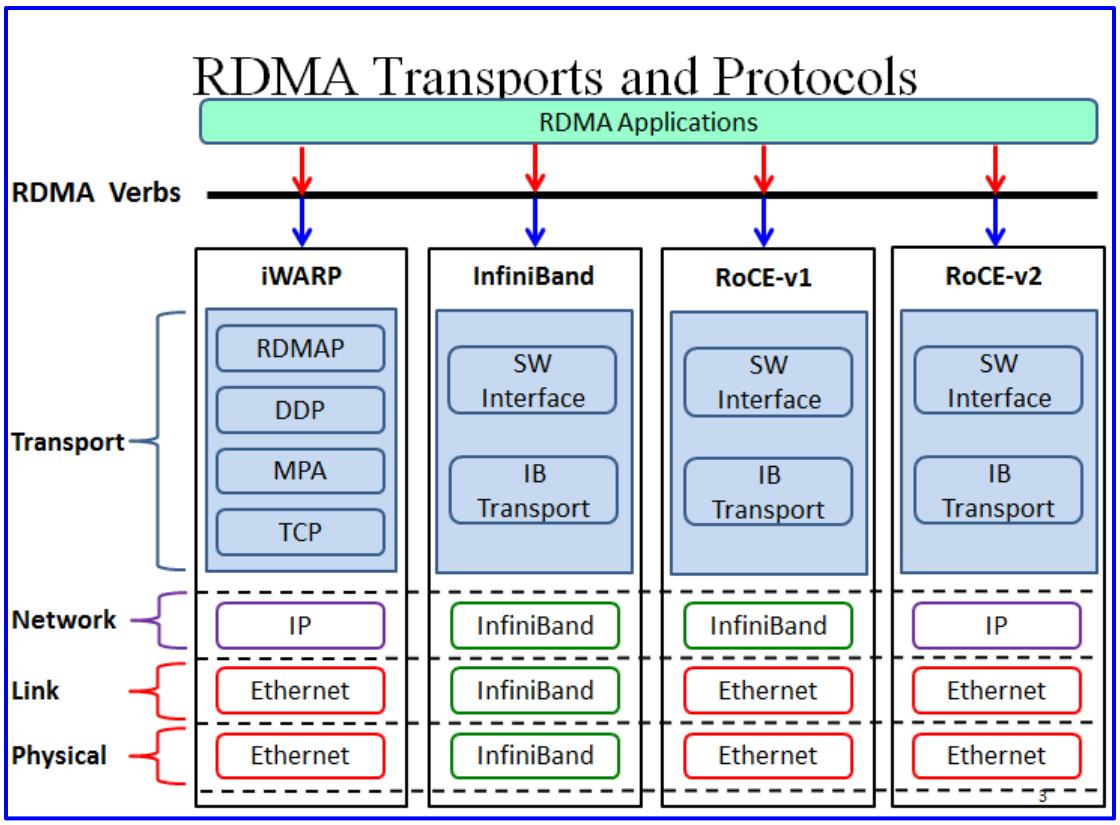
Course pre-requisites

1) Knowledge of C and Linux programming
2) Experience with sockets programming
3) Experience with parallel programming with threads

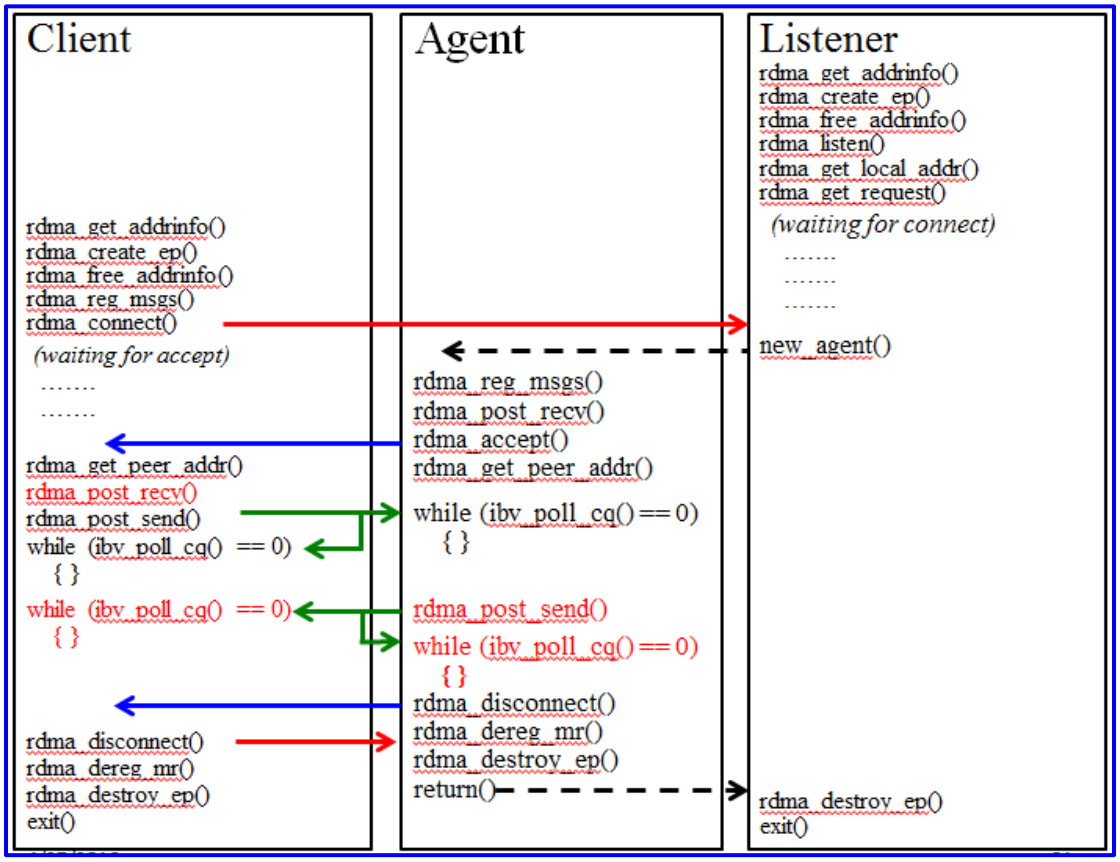
The following are some of the topics discussed in the course:

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) RDMA versus Sockets
   a) Setup differences
   b) Transfer differences
   c) Summary of the similarities and differences
   d) Implications for sockets programmers
4) Fabric Management
   a) Discussion of the fabric management in InfiniBand and how RoCE differs
   b) Review of the InfiniBand Subnet Manager (SM) used to configure fabrics
   c) Discussion of the Subnet Administrator and routing in InfiniBand
5) RDMA Connection Management (CM)
   a) Explanation of IPoIB and how it is used to exchange address info
   b) Verbs used in the CM
   c) Demonstration of CM exchange
6) Channel Semantics - 2-sided operations using send and receive
   a) Creating and sending work requests
      i) Send queue is controlled by local CA
      ii) Sending work completions

b) Creating and posting receive work requests
    i) Responder must post a receive buffer before the send is put on the wire.
    ii) Local CA processes receive work requests in FIFO order
    iii) Receive work completions
7) Memory semantics -1-sided operation
    a) Application on "active" side does a read or write - application on "passive" side does nothing!
    b) Client on the active side initiates transfer and deals with completion
    c) RDMA_READ operation - active side "pulls" data into its virtual memory from virtual memory on passive side
    d) RDMA_WRITE operation - active side "pushes" data from its virtual memory into virtual memory on passive side
    e) Both operations post to the send queue - receive queues are not involved
8) Memory Registration
    a) All memory used in transfers must be registered
    b) Memory is pinned so OS will not page out registered memory
    c) Rkeys and virtual addresses
9) Synchronous versus asynchronous programming
10) Aligning buffers on proper boundaries
11) Work requests
    a) Signaled versus un-signaled
12) RDMA Write with Immediate
    a) This is one-and-a-half-sided transfer
    b) 32-bit out-of-band "immediate" value
    c) Not applicable in iWARP
13) Inline data
    a) Used for small messages
    b) Data directly put onto the wire without referencing memory
14) Shared Receive Queues (SRQ)
15) Support for multicast operations in RDMA
16) RDMA structures
    a) See overview diagram and course material for full explanation
17) Fencing
    a) RDMA_READ may overlap with following operations RDMA_WRITE or RDMA_SEND
18) POSIX/Unix poll()
    a) Using poll for RDMA cm_events
    b) Using poll for RDMA completions
19) Performance issues
    a) How to minimize latency
    b) How to maximize bandwidth
    c) Reducing CPU utilization
20) Accessing Device Information
    a) Getting identification and status information about all RDMA devices on the system



Connection Management Flow



Architectural Overview