The Case for RDMA

Jim Pinkerton
RDMA Consortium
5/29/2002

Agenda

- What is the problem?
  - CPU utilization and memory BW bottlenecks
  - Offload technology has failed (many times)
- RDMA is a proven sol'n to the problem
- What is RDMA?
  - Example protocol – Sockets Direct Protocol
  - Common objections to RDMA

Existing Performance Bottlenecks

- Network CPU utilization limits the CPU-bound application
  - Any size message
    - Protocol overhead (interrupts, ack processing, etc)
  - Medium/Long messages
    - Receive Copy overhead
  - Short messages
    - Kernel bypass (completions are the issue)
- Bandwidth limits
  - Receive Copy limits single stream BW to the bcopy rate of a single CPU

Scaling - Throughput

Test	Throughput (Mb/sec)	Tx CPUs	Rx CPUs
1 GBE, TCP	769	0.5 CPUs	1.2 CPUs
WSD SAN	891	0.2 CPUs	0.2 CPUs
10 G/s, TCP	7700	5.0 CPUs	12 CPUs
10 G/s TCP Offload or WSD SAN	9000	0.5 CPUs	0.5 CPUs

We've got a Problem here…

Red = Conjecture

All tests on Windows 2000, Service Pack 2, running ttcp throughput tests

Can't wait for Moore's Law to fix this

Todays tests: 64 KB window, 64 KB I/Os, 2P 600 MHz PIII,
Tomorrow's tests: 640 KB window, 64 KB I/Os
THIS IS A BEST CASE SCENARIO – 9000 B MTUs. For 1500 B MTU, scale packet rate by 6

White paper at: http://www.microsoft.com/windows2000/techinfo/howitworks/default.asp

Scaling - Transactions

- Increased Data Center Capacity
  - CPU bound apps have more CPU
  - Distributing applications are more attractive if less overhead
  - Maintenance functions have less overhead
    - Backup, restore, content distribution
- Null RPC tests - 60% better with RDMA
  - measure the network overhead of RPC
  - while() loop doing RPCs which don't perform any work

Test	Null RPC/sec	Client % Utilization	Server % Utilization
WSD SAN, XP	34,162	97.4%	89%
WSD SAN, W2K	28,860	98.6%	93.7%
TCP, W2K	21,310	95.6%	82.5%

Scaling - Memory BW

Fabric	Raw Bandwidth (BW)	Memory BW for Bcopy (3x raw rate)	CPUs for Bcopy at 200 MB/sec	Total Memory BW (Bcopy + DMA)
Fibre Channel	100	300	1.5	400
GBE	125	375	1.9	500
10 GBE	1250	3750	18.8	5000
IB 1x	250	750	1.25	1000
IB 4x	1000	3000	5.0	4000
IB 12x	4000	12000	20.0	16000

For receive we've got a problem without zero copy…

Red = Conjecture

Existing Cost Bottlenecks

- **Initial Cost**
  - TCP Offload NICs have a huge amount of buffering
    - High speed memory **interface** to off chip memory is expensive
    - High speed memory is expensive
    - Puts the NIC vendor in an awkward position
      - Are they designing for high latency links or low latency – maps to how much buffering is required.
  - Today's proprietary RDMA NICs take above cost and add more
- **Management Cost**
  - Today's proprietary RDMA NICs require
    - Proprietary management applications
    - Separate network within the data center
- **TOE does not solve the receive copy**
  - Middleware library header/data split

RDMA is a proven approach for Today's Applications

- Transactional Database
  - SQL, DB2, Oracle all get best performance from RDMA fabrics today (VIA)
  - SDP – Sockets Direct Protocol – is also used
- File Oriented Storage
  - DAFS, BDS (SGI), NFS (research by Sun)
- Block Oriented Storage
  - ANSI SRP, ANSI SST
- High Performance Computing
  - Various MPI libraries based on VIA
  - ASCI Blue Mountain (SGI)
- Backup
  - Backup system from SGI
- Summary
  - **All met their targeted performance goals**
  - **All are currently limited to the Data Link Layer**
  - **All the above are commercial products**
  - **Almost none of the above are/were commercially successful**

Why can offload/RDMA work now?

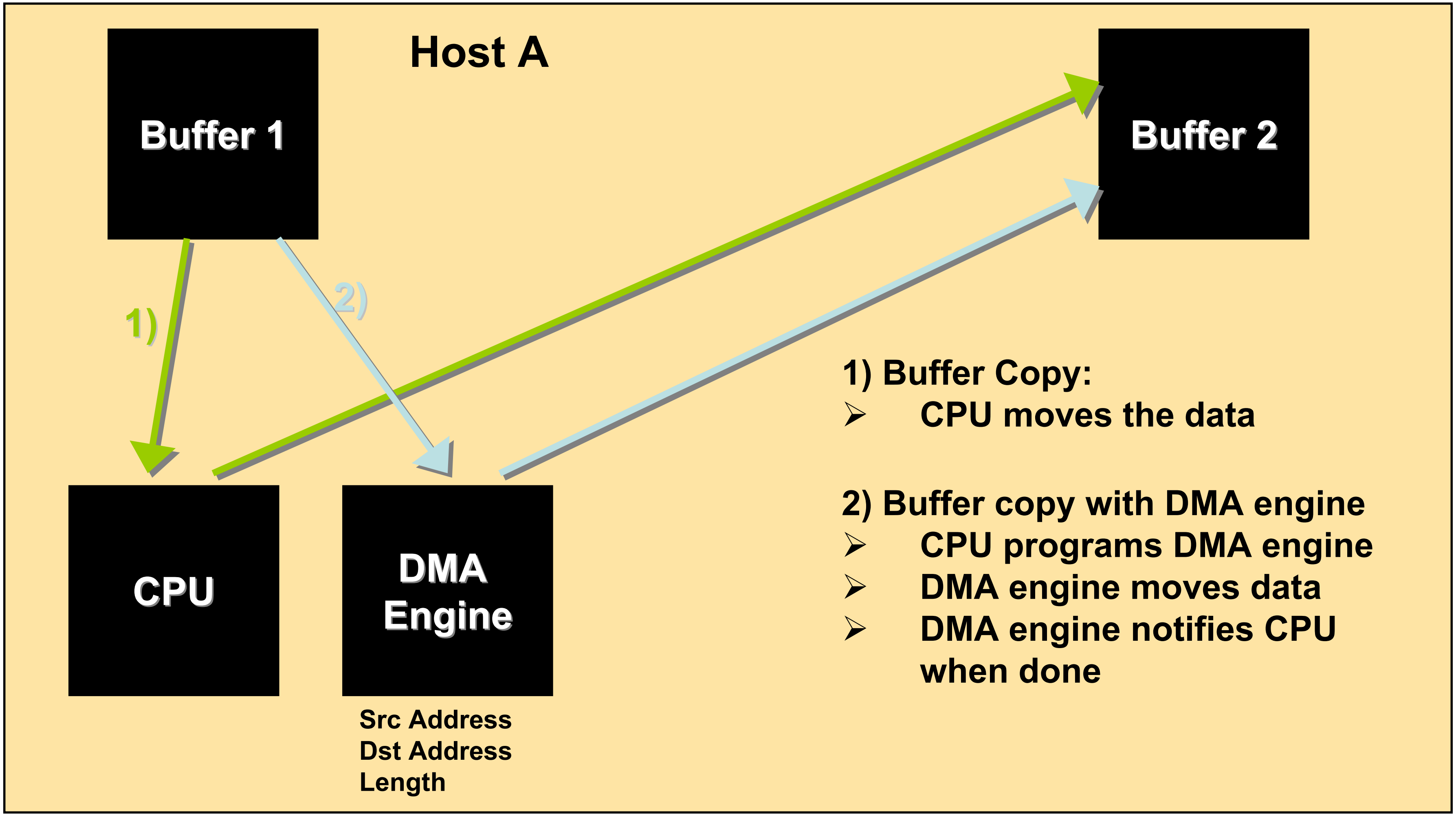
- NIC volume is approaching CPU volume
- Ratio of CPU power to network speed is closing
- 10 GB Ethernet has a serious problem
- Key problems of offload technology
  - NICs are too expensive
    - Initial cost – solved by silicon integration, volume
    - Management cost – currently proprietary fabric, solved by moving to standard network infrastructure (ethernet)
  - Applications don't take advantage of it
    - Sockets Direct Protocol enables unmodified applications to use it
- Think of Graphics offload
  - Early days CPU attempted to take over the graphics offload market
  - Strong need, with applications, created a self-sustaining innovative cycle of profits and next generation hardware

Proposal: Create an RDMA Protocol at the Transport Layer

Enable RDMA applications to "**get off the link**" by creating an RDMA protocol shim at the transport layer

So What Is RDMA?

Traditional DMA



Src Address
Dst Address
Length

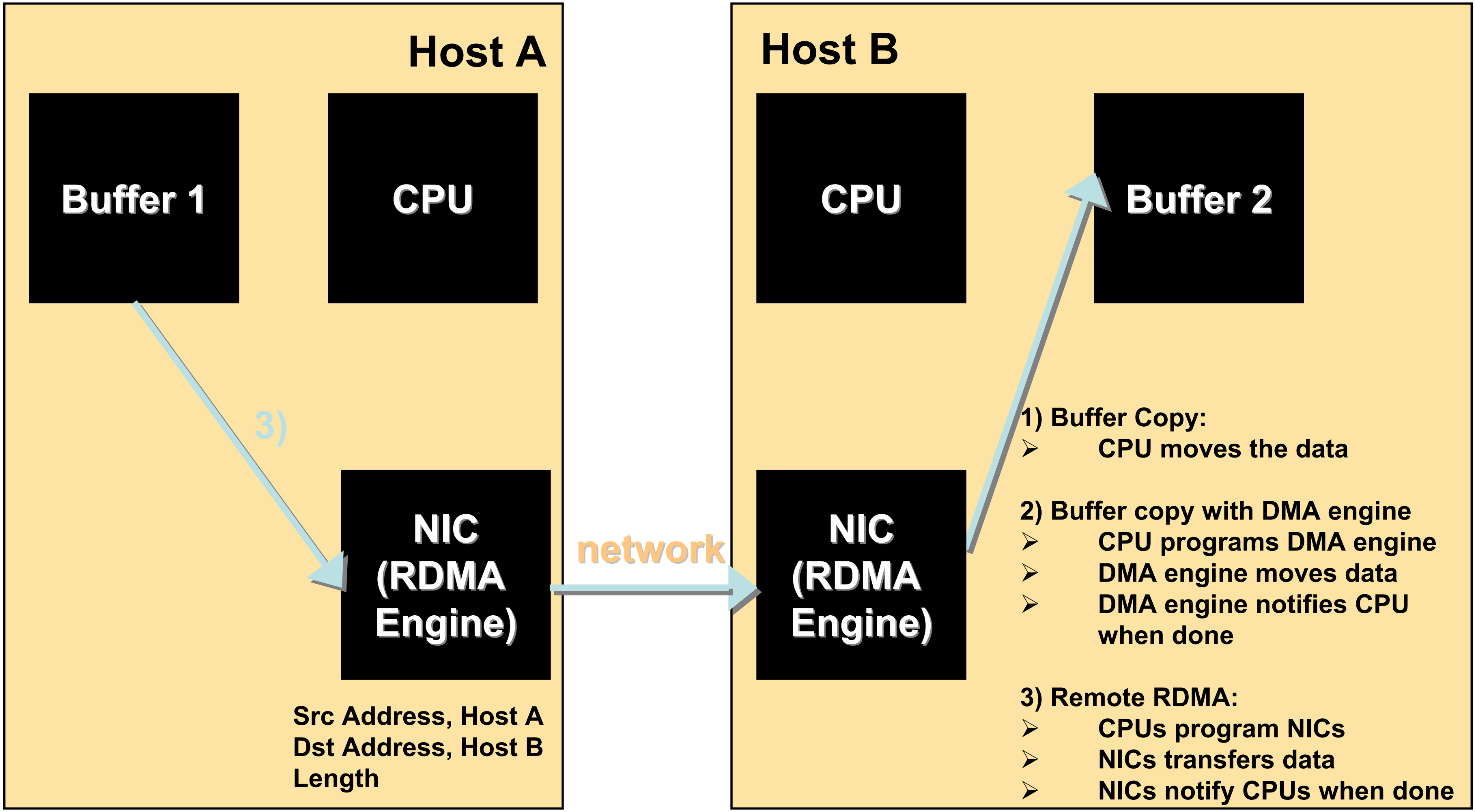
1) Buffer Copy:

- ➢ CPU moves the data

2) Buffer copy with DMA engine

- ➢ CPU programs DMA engine
- ➢ DMA engine moves data
- ➢ DMA engine notifies CPU when done

Remote DMA



Src Address, Host A
Dst Address, Host B
Length

1) Buffer Copy:
- ➢ CPU moves the data

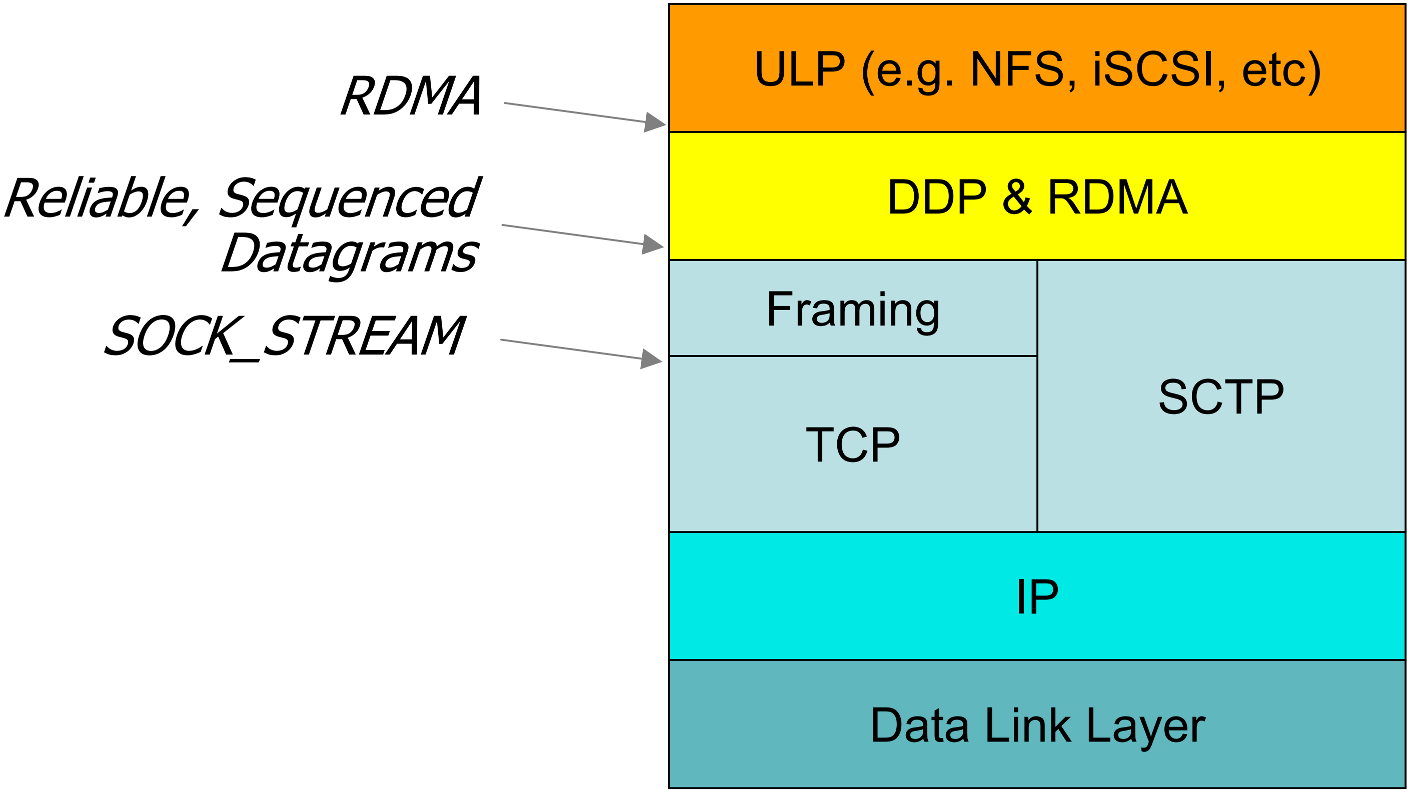
2) Buffer copy with DMA engine
- ➢ CPU programs DMA engine
- ➢ DMA engine moves data
- ➢ DMA engine notifies CPU when done

3) Remote RDMA:
- ➢ CPUs program NICs
- ➢ NICs transfers data
- ➢ NICs notify CPUs when done

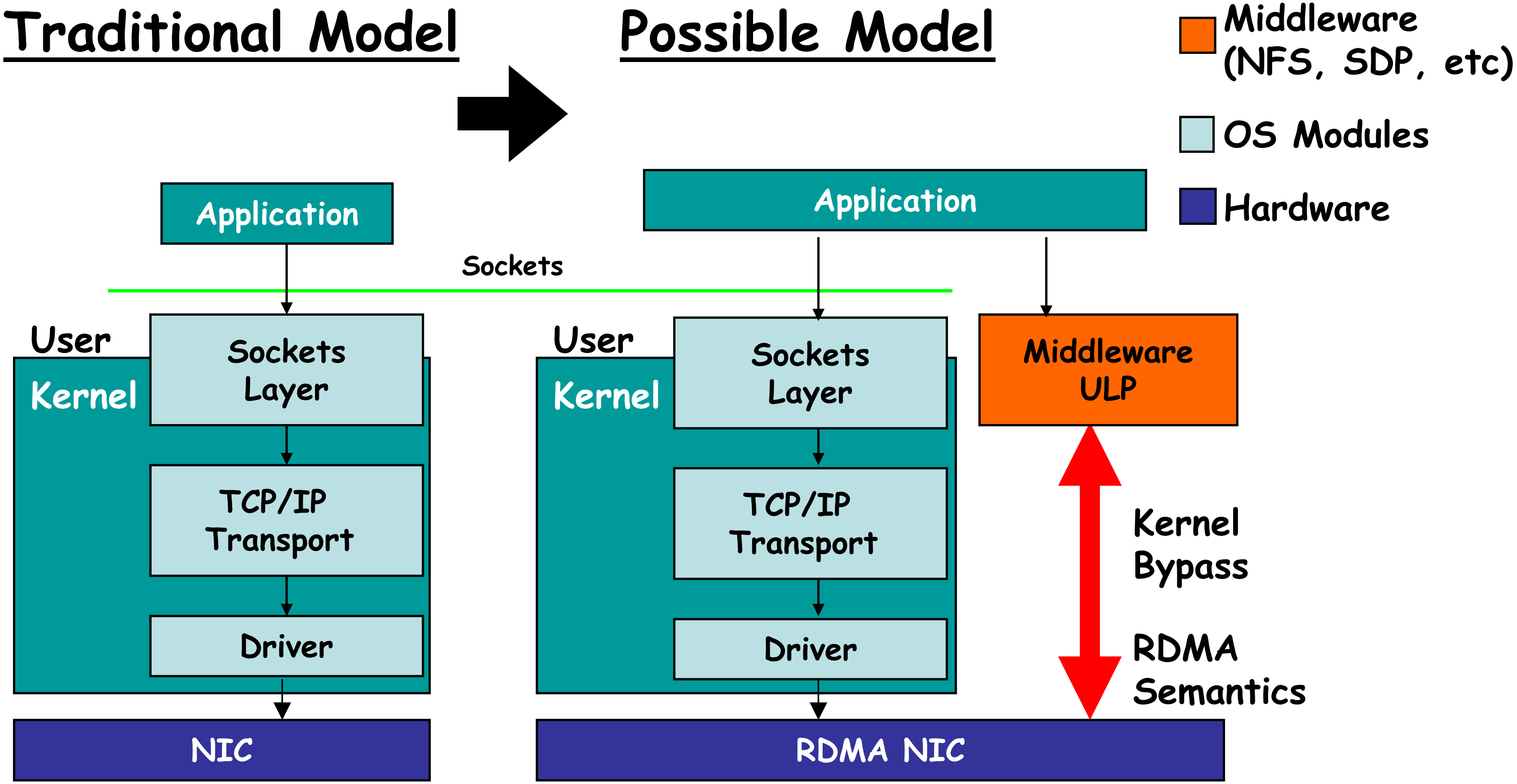
So What is RDMA?

- Typically 3 Data Transfer Mechanisms
  - RDMA Write
  - RDMA Read
  - Sequenced Reliable Datagrams (**Sends**)
    - Pass thru to SCTP (some outstanding issues)
    - Requires Framing for TCP
- Transfer mechanisms can be combined by the ULP to create ULP unique sequences that don't require the destination to process intermediate operations
  - Explicit ULP source controlled event notification
- Enables ULP to
  - explicitly demultiplex header from data
  - explicitly manage their buffers
- Enables a low-latency infrastructure in the data center
- **Shown to be a useful model for a wide variety of ULP application types**

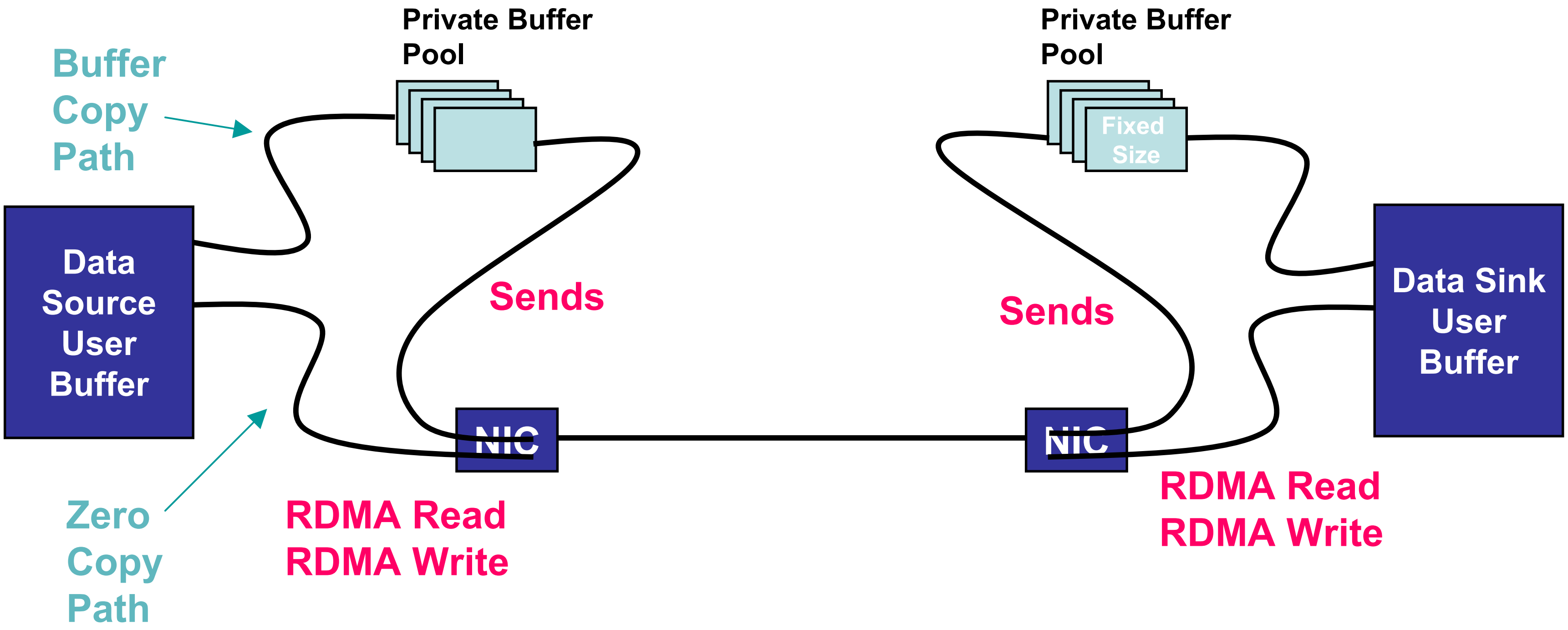
Proposed RDMA Layering



One Possible Kernel Bypass Architectural Model

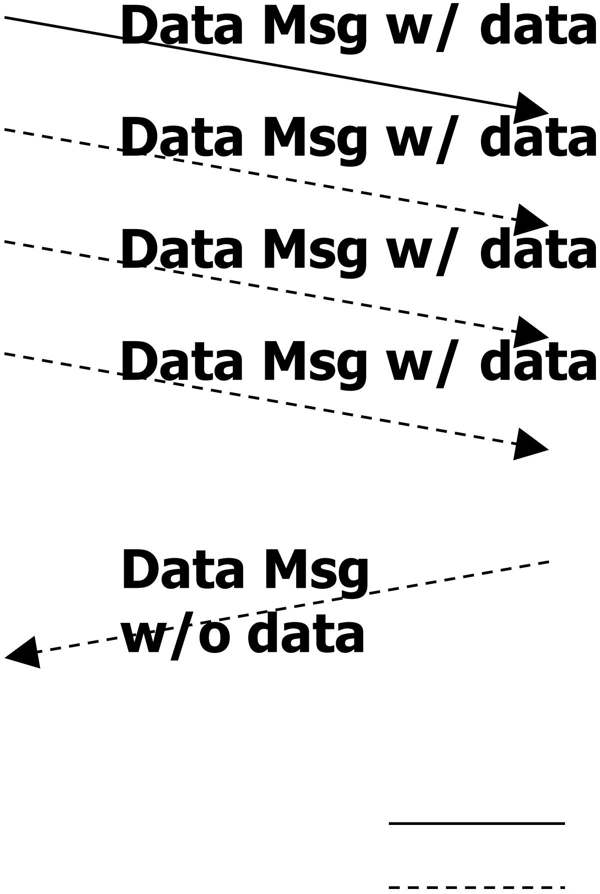


Ex: RDMA ULP - Buffer Model



- Enables buffer-copy when
  - Transfer is short
  - Application needs buffering
- Enables zero-copy when
  - Transfer is long

Ex: RDMA ULP - Bcopy



Data Source

Data Sink

Send of data in buffer size chunks

Data Msg w/ data

Data Msg w/ data

Data Msg w/ data

Data Msg w/ data

Receive data in buffer size chunks

Flow control update is piggybacked on reverse channel traffic

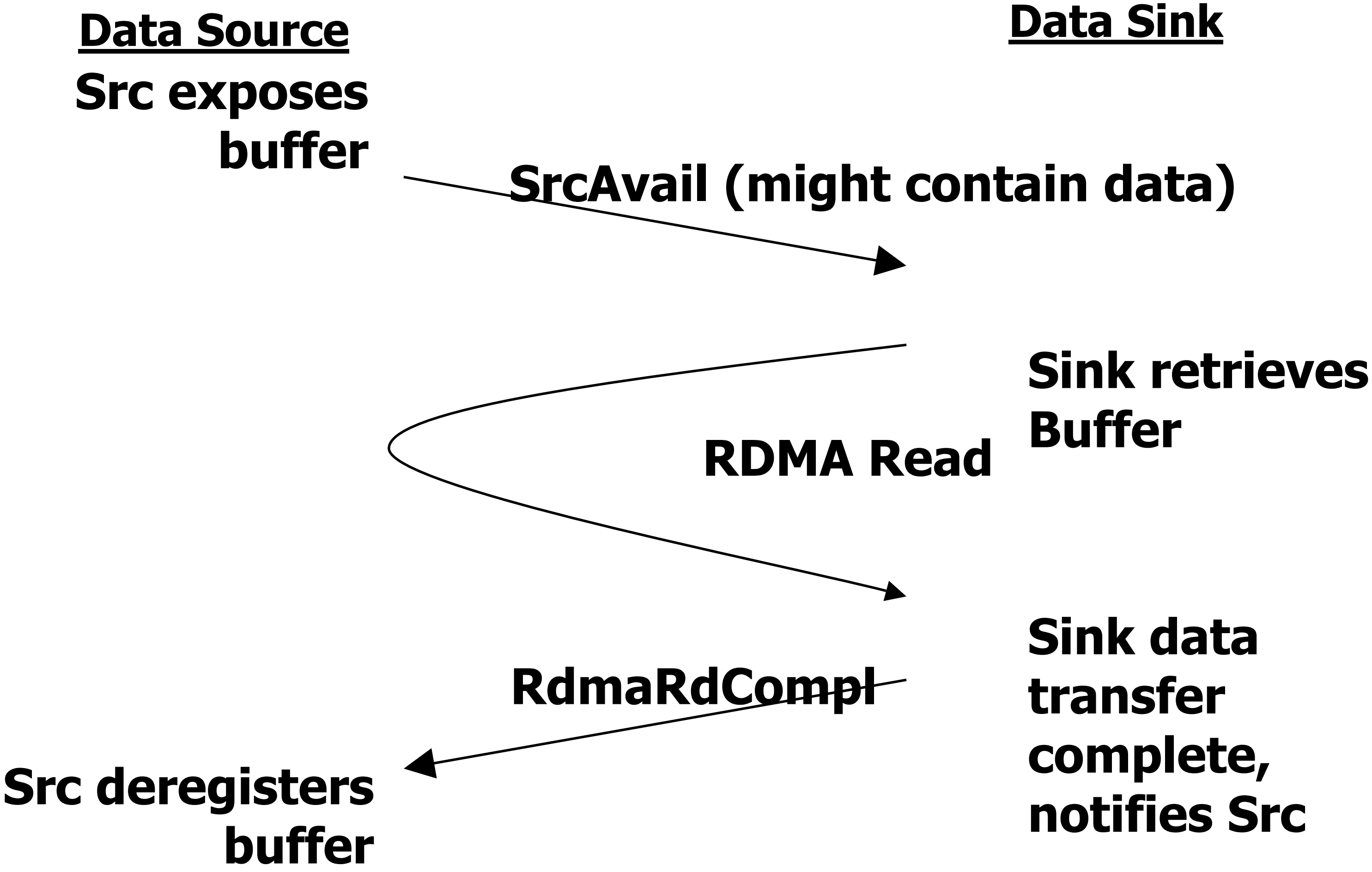
Data Msg w/o data

No ACK for data because link is reliable

Required msg

Optional msg

Ex: RDMA ULP - Read Zcopy



Data Source

Data Sink

Src exposes buffer

SrcAvail (might contain data)

Sink retrieves Buffer

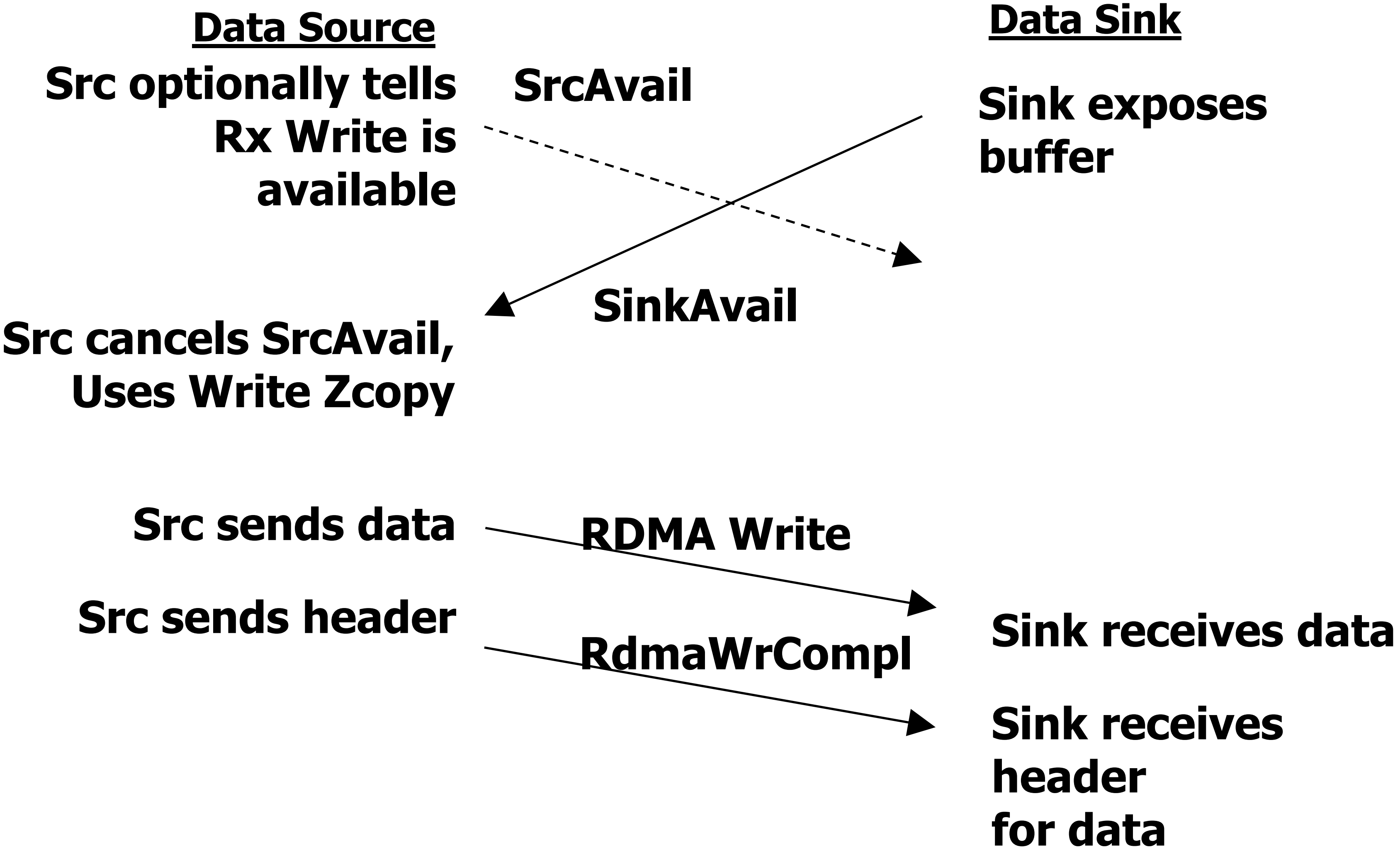
RDMA Read

Sink data transfer complete, notifies Src

RdmaRdCompl

Src deregisters buffer

Ex: RDMA ULP - Write Zcopy



Data Source

Data Sink

Src optionally tells Rx Write is available

SrcAvail

Sink exposes buffer

SinkAvail

Src cancels SrcAvail, Uses Write Zcopy

Src sends data

RDMA Write

Sink receives data

Src sends header

RdmaWrCompl

Sink receives header for data

Other Methods That Are Not General or Don't Scale

- Page flipping
- Header/payload separation
- Scatter Gather List (SGL)
- NIC memory mapping
- Terminate ULP in the NIC

Why Now

- Industry has played quite a bit with RDMA. Solutions are converging.
  - Open October 2001 RDMA face-to-face had wide consensus on requirements for the protocol
  - Proprietary RDMA solutions have shown they don’t have the volume to be sustainable (interoperable standard is required)
- Ratio of network I/O to CPU has changed
- Volume of network cards now approaches volume of CPU
  - There is sufficient profit to continue improvements in the technology while CPU speeds increase

DDP & RDMA

- Direct Data Placement (DDP)
  - Some within the IETF prefer to solve only the receive side copy problem
  - Simplification of RDMA Write
- RDMA and DDP
  - Some prefer to solve receive side copy, **plus**:
    - Short message problem
    - Create a richer message paradigm
    - Optimizations unique for:
      - Distributed applications
      - High Performance Computing (HPC) Applications
      - Pull data instead of push data
      - Distributed lock management

Some Objections to RDMA

- More complex API than TCP/stream interface
  - A new API is not required (but worthwhile for some applications) – Sockets Direct Protocol (SDP)
- RDMA-accelerated ULPs not wire compatible with unaccelerated variants
  - Extremely valid – but worth the cost
- Hardware vendors must all agree for approach to succeed in the market
  - Extremely valid – but worth the cost

Some Objections to RDMA

- Security concerns about opening memory on the network
  - Hardware enforces application buffer boundaries
    - Makes it no worse than existing security problem with a 3rd party inserting data into the TCP data stream
  - Buffer ID for one connection must not be usable by another connection

Bottom Line

- There is a real problem:
  - In scaling today's applications in the data center
  - In scaling the network to 10 GBit Ethernet speeds
- RDMA and Direct Data Placement is a proven technology to solve the problem – but several problems need to be solved
  - RDMA needs a transport layer encapsulation
  - RDMA spec must be an open and interoperable standard
- **Thus RDMA should be standardized on Ethernet fabrics using Internet protocols**

Additional Reading

- IETF RDMA Problem Statement
  - http://www.ietf.org/internet-drafts/draft-romanow-rdma-over-ip-problem-statement-00.txt
  - Proposed charter on rdma reflector at yahoogroups.com
- Supporting Material
  - http://www.microsoft.com/windows2000/techinfo/howitworks/default.asp
  - http://www.cs.duke.edu/ari/publications/end-system.{ps,pdf}
  - H.K. Chu, "Zero-copy TCP in Solaris", Proc. of the USENIX 1996 Annual Technical Conference, San Diego, CA, Jan. 1996
  - V. S. Pai, P. Druschel, W. Zwaenepoel, "IO-Lite: a unified I/O buffering and caching system", Proc. of the 3rd Symposium on Operating Systems Design and Implementation, New Orleans, LA, Feb. 1999
  - See references at end of problem statement