Sockets Direct Protocol v1.0 RDMA Consortium

Jim Pinkerton

RDMA Consortium

10/24/2003

Agenda

- Problems with existing network protocol stacks
- Sockets Direct Protocol (SDP)
  - Overview
  - Protocol details
    - Connection setup, including SDP Port Mapper
    - Data transfer mechanisms
    - Modes
    - Buffering issues
    - Comparing SDP on iWARP to SDP on InfiniBand

Problems with Existing Protocol Stacks

- Excessive CPU utilization
  - Memory-to-memory copying of data
  - User context switches
  - User/kernel transitions
  - Interrupts
  - Protocol overhead

All Contribute to:
- **Decreased I/O Operations per second**
- **Increased Latency**
- **Decreased Bandwidth**

- Excessive memory bandwidth utilization
  - Copying of data triples or quadruples memory bandwidth requirements. Data is moved into intermediate memory buffers from the network, then copied to the final buffer (i.e. moved to the CPU, and then moved back to a new memory location) for a total of 3x bandwidth. Often memory coherency traffic can effectively cause a total of 4x bandwidth consumption.

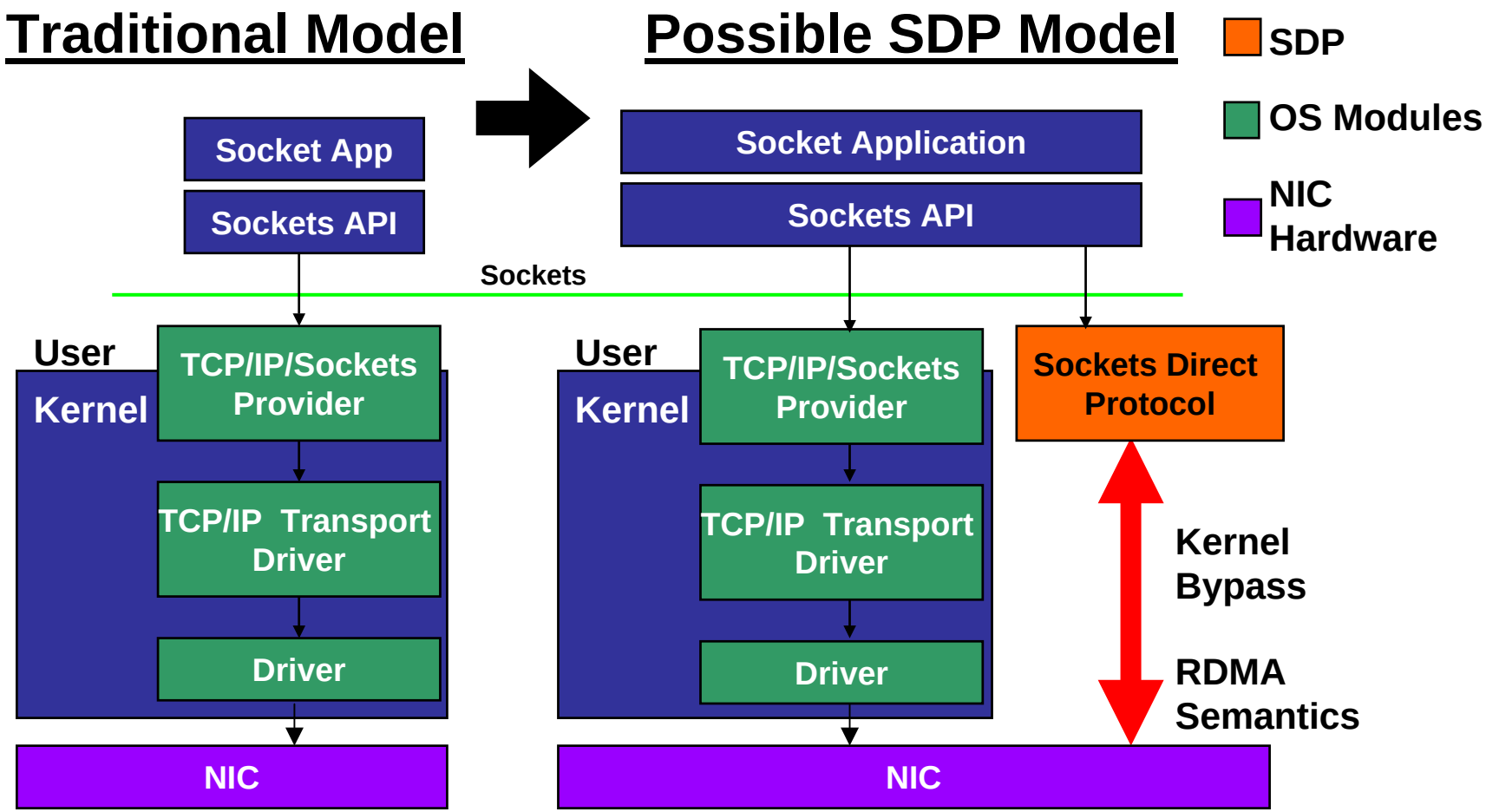
SDP Highlights

- **Enable RDMA optimized transfers while maintaining traditional socket stream semantics**
  - SDP Port Mapper enables SDP to integrate with existing TCP port namespace
  - SDP over iWARP can cross multiple subnets
  - RDMA Consortium SDP requires iWARP RDMA
- Protocol offload
  - Reliable, in-order delivery in hardware
  - NIC demultiplexes data stream instead of OS
- No copying of data when appropriate
  - Use of RDMA Reads and Writes enables direct data placement into application buffer
- Architectural optimizations
  - Kernel bypass, Interrupt avoidance
- Based on an existing protocol
  - In commercial use, field proven, shipping on existing RDMA networks today
- Compatible with SDP on InfiniBand

SDP Focus

- **Map byte-stream protocol to iWARP's RDMA Write, RDMA Read, and Send transfers**
  - Sockets is the dominant API for networking programming (traditionally used for TCP/IP)
- **Socket stream semantics**
  - SOCK_STREAM w/ TCP error semantics, same TCP port range
  - Graceful & abortive close, including half closed sockets
  - IPv4 or IPv6 address resolution
  - Out-of-Band data, socket options
  - Socket duplication
- **SDP Optimizations**
  - Transaction oriented applications
  - Mixing of small and large messages

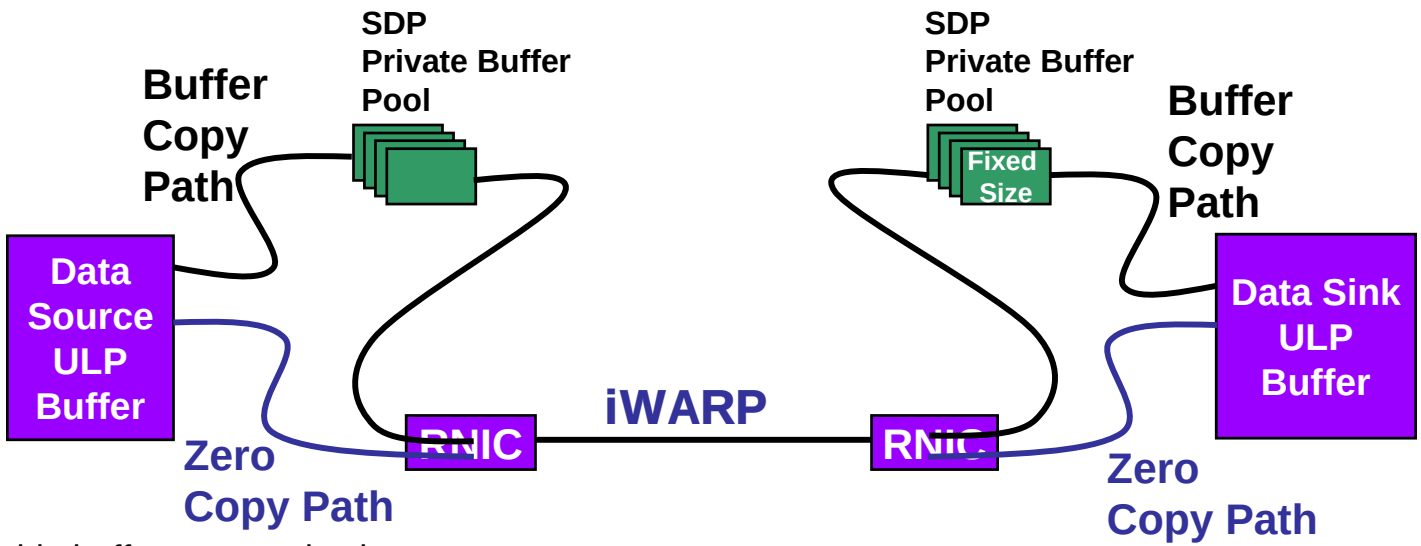
Example SDP Architectural Model



SDP Terminology

- **Connecting Peer** – The side of the connection which initiated connection setup.
- **Accepting Peer** – The side of connection which received the connection setup request.
- **Data Source** – The side of connection which is sourcing the ULP data to be transferred.
- **Data Sink** – The side of connection which is receiving (sinking) the ULP data.
- **Data Transfer Mechanism** – The mechanism to move ULP data from the Data Source to the Data Sink (Bcopy, Write Zcopy, Read Zcopy, and Transactions).
- **Flow Control Mode** – The state that the half connection is currently in (Combined, Pipelined, or Buffered). Constrains which Data Transfer Mechanisms can be used.
- **Bcopy Threshold** – If the message length is under a threshold, use the Bcopy mechanism. The threshold is locally defined.

SDP Buffering Model

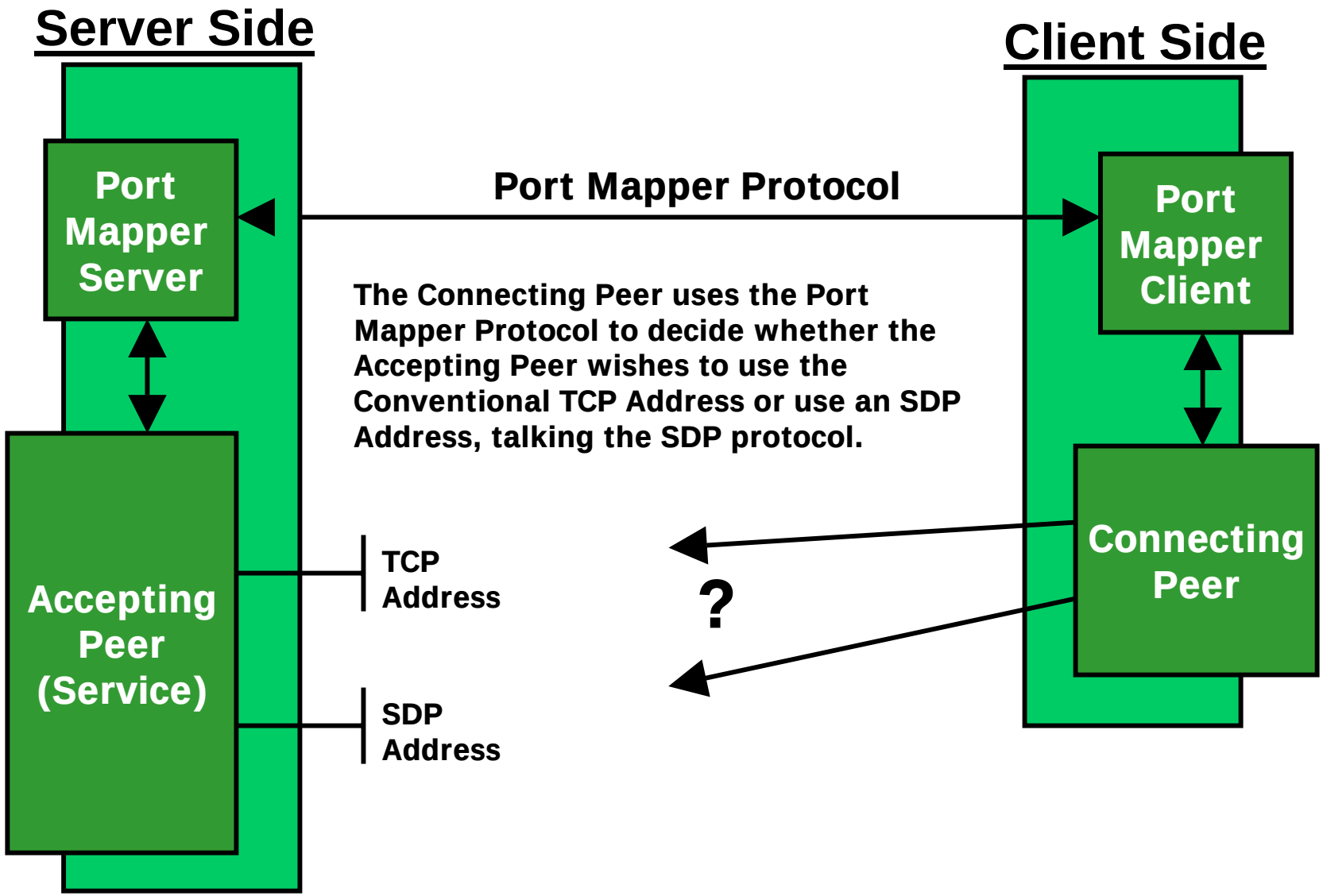


- Enable buffer-copy path when
  - **Transfer is short** – Overhead associated with pinning the ULP buffer in memory, advertising the buffer to the Remote Peer, and then transferring the data is higher than simply copying the data into the send SDP Private Buffer Pool, sending it directly to the Remote Peer's receive SDP Private Buffer Pool, and then copying it into the ULP buffer.
  - **Application needs buffering** – Some applications require the network to buffer the data for good performance. For example, if the application has many short transfers to perform, but only allows one transfer to be outstanding at a time, it is usually more efficient to copy the transfers into a send buffer and return to the application immediately. The ULP data is transferred to the Remote Peer at a later time.
- Enable zero-copy path when
  - **Transfer is long** – If the transfer is long, then the overhead to pin the ULP send and receive buffers in memory and exchange a steering tag to directly transfer the data between the ULP buffers is small compared to the cost of copying all the data into the SDP Private Buffer Pool.

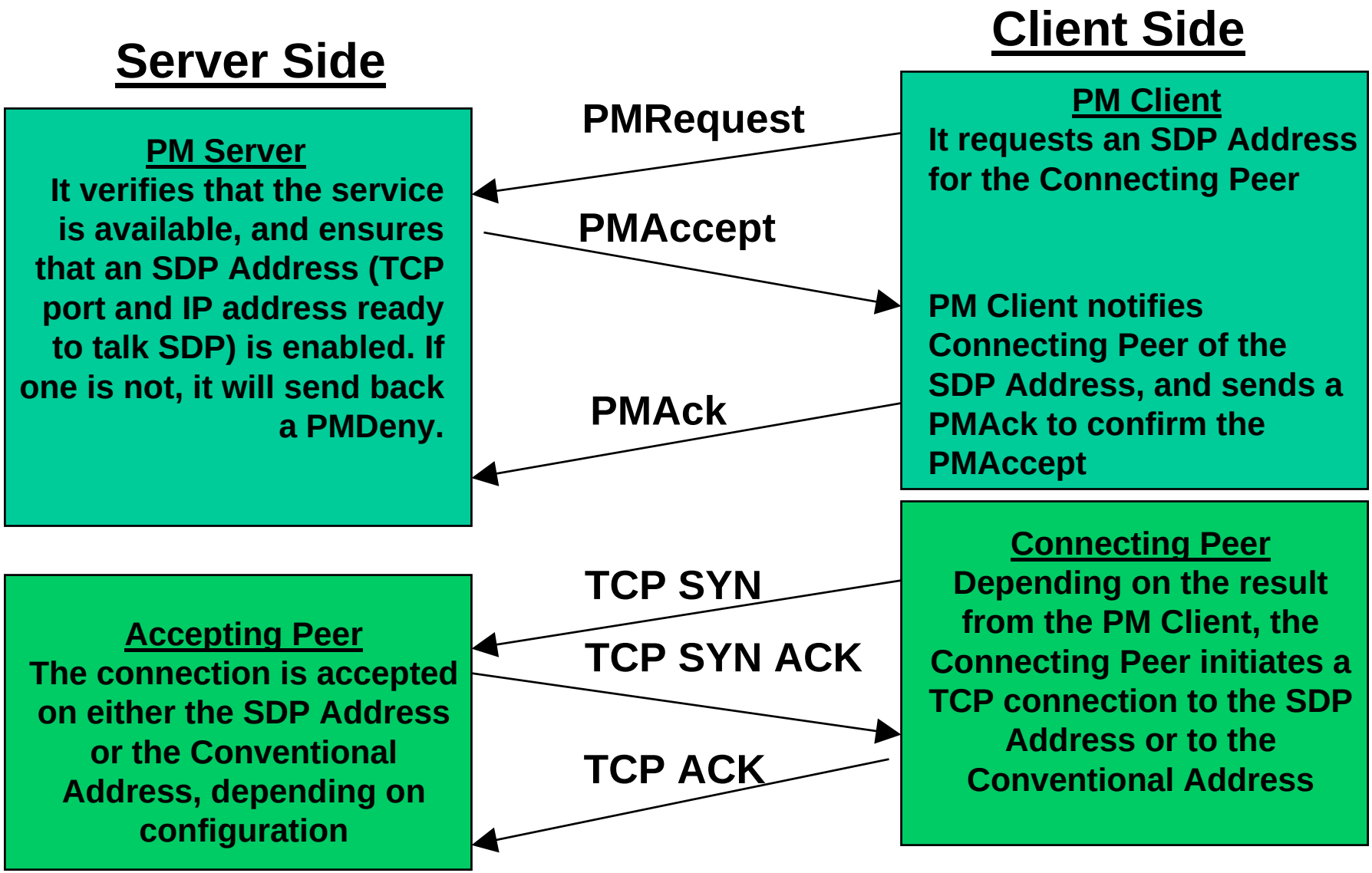
SDP Port Mapper

- SDP Port Mapper enables re-use of the existing network service mapping infrastructure
  - Continue to use existing mechanisms to map a service name to a TCP Port and IP address (referred to as the Conventional Address).
  - SDP Port Mapper Protocol enables an SDP Connecting Peer to discover an SDP Address given a Conventional Address. An SDP Address is a TCP Port and IP Address for the same service, but the SDP Address requires data to be transferred using SDP over RDMA.
  - Connecting Peer uses the SDP Address to establish a TCP connection.
  - Connecting Peer uses SDP Hello Message to advertise startup parameters and then enables iWARP mode
  - All further SDP communication uses iWARP

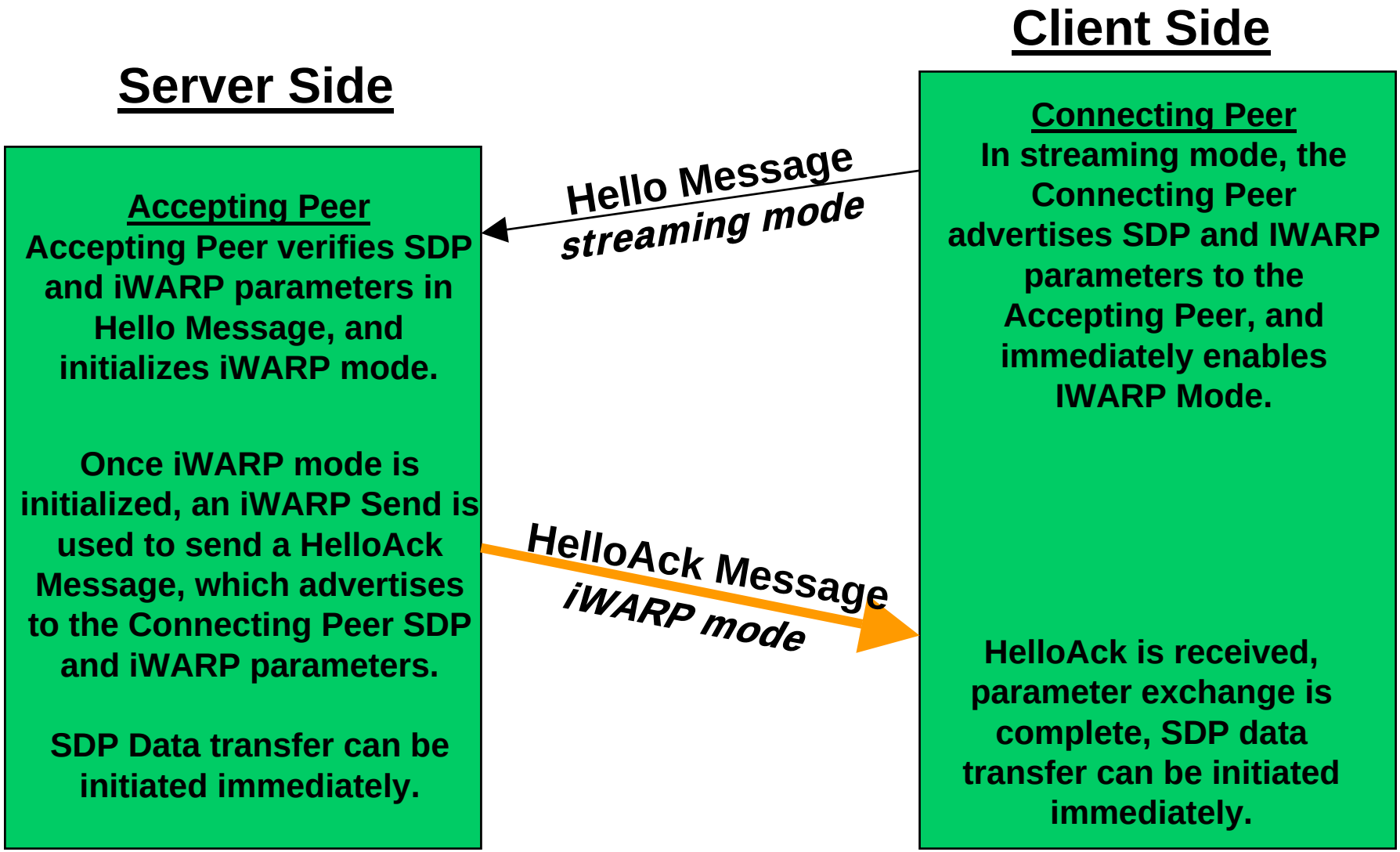
Port Mapper Protocol Architecture



SDP Port Mapper Protocol



SDP Enables iWARP Mode

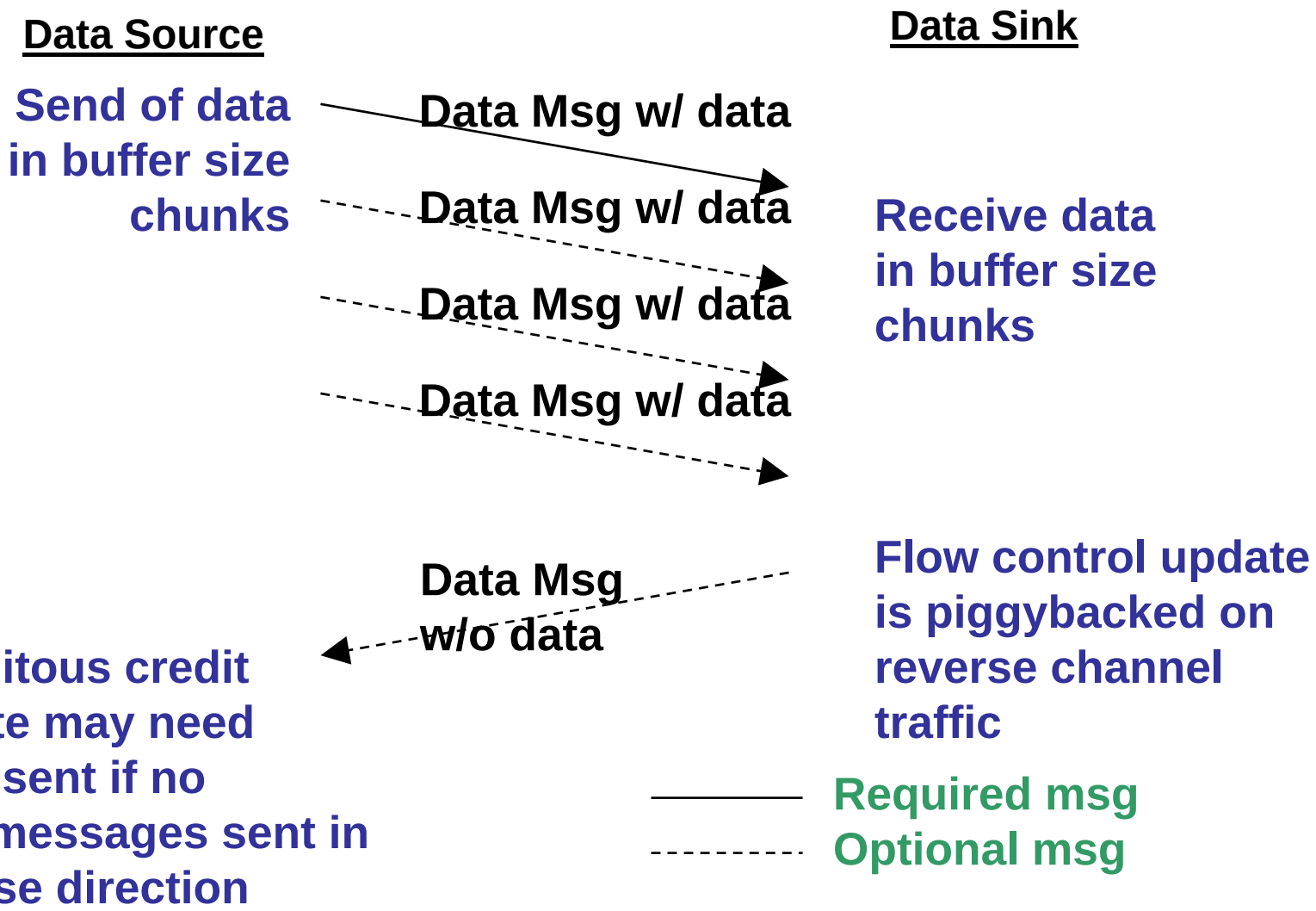


Streaming mode is conventional TCP/IP data transfer
iWARP mode requires iWARP over TCP/IP

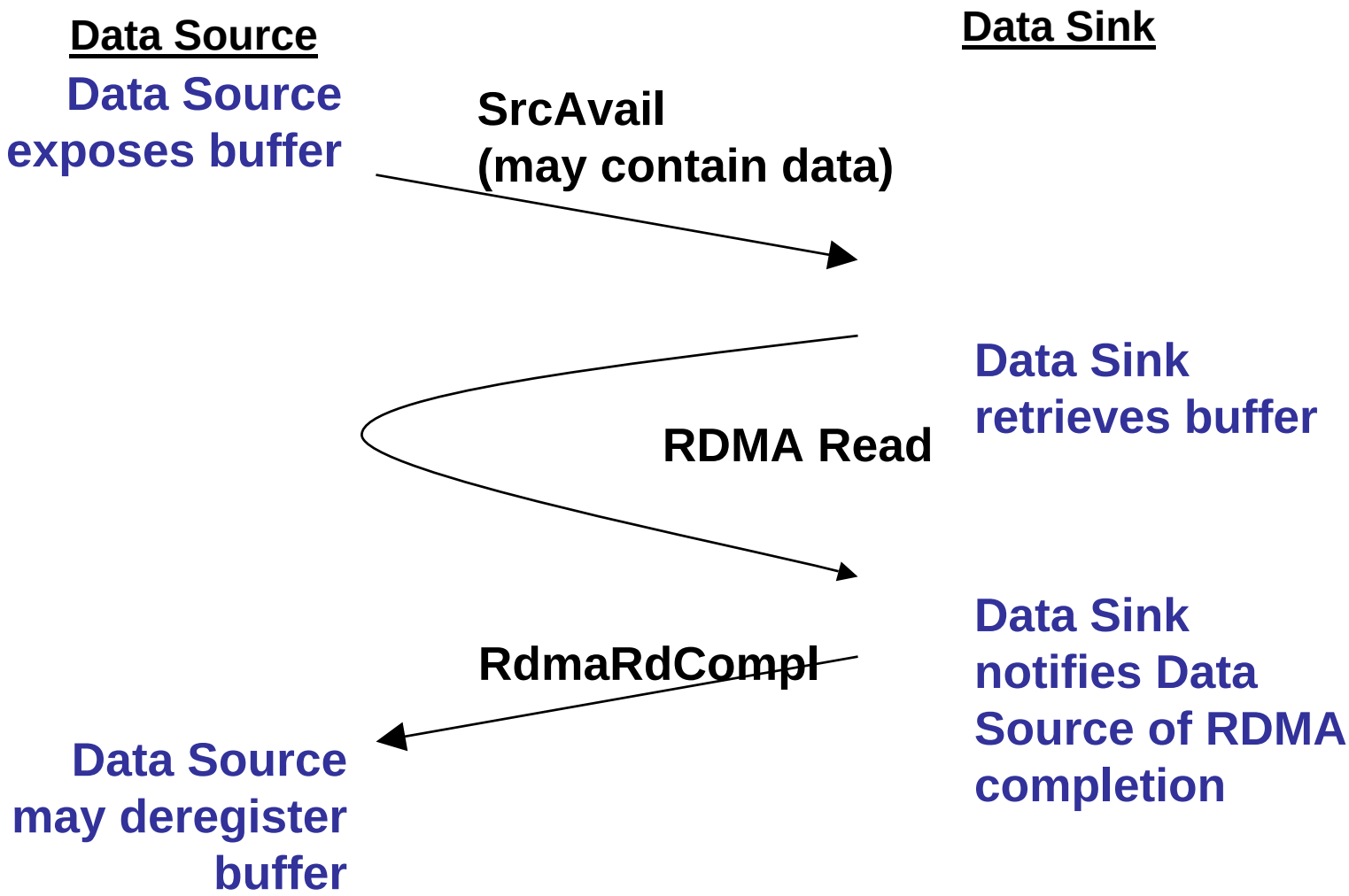
SDP Data Transfer Mechanisms

- **Bcopy** - Transfer of ULP data from send buffers into receive Private Buffers.
- **Read Zcopy** - Transfer of ULP data through RDMA Reads, preferably directly from ULP Buffers into ULP Buffers.
- **Write Zcopy** - Transfer of ULP data through RDMA Writes, preferably directly from ULP Buffers into ULP Buffers.
- **Transaction** - An optimized ULP data transfer model for transactions. It piggy-backs ULP data transfer using Private Buffers on top of the Write Zcopy mechanism used to transfer ULP data on the opposite half-connection.

SDP Data Transfer Mechanisms: Bcopy

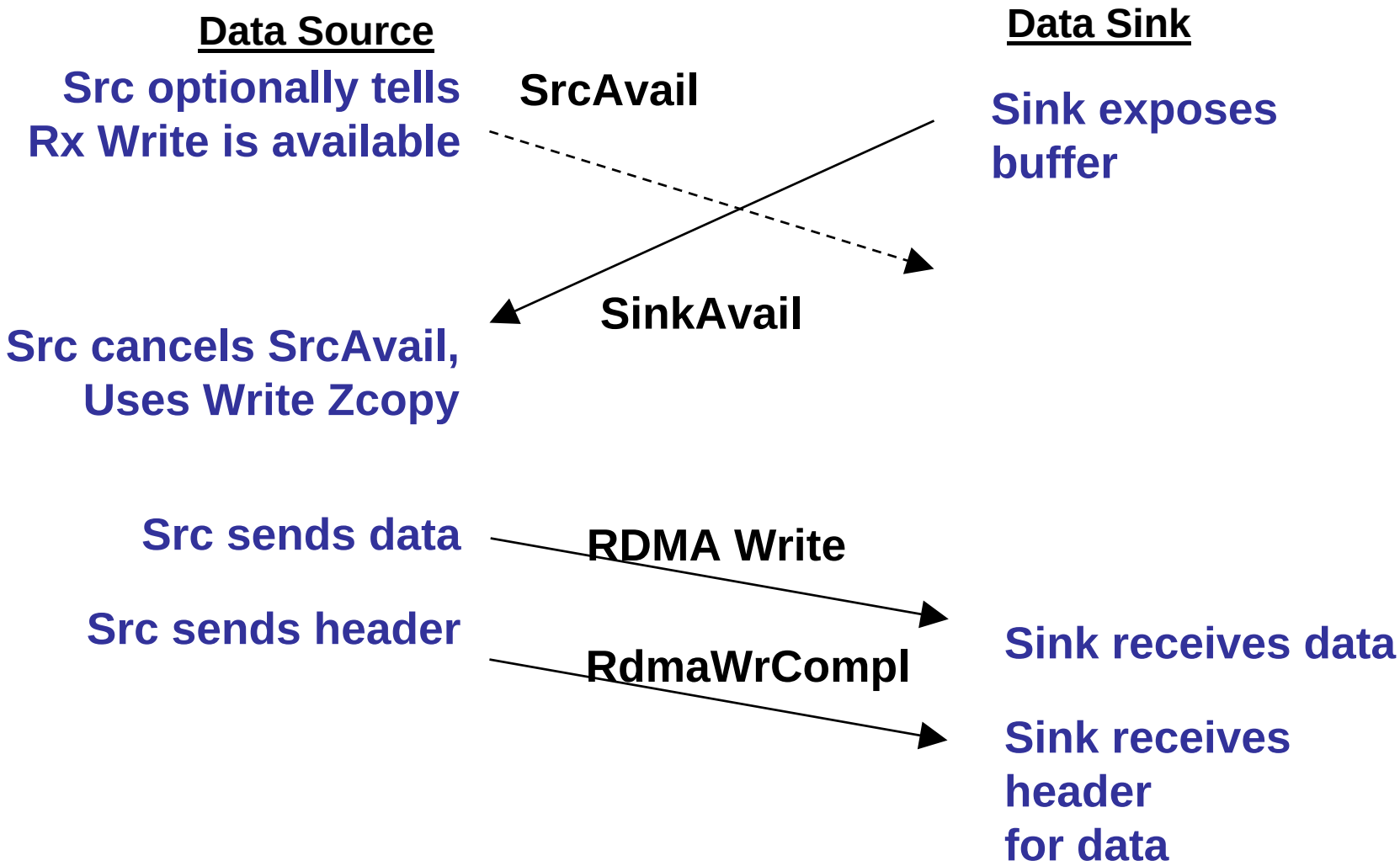


Data Transfer Mechanisms: Read Zcopy



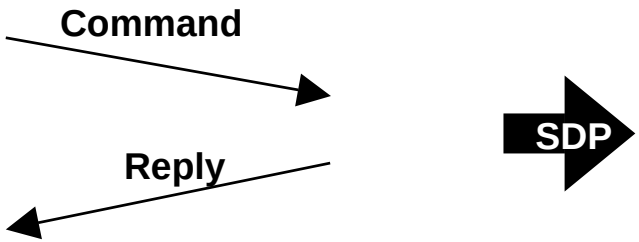
Advantage: One less operation at Data Source.

Data Transfer Mechanisms: Write Zcopy



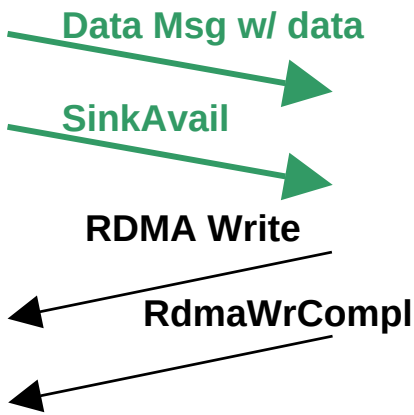
Data Transfer Mechanisms: Transactions

Data Transaction



SDP

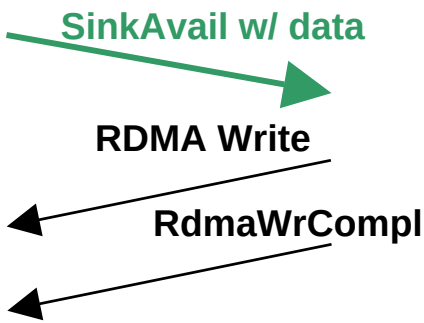
SDP Data Transfer



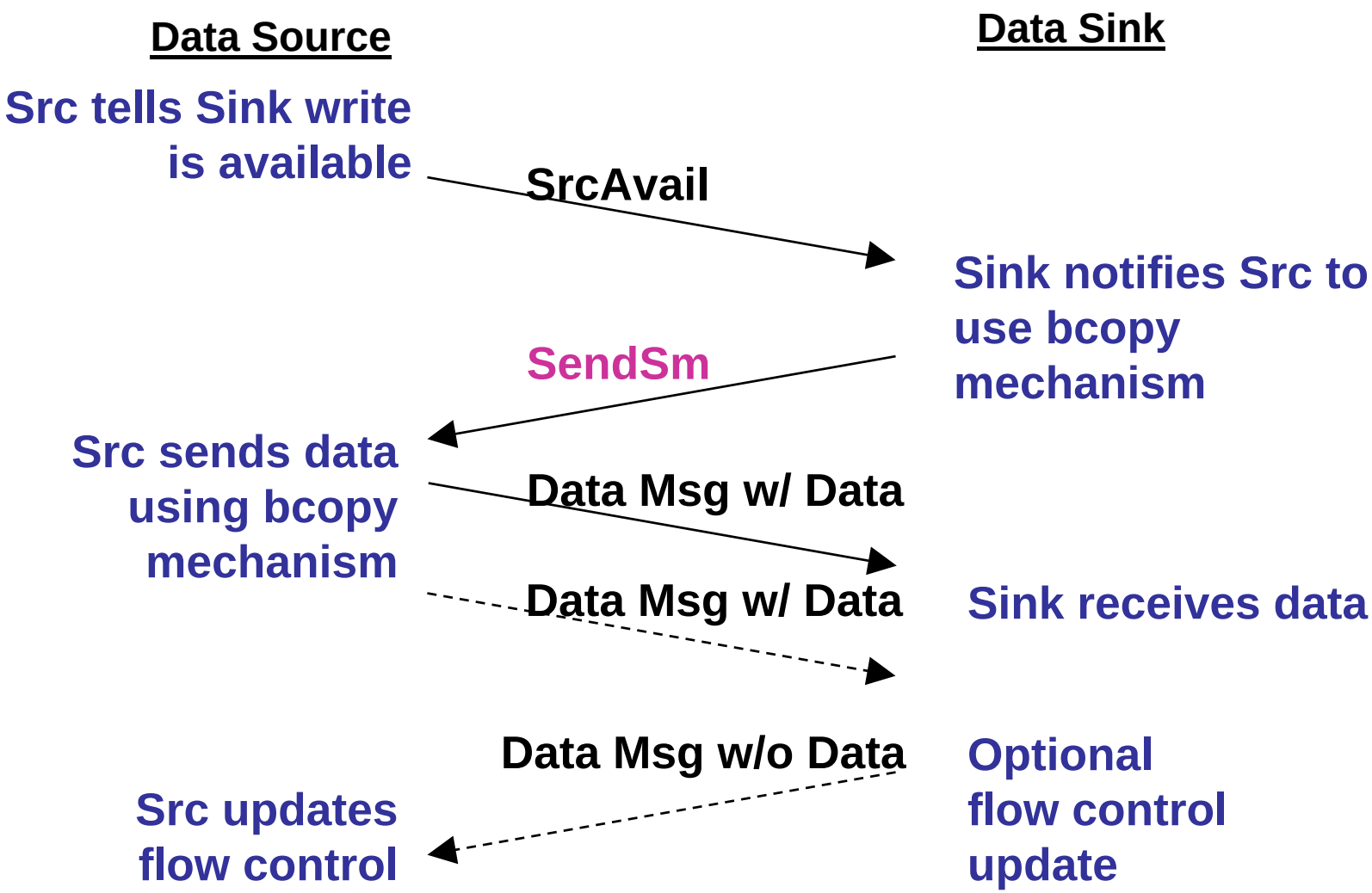
Notes:

- Typically short commands, medium to long replies
- Typical Transfer Mechanisms
  - Command – bcopy
  - Reply – bcopy or zcopy
- Transaction Transfer Enables
  - Fewer messages
  - Lower threshold for zcopy

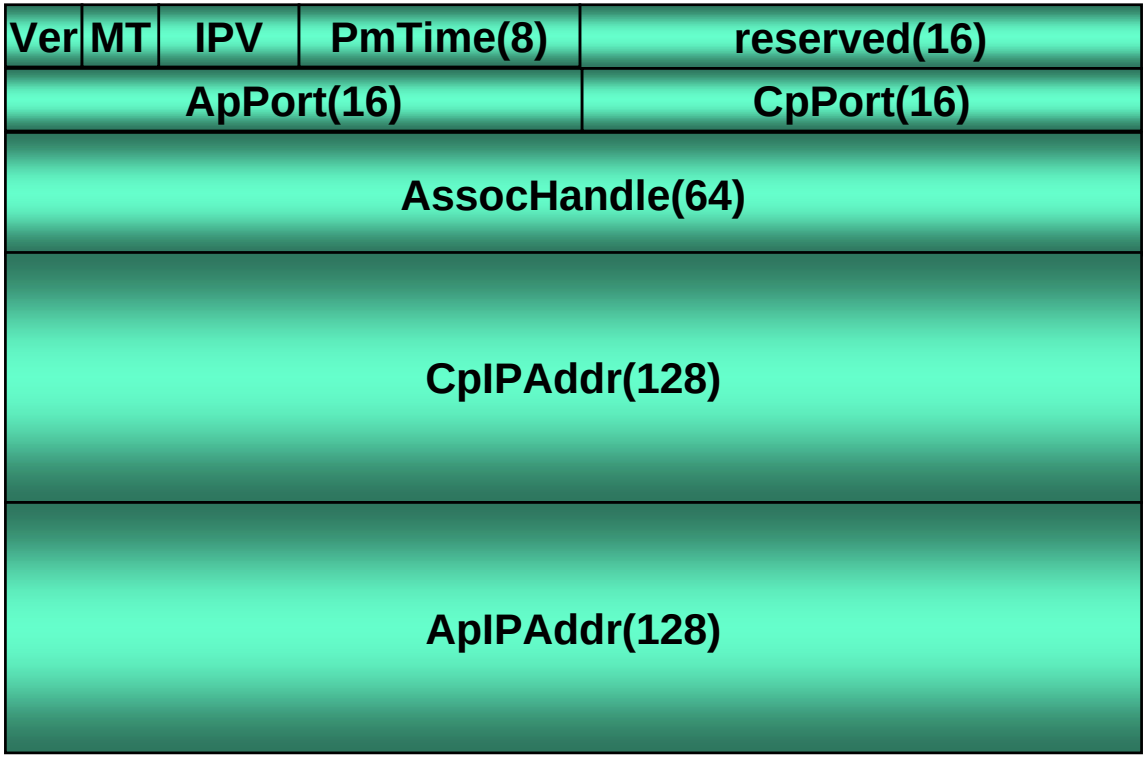
SDP Transaction



Data Transfer Mechanisms: Forcing use of Bcopy

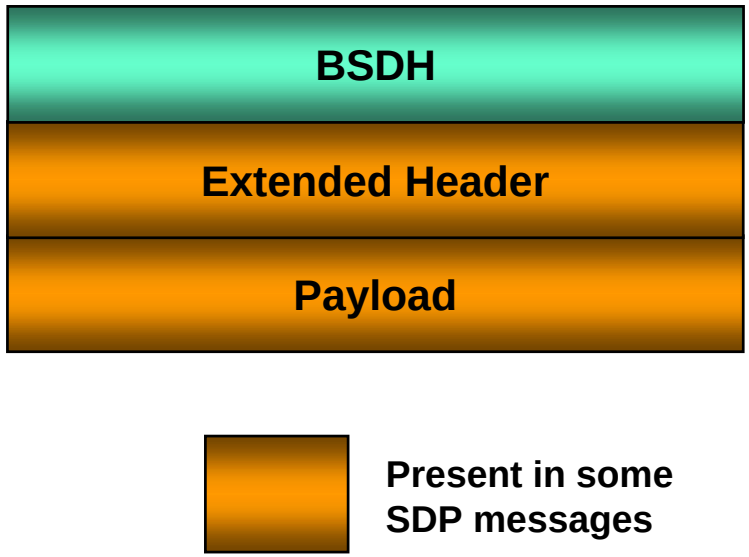


SDP Port Mapper Packet Format



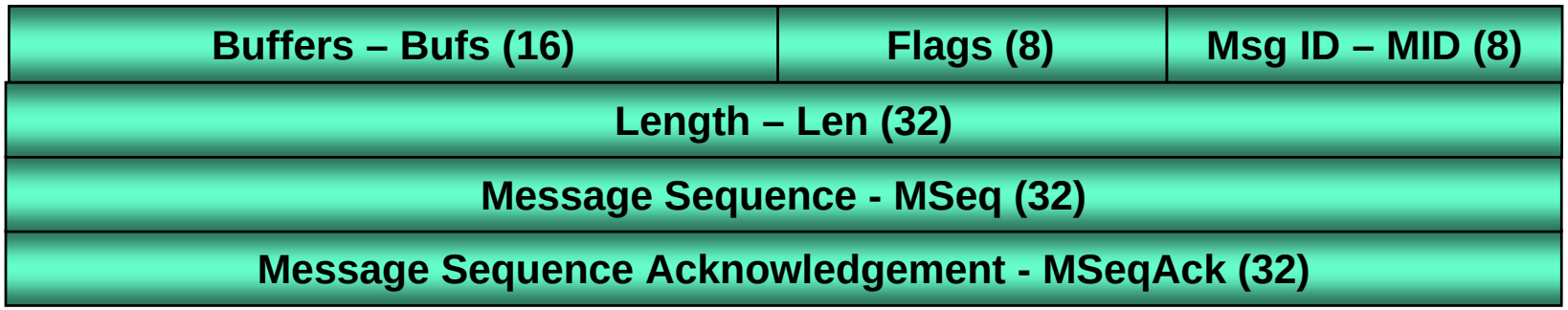
- Ver – Protocol version number
- MT – Message Type (PMRequest, PMAccept, PMDeny, PMAcknowledge)
- IPV – IP protocol version (either IP version 4 or IP version 6)
- PmTime – Amount of time the PM Client can cache the returned SDP Address
- ApPort – Accepting Peer Port
- CpPort – Connecting Peer Port
- AssocHandle – Connecting Peer's Association Handle for a specific request
- CpIPAddr – Connecting Peer's IP Address. If IPv4, only 32 bits are valid
- ApIPAddr – Accepting Peer's IP Address. If IPv4, only 32 bits are valid

SDP Packet Format & Connection Setup



- Base header is in all messages (BSDH), includes:
  - Private buffer flow control information
  - Length of message
  - Flags
- Extended Headers for some message types
- ULP Payload for some message types

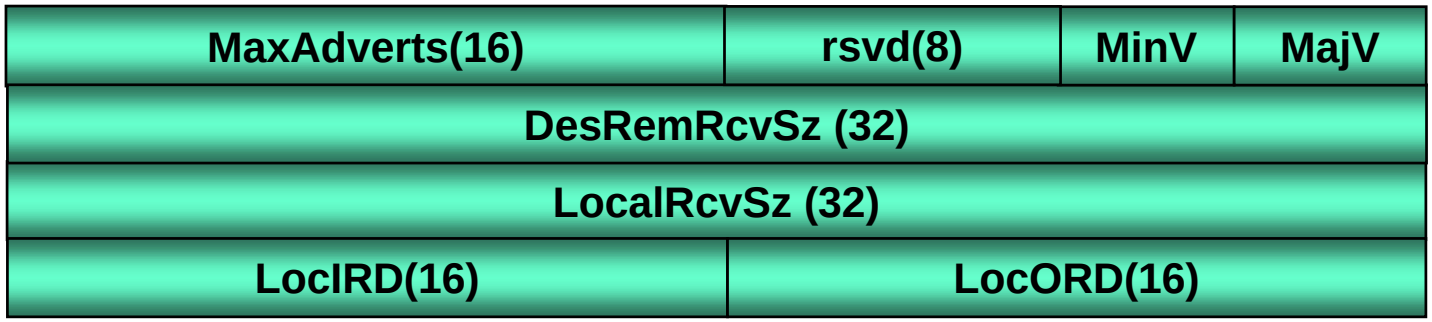
Base Socket Direct Header (BSDH)



- Message ID Types:
  - Connection Management
    - Hello, HelloAck
    - Disconn, AbortConn, SuspComm, SuspCommAck
  - Data Transfer
    - Data, SrcAvail, SrcAvailCancel, RdmaWrCompl, SinkAvail, SinkAvailCancel, SinkCancelAck, RdmaRdCompl
  - Special
    - SendSm, ModeChange, ChRcvBuf, ChRcvBufAck
- Flags
  - OOB present
  - OOB pending
  - Switch to pipelined mode
- Bufs
  - Number of currently posted Receive buffers
- Length, in bytes, of this message
- Message Sequence (Number)
- Message Sequence Acknowledgement
  - Message sequence number of the last message received by the socket sending this message
- OOB Data
  - If OOB present flag is set, last byte of **payload** contains the OOB data.

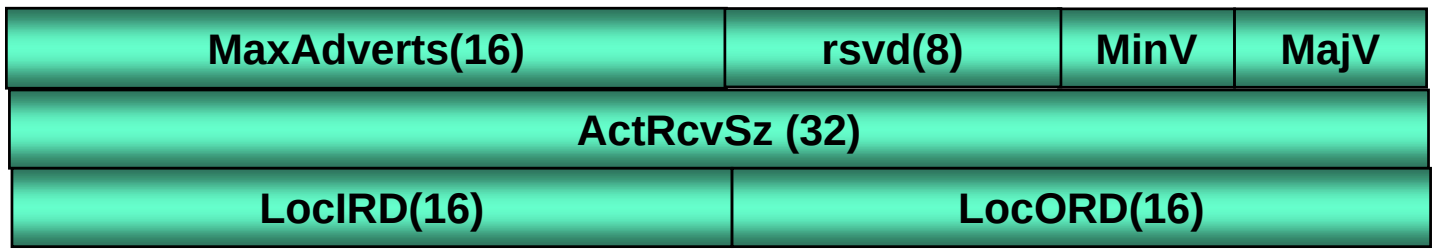
Connection Setup Messages

Hello message – Initialize remote peer state



- MajV, MinV – Major and minor protocol version numbers
- MaxAdverts – Maximum number of RDMA advertisements remote peer can send
- LocalRcvSz – Size of local receive private buffers, in bytes
- DesRemRcvSz – Desired size of remote peer's receive private buffers, in bytes
- LocalPort – Local TCP port number
- LocIRD – Local In-bound RDMA Read Requests allowed
- LocORD – Local Out-bound RDMA Read Requests allowed

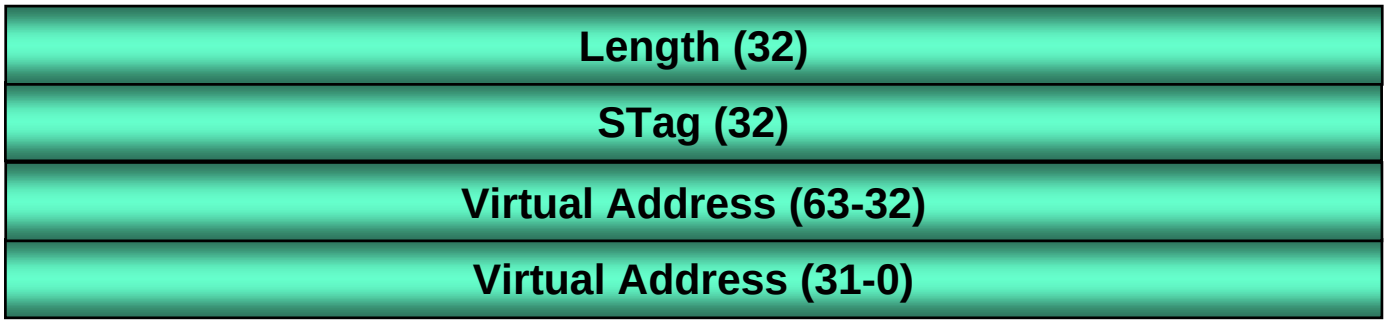
HelloAck message – Initialize local peer state



- ActRcvSz – Actual size of the Remote Peer's receive private buffers, in bytes

Source Available Header

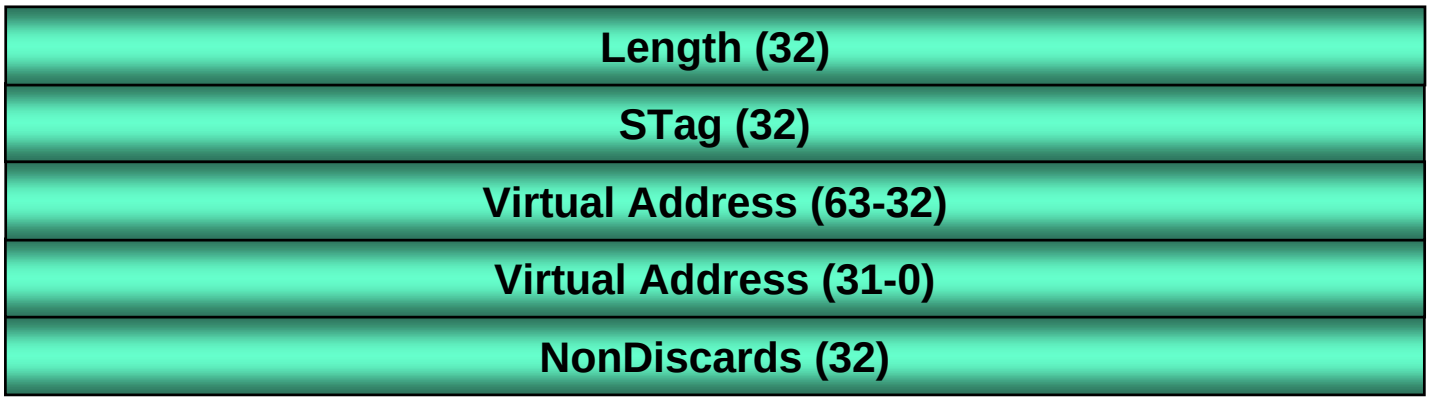
Used by the Data Source to tell the Data Sink of the availability of a RDMA Buffer that can be transferred through a RDMA Read.



- Length
  - Length, in bytes, of the RDMA buffer.
- Virtual Address
  - Start of RDMA buffer
- STag
  - STag for RDMA buffer

Sink Available Header

Used by the data sink to inform the data source of the availability of an RDMA buffer that can be filled through a RDMA Write operation.



- Length - Size, in bytes, of the receive RDMA buffer represented by the VA and STag.
- Virtual Address - Start of RDMA buffer
- STag - STag for RDMA buffer
- NonDiscards – Sent by the Data Sink. It counts the number of ULP payload carrying messages that have arrived that also did not cause a previously sent SinkAvail Message to be discarded.

RDMAs and Completion Headers

RDMA Read or Write – RDMA Reads and Writes do not carry an SDP header.

RDMAWrComp - Tells destination how much data was written.

- Length - Size of data transferred through the preceding RDMA Write(s).

RDMARdComp - Tells destination how much data was read.

- Length - Size of data transferred through the preceding RDMA Read(s).

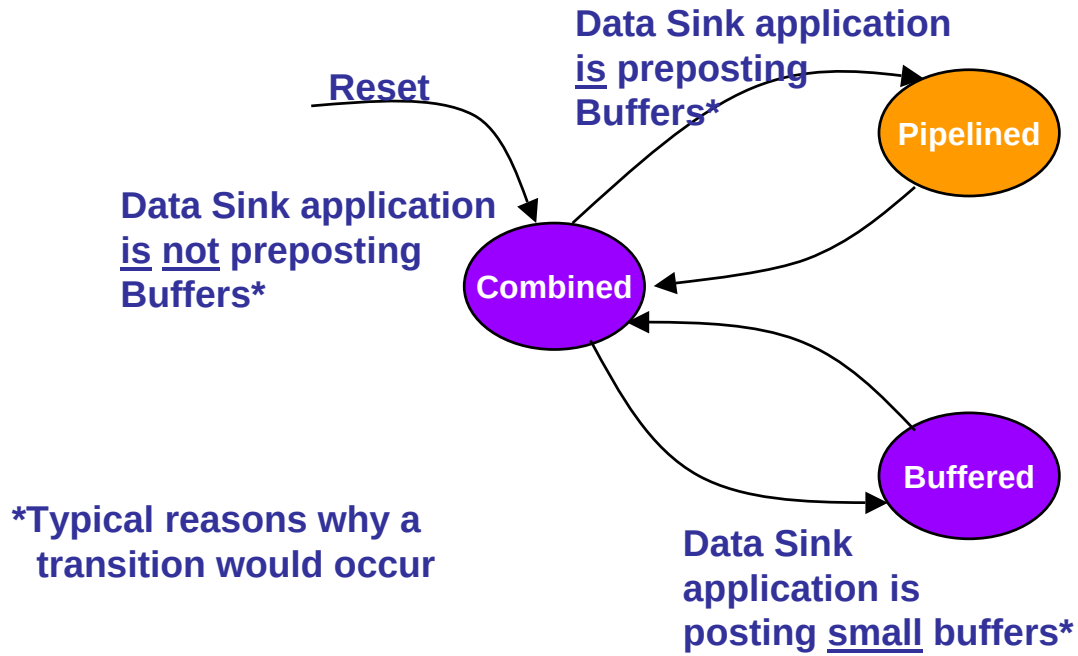
SDP Modes

- Modes restrict which data transfer mechanisms can be used
  - Bcopy data transfer mechanism is used in all modes when the ULP buffer at the Data Source is smaller than the Bcopy Threshold
- Buffered Mode
  - Used when receiver wishes to force **all** transfers to use the Bcopy mechanism
- Combined Mode
  - Used when receiver is not pre-posting buffers and uses peek/select interface (Bcopy or Read Zcopy, only one outstanding)
- Pipelined Mode
  - Highly optimized transfer mode – multiple write or read buffers outstanding, can use all data transfer mechanisms (Bcopy, Read Zcopy, or Write Zcopy)

SDP Mode Transitions

Notes:

- **Transitions initiated by ModeChange Message**
- **All modes enable Bcopy mechanism if I/O is below Bcopy_Threshold**



ModeChange - Tells destination of change in flow control method.

S	Mode	Reserved (28)

- S - Direction of mode change (send,1 vs. receive,0).
- MCH – New mode
  - 0 = BUFF_MODE
  - 1 = COMB_MODE
  - 2 = PIPE_MODE

SDP Buffering Issues

- Sockets API over either TCP or SDP implementations has a potential deadlock if transport buffering is not correctly implemented.
  - The deadlock scenario:
    - Both sides, at the same time, issue a send() that is larger than the amount of buffering in the fabric
    - Both sides do not post a receive buffer until the send() completes
    - All buffers in the fabric are filled, send can not complete on either peer
    - Deadlock
  - Solution:
    - Socket options SO_SNDBUF, SO_RCVBUF should be followed by transport protocol (TCP/IP or SDP/IB), and
    - Correctly written applications which transfer data per above scenario must never post a buffer larger than SO_SNDBUF + SO_RCVBUF
- Reserved private buffers for deadlock avoidance
  - Two private buffers reserved for forward progress under above condition
    - One buffer for SDP control messages (no ULP payload)
    - One buffer for SDP credit updates (no ULP payload)

Resizing Receive Private Buffers

- Excessive fragmentation overhead at Data Source
  - Problem: ULP at Data Source is consistently posting buffers larger then the receive private buffer size of remote peer (and smaller than Bcopy Threshold)
  - Solution: Request remote peer to change its receive private buffer size

ChRcvBuf – Send by data source to data sink, to request a change in the size of new receive buffers.

- DesSz - Desired size of new receive buffers.

ChRcvBufAck – Sent by data sink to data source, to inform the latter of the actual size of new receive buffers.

- ActSz - Actual size of remote peer's receive buffers (may not have changed).

Additional SDP Features

- Socket cloning (duplication) support
  - SuspComm & SuspCommAck message to flush all data transfers
  - Close the connection
  - Open new connection in new address space.
- Canceling data buffer advertisements
  - SrcAvail cancellation - SrcAvailCancel, SendSm SDP messages
  - SinkAvail cancellation - SinkAvailCancel, SinkCancelAck SDP messages

Comparison with InfiniBand SDP

- SDP is on iWARP over TCP, rather than Reliable Connected RDMA on InfiniBand
  - Semantics of both RDMA infrastructures are similar, except iWARP Invalidate capability
    - Invalidate for SDP on iWARP is optional - receiver must support the Remote Peer not using the invalidate.
    - This allows interoperability with SDP on InfiniBand
- SDP on iWARP can cross the Internet
  - SDP Port Mapper solves crossing multiple subnets
  - SDP on InfiniBand is limited to a single subnet
- SDP on iWARP and SDP on InfiniBand are designed to be interoperable
  - SDP Messages after connection setup are exactly the same
    - Including SDP Modes, Data Transfer Mechanisms, Socket cloning, Connection Teardown
  - SDP connection setup is different (next slide)

SDP Connection Setup – SDP/iWARP Compared with SDP/InfiniBand

- SDP over iWARP Connection setup is different
  - Initial connection is setup with TCP/IP, not as an InfiniBand Reliable Connection (RC)
  - Hello and HelloAck Messages are slightly different
    - IP addresses, IP version, and TCP port number are removed
      - Redundant due to the TCP/IP header
    - Maximum Zcopy Buffers to Advertise is 65535, InfiniBand is 255
      - SDP over iWARP must go longer distances, thus the bandwidth/delay product required to fill the network could potentially require a larger number of pre-posted buffers.
    - LocIRD and LocORD added to SDP over iWARP
      - InfiniBand RC negotiated them as part of InfiniBand connection setup. iWARP does not, thus LocIRD/LocOrd are negotiated at the SDP layer.