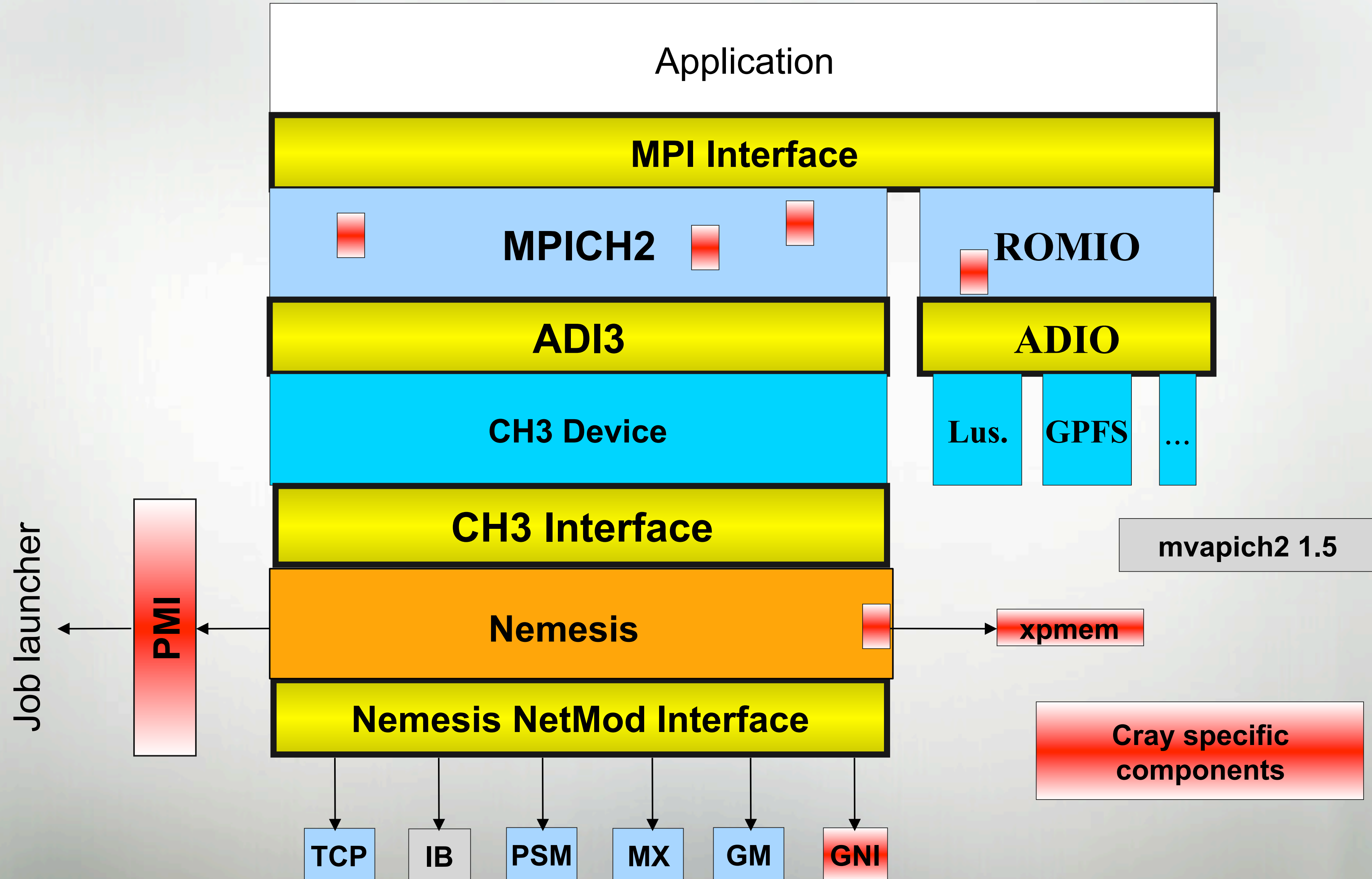
A uGNI-Based MPICH2 Nemesis Network Module for Cray XE Computer Systems

Howard Pritchard and Igor Gorodetsky
Cray, Inc.

Cray User Group Conference 2011

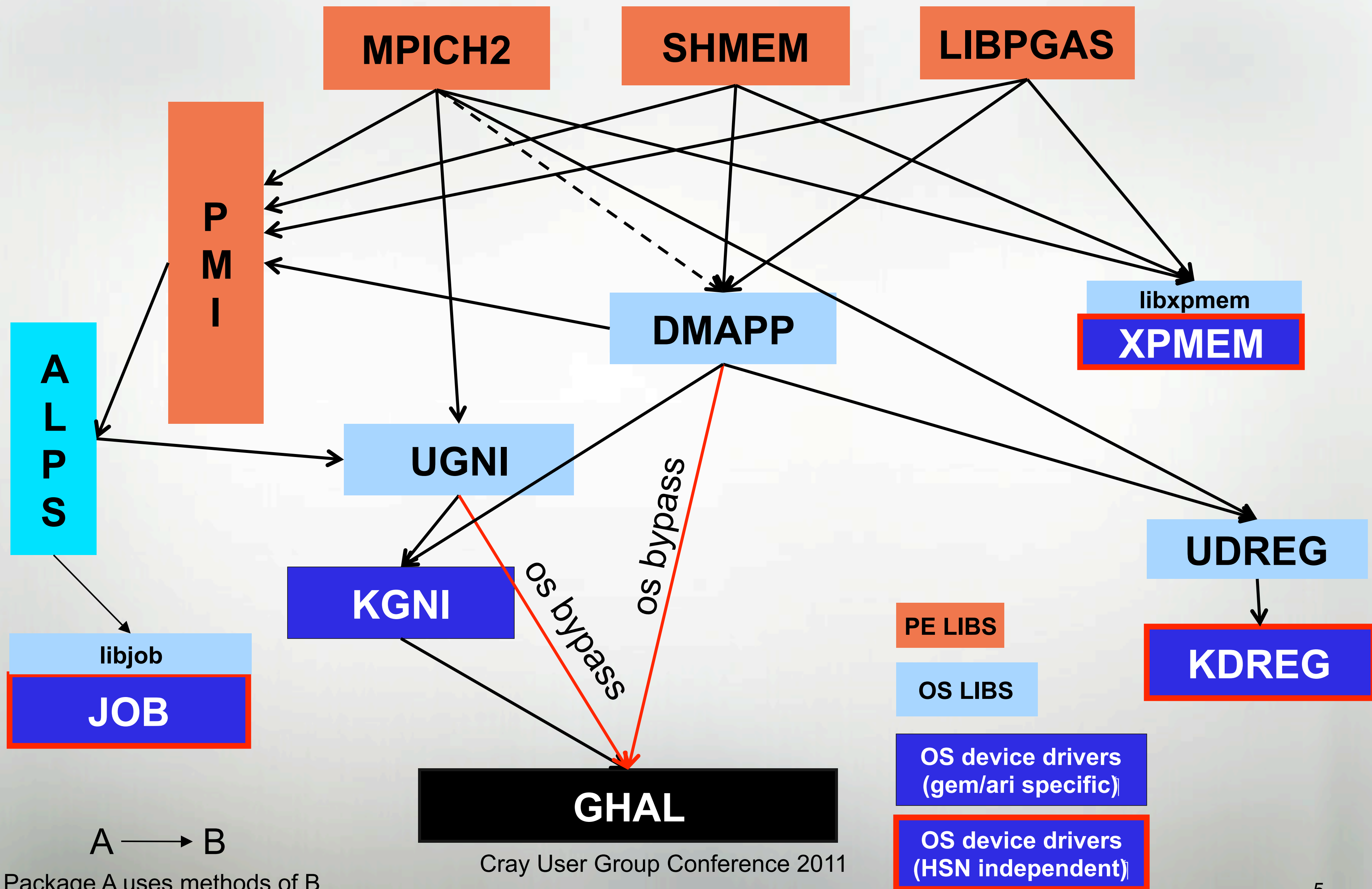
MPICH2 Nemesis

MPICH2 on Gemini/Aries



GNI Software Stack

MPICH2 in Gemini Software Stack



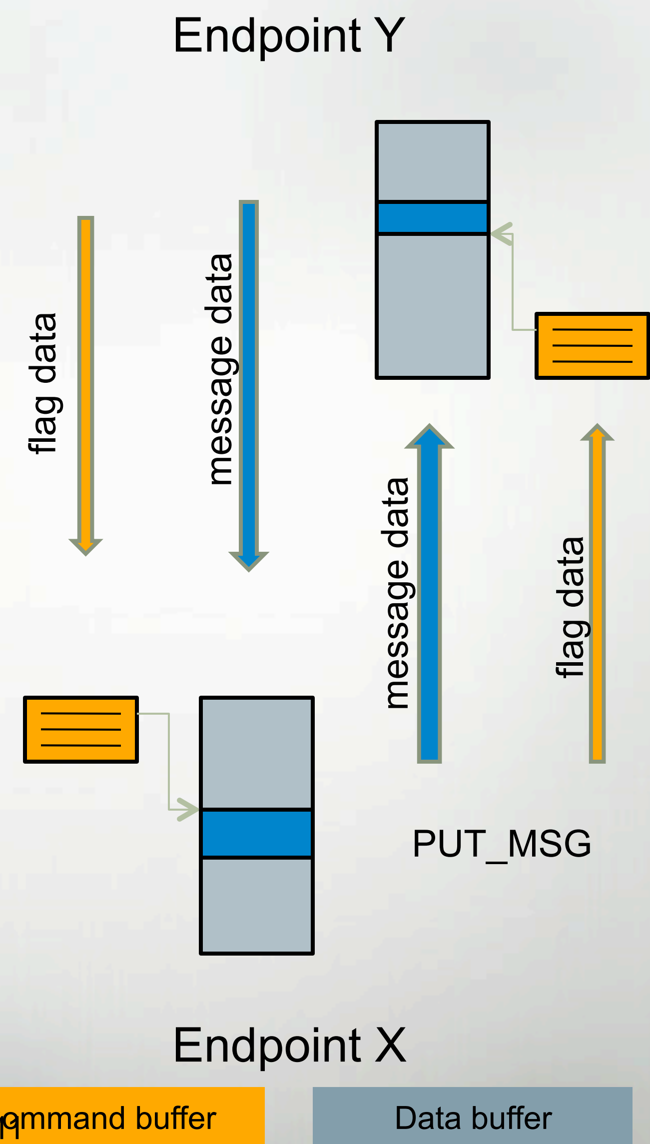
uGNI Netmod

How MPICH2 Uses GNI – Key Concepts

- A connection oriented approach based on GNI SMSG mailboxes is used
  - Lowest latency, highest message rates
  - Reliable connections, can ride through network faults
- Characteristics of Gemini memory registration hardware influenced MPICH2 GNI Network Module (Netmod) design.
- All network transactions are tracked. There is clean separation between data transfers and control messages. No fire-and-forget. This makes fault tolerance support much simpler.

How MPICH2 Uses GNI –SMSG Mailboxes

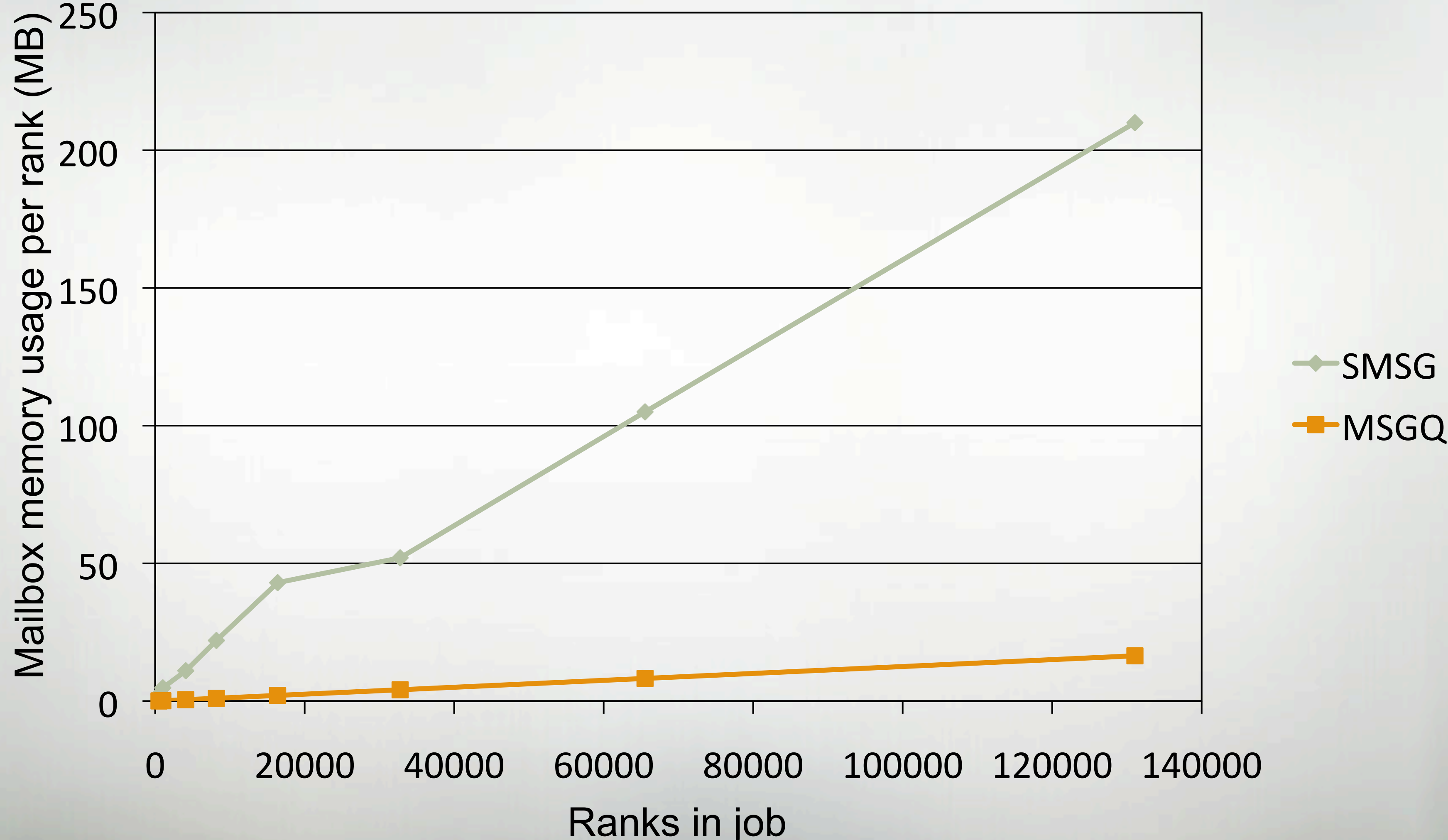
- Uses DSMN hardware in Gem/Ari
- Messages delivered in order even though ‘adaptive’ routing is used
- Tolerant to transient network faults
- FLOW CONTROL.   If the receiver stops dequeuing messages, sender runs out of credits and stops sending. No polling remote variables, queue overruns, etc.
- MPICH2 and GNILND (Lustre, DVS, etc.) share same mailbox code
- Memory per mailbox controlled by application. It can be small ~1000 bytes or so.
- User-space has option to use shared mailboxes (MSGQs) to reduce memory footprint.



How MPICH2 Uses GNI – SMSG Mailboxes (2)

- By default, connections (mailboxes) are established dynamically using the scalable, but low performance datagram (BTE_SEND) path.
- Mailboxes are normally mapped to large pages to reduce TLB pressure when processing messages from many different mailboxes. For better performance a subset of mailboxes/rank will soon be placed on DIE0 memory if user chooses.
- A RX Completion Queue (part of DSMN) is used to lookup which mailbox to check for incoming messages. If the CQ becomes overrun, app doesn't die, just scan all the mailboxes.
  - Some users very much like this – the "I just want to get through this silly part of the code without dying or doing big rewrite" crowd
  - Some users don't like this because they'd prefer to die and figure out how to fix things rather than run slow.

GNI Max. Memory Usage for SMSG/MSGQ – full connectivity (24 ranks/node)



Day in the Life of an Inter-node MPI Message

- Eager Protocol
  - For a message that can fit in a GNI SMSG mailbox (E0)
  - For a message that can’t fit into a mailbox but is less than MPICH_GNI_MAX_EAGER_MSG_SIZE in length (E1)
- Rendezvous protocol (LMT)

Day in the life of Message type E0 (1)

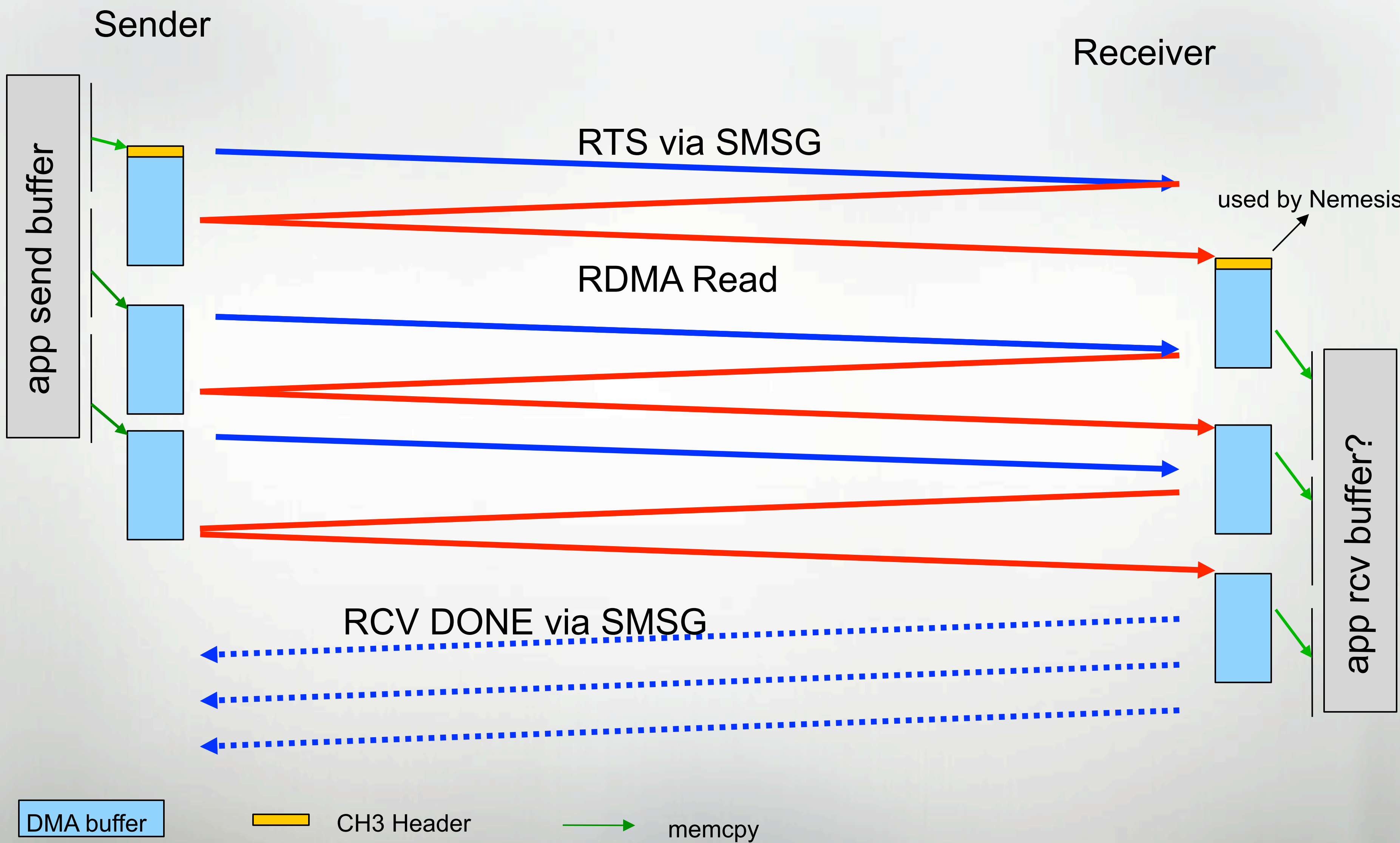
- Protocol for messages that can fit into a GNI Smsg mailbox
- The default varies with job size, although this can be tuned by the user to some extent

ranks in job	maximum bytes of user data
<= 1024	984
>1024 && <=16384	472
> 16384	216

Day in the Life of an E1 message

- For good performance, switching from an Eager protocol to Rendezvous at the small maximum messages sizes possible for GNI SMSG mailboxes is not acceptable, except for IMB, etc.
- For this reason, the GNI Netmod has a leave-the-data-at-the-source-but-send-the-header GET-based Eager protocol for messages too large to fit into a mailbox, but less than or equal to MPICH_GNI_MAX_EAGER_MSG_SIZE bytes

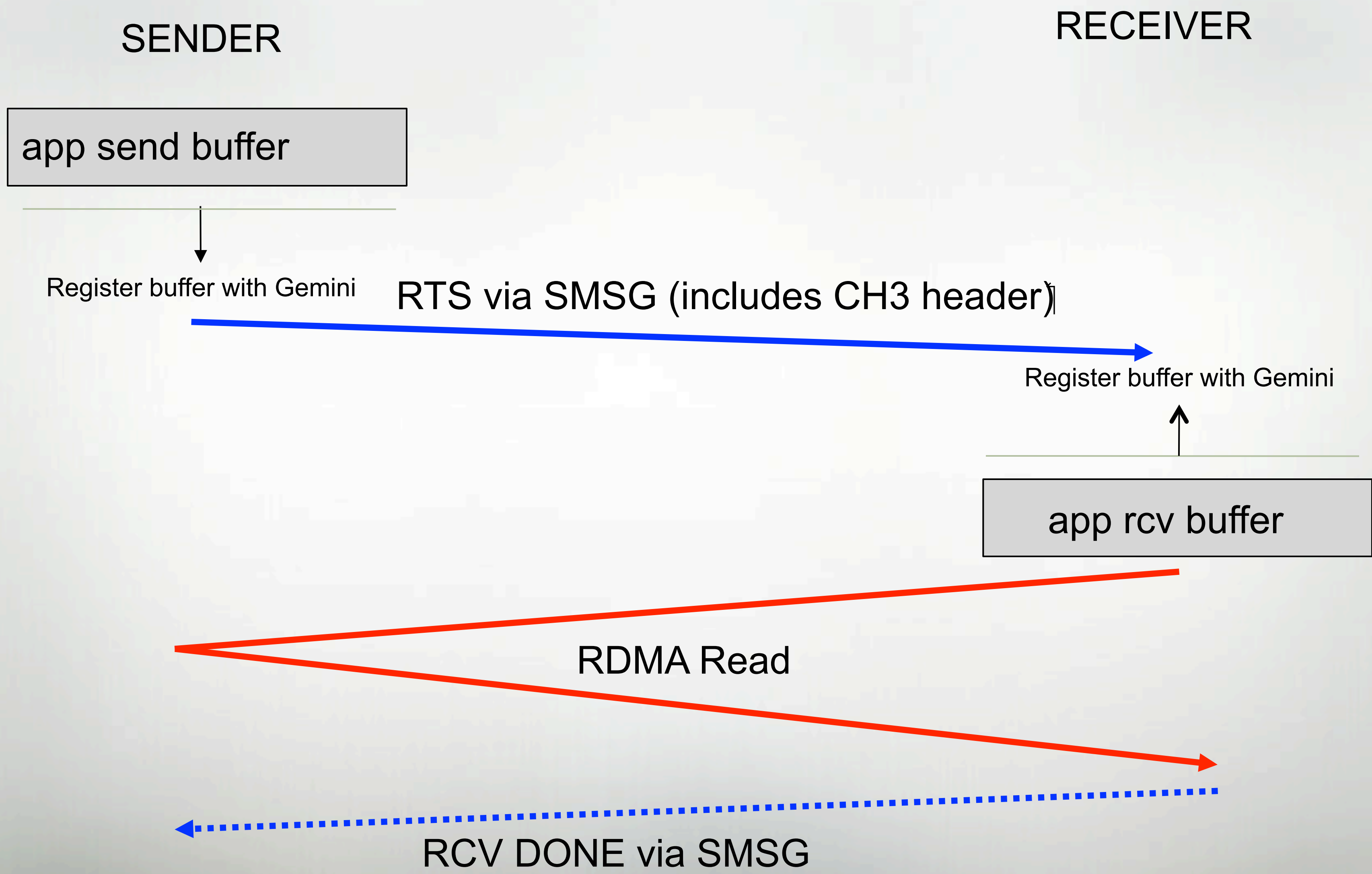
Day in the life of Message type E1 (2)



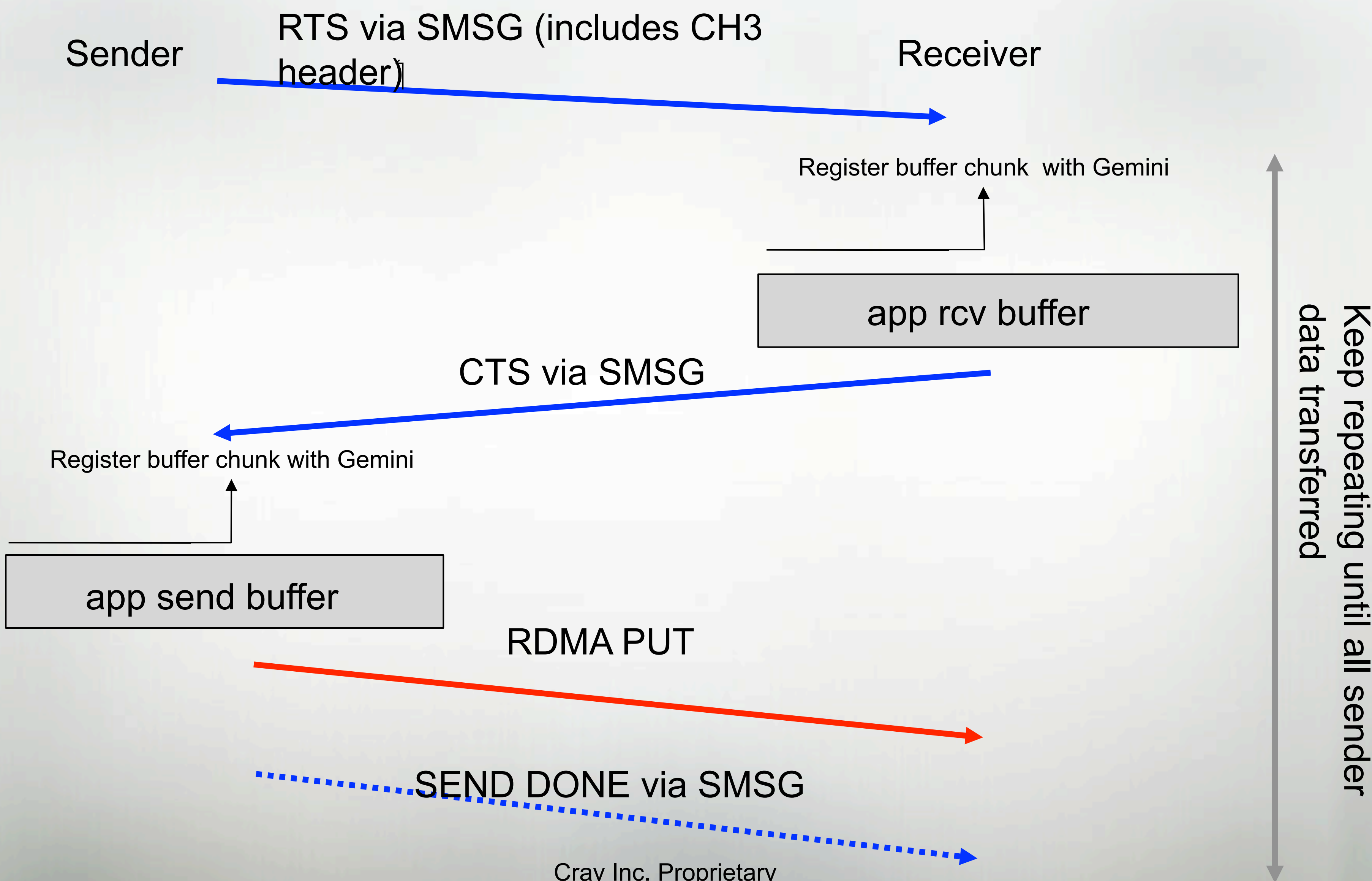
Day in the Life of Message type LMT

- *LMT* stands for Long Message Transfer.
- This is a rendezvous protocol. The Nemesis match engine has to have matched the receive with the send before an LMT begins
- Nemesis provides the infrastructure for RDMA style NICs like Gemini to make use of zero-copy without reinventing wheels
- The GNI Netmod makes use of this infrastructure, as does the XPMEM component of the shared memory part of Nemesis (intra-node transfers)
- Two methods are used by the GNI Netmod, depending on size of the message
  - RDMA read method (receiver pulls the data)
  - RDMA write method (max bandwidth)

Day in the life of an LMT using RDMA Read



Day in the life of an LMT using RDMA Write



Notes for LMT RDMA Paths

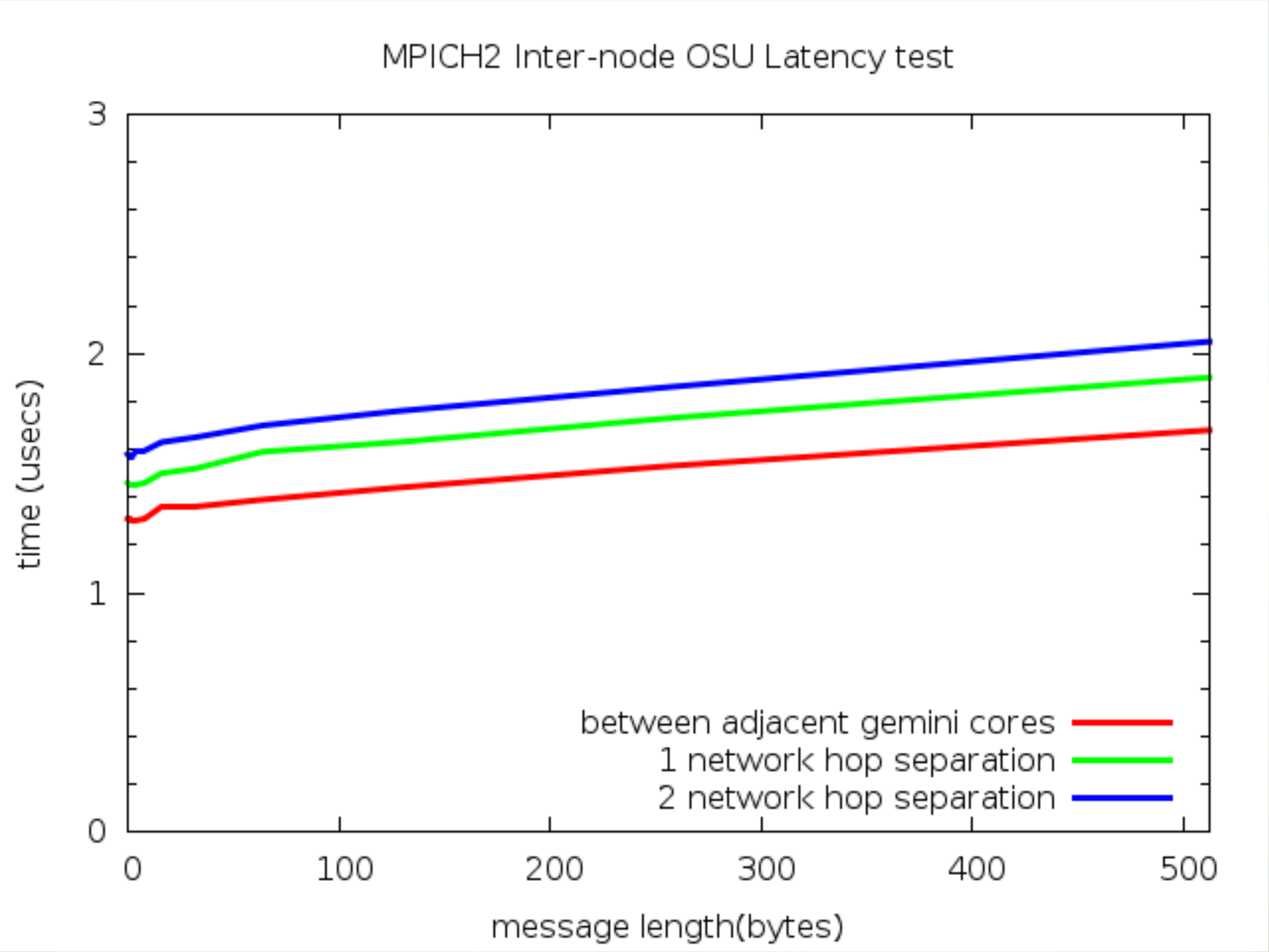
- RDMA Read path offers best opportunity with current MPICH2 to get some overlap of compute with communicate, at least for the sender
- There are alignment restrictions for source when using RDMA Read path
  - Dword aligned start addr
  - Integral number of dwords message length
- RDMA read delivers suboptimal network bandwidth utilization in the general alignment case for send and receive buffers
- RDMA Write offers highest bandwidth path, not sensitive to alignment of send and receive buffers
- Not possible to get much overlap of compute with communicate for the RDMA Write path with current MPICH2 software

Performance

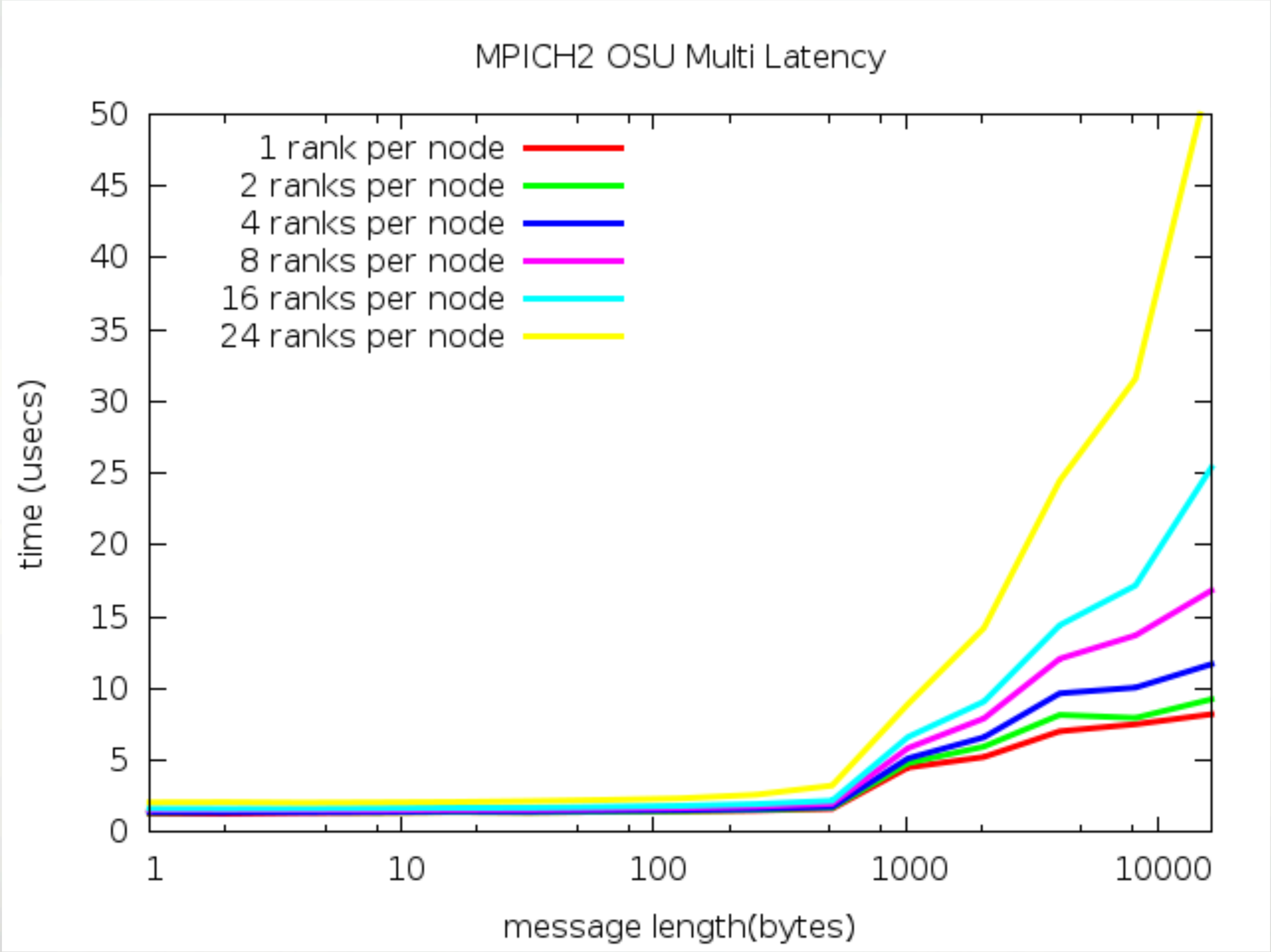
Performance Notes

- Tests were done on the following system
  - Cray XE with 2.0 GHz Magny Cours (12) – 24 cores per node – system
  - Cray Linux Environment (CLE) 3.1.61 and a pre-release MPT 5.3 (MPICH2)
- Not intended as advertising material for maximum possible performance (use 2.4 GHz processors for that)

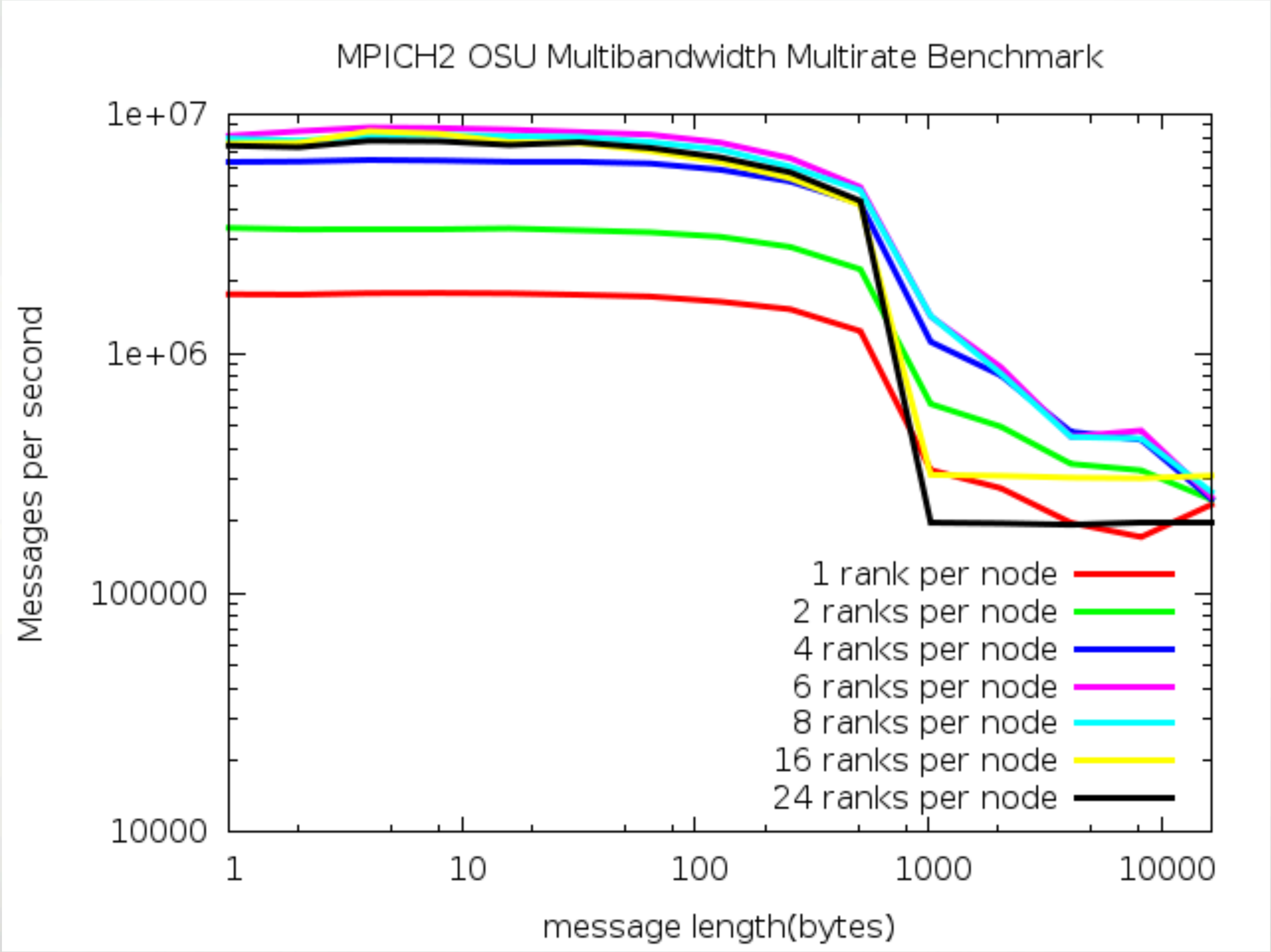
Single pair MPI Latency



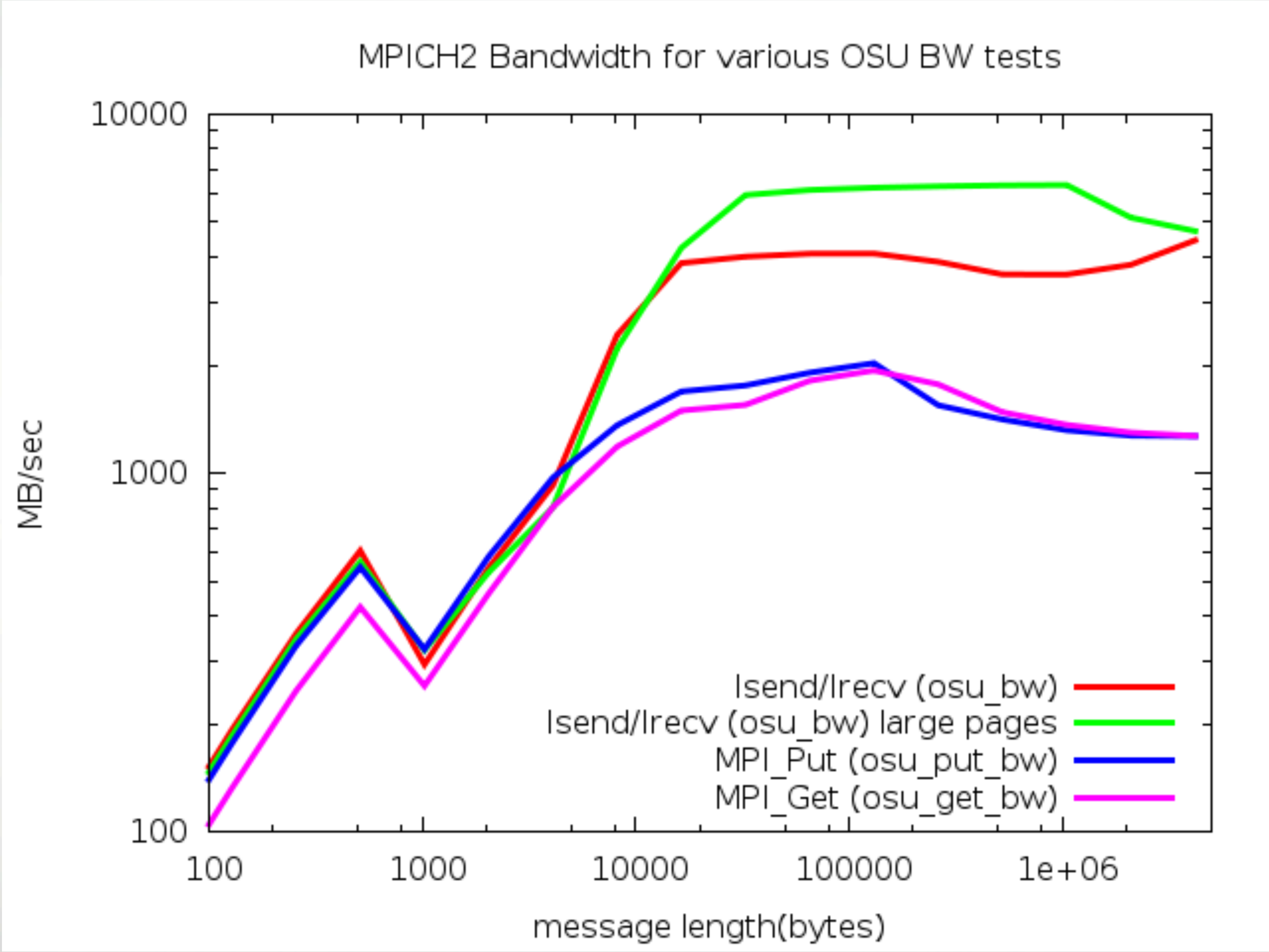
Multi-pair Latency



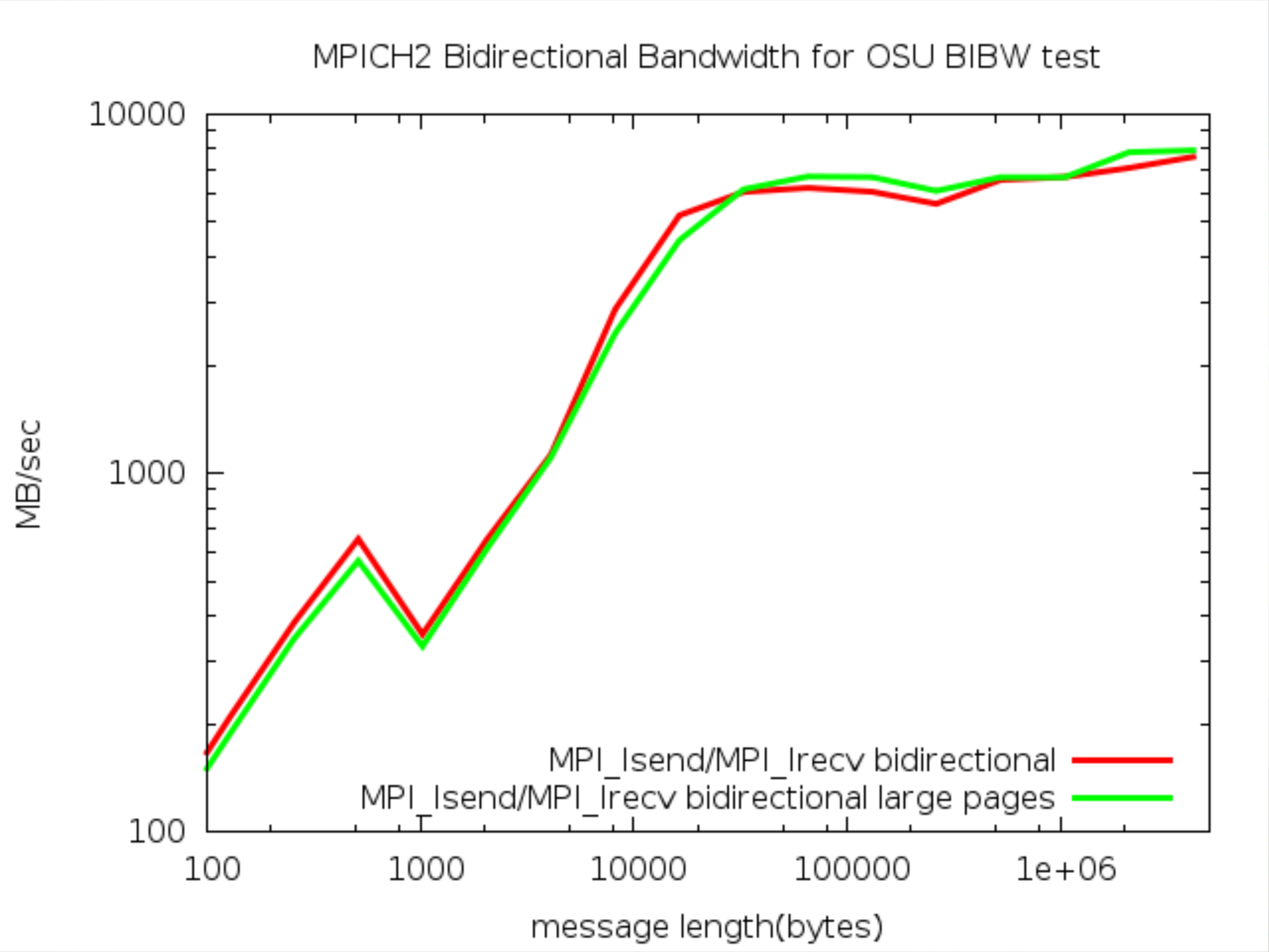
MPI Message Rate (2.0 GHz Magny Cours -12)



MPI Bandwidth using Different Protocols



MPI Bisection Bandwidth



Going Forward

- Checkpoint/restart support
- Improvements to support better overlap of communication with computation
- Improvements for short-vector MPI_Allreduce, etc.
- MPI-3 (long term)

Questions?

CRAY

THE SUPERCOMPUTER COMPANY