High Performance RDMA Protocols in HPC

Tim Woodall - NetQos

Galen Shipman - LANL

George Bosilca - UTK

Richard L. Graham - LANL

Arthur B. Maccabe - UNM

LA-UR-06-1268

RDMA - “issues”

- Explicit memory registration
  - Memory registration requires OS trap and other OS work linear in the number of pages (OS Bypass?)
  - Freeing doesn’t unregister memory (so we play dirty tricks)
- Operations must be fully described by initiator
  - Round trip required in order to allow applications to “receive” into arbitrary memory locations
- MPI places no restriction on what memory is used for Send/Receive
  - RDMA semantics don’t map naturally to MPI Send/Recv semantics

RDMA - “solutions”

- Copy In/Out
  - Explicit copies to/from registered buffers
    - What about zero copy?
- Leave Pinned (AKA - Registration Cache)
  - Register then RDMA, caching the registration for later
    - What about free()?
    - Does the application even reuse its buffers regularly?
- Pipeline Protocol
  - Register and RDMA concurrently
- Leave Pinned Pipeline
  - Same as the pipeline protocol only the registration is cached for later

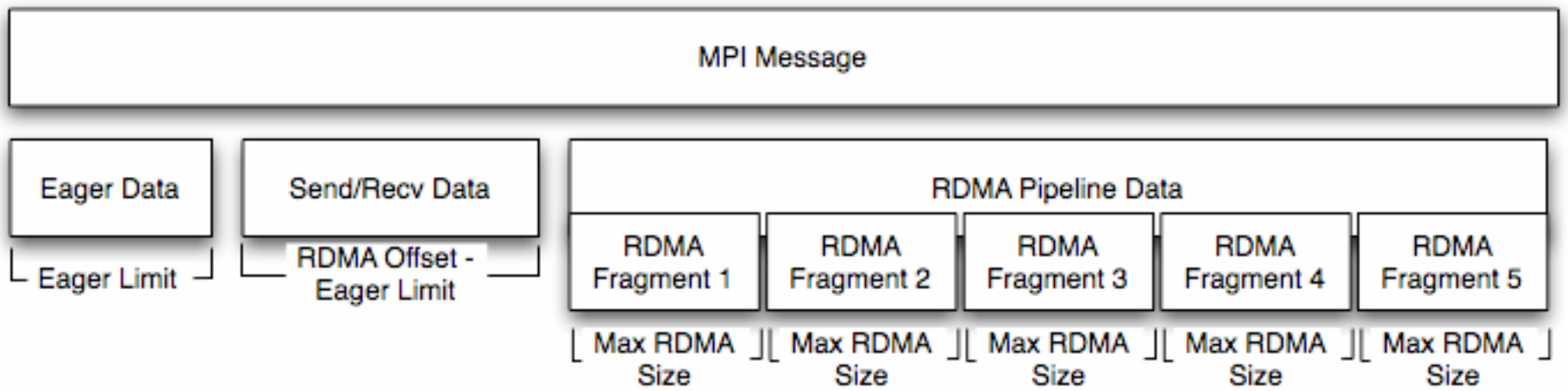
“Leave Pinned”

- For contiguous data
- Uses the Mpool to register the entire buffer up front and cache the registration via the Rcache
- Initiate a single RDMA read or write (per available RDMA BTL)
- Subsequent send/receive from the same user buffer can avoid registration by searching for the registration in the Rcache

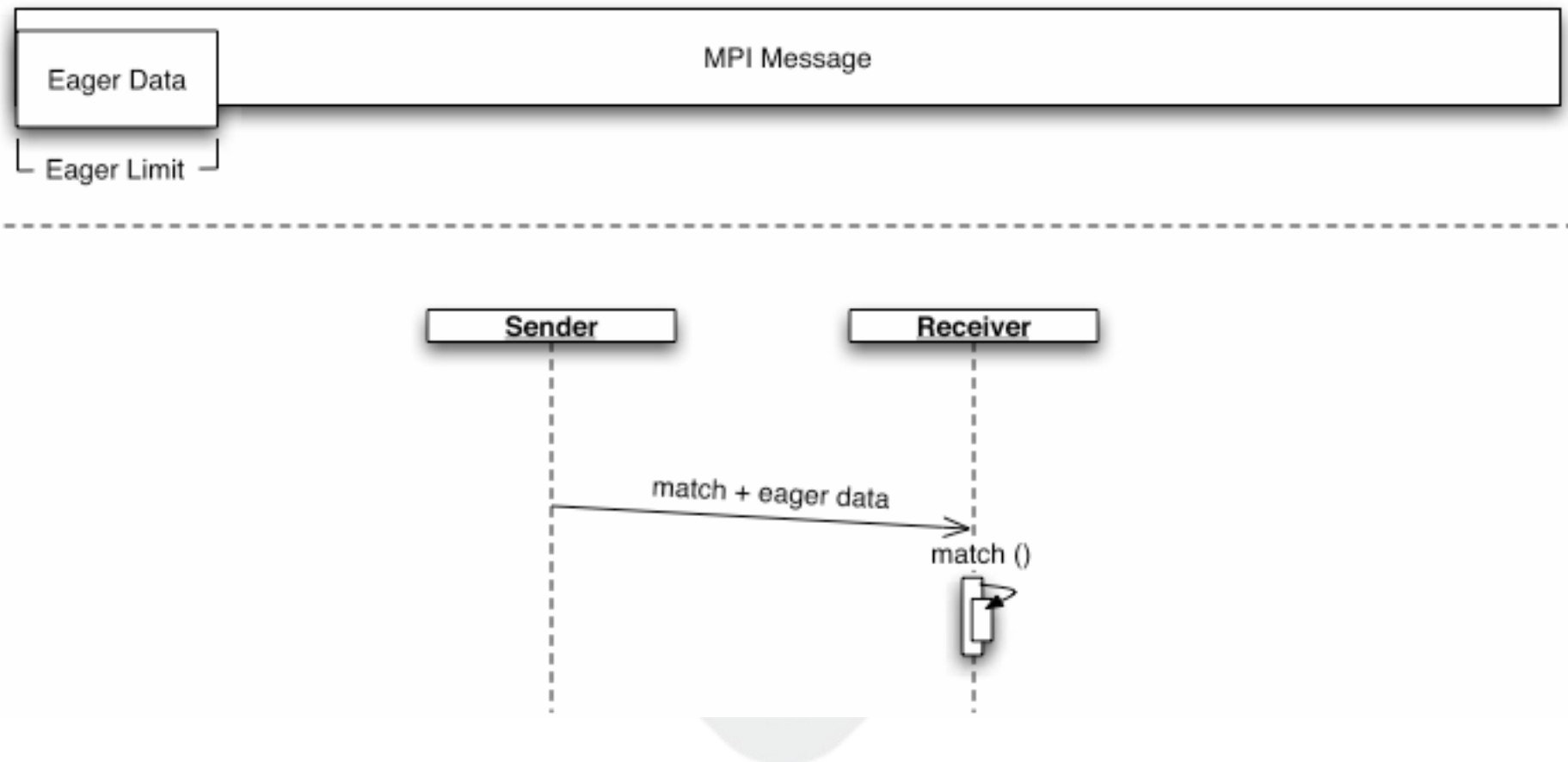
Pipeline Protocol

- Overlaps memory registration with RDMA operations
- Uses Mpool to register memory in chunks (BTL max RDMA size)
- Initiate multiple RDMA operations at once (up to BTL pipeline depth)
- Cover the cost of pipeline initialization through Copy In/Out protocol

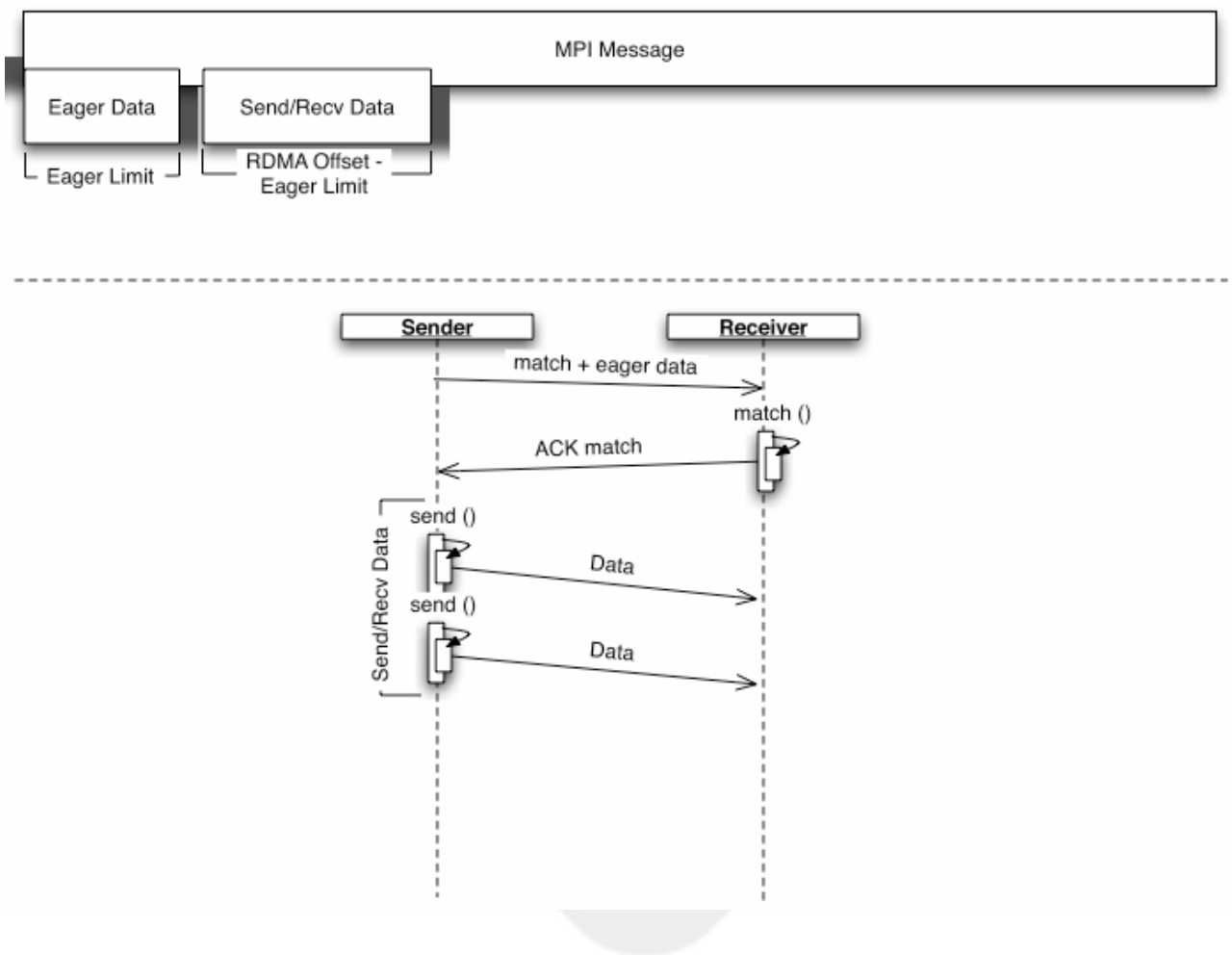
Pipeline Protocol - Message Layout



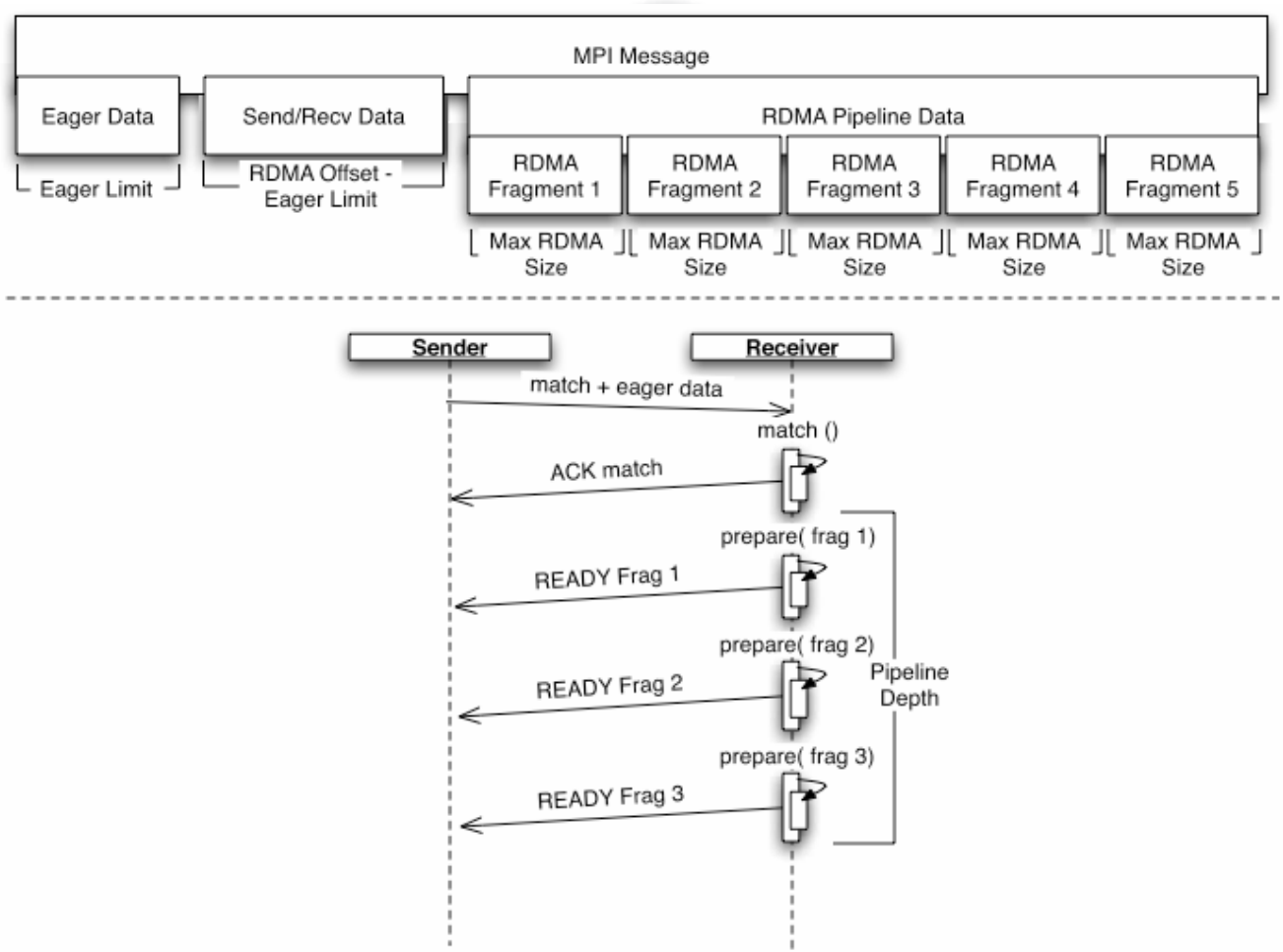
Pipeline Protocol - Eager+Match



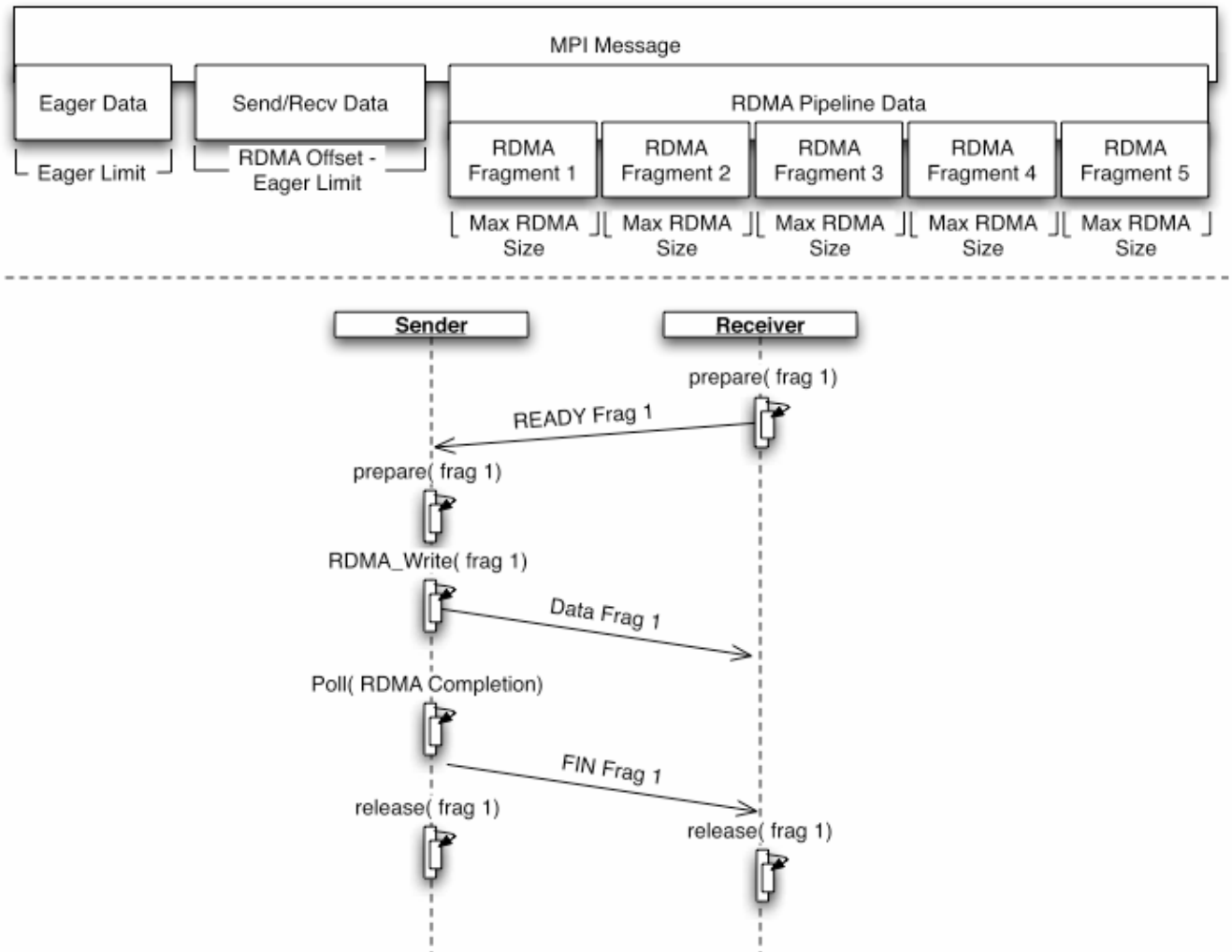
Pipeline Protocol - Send up to RDMA Offset



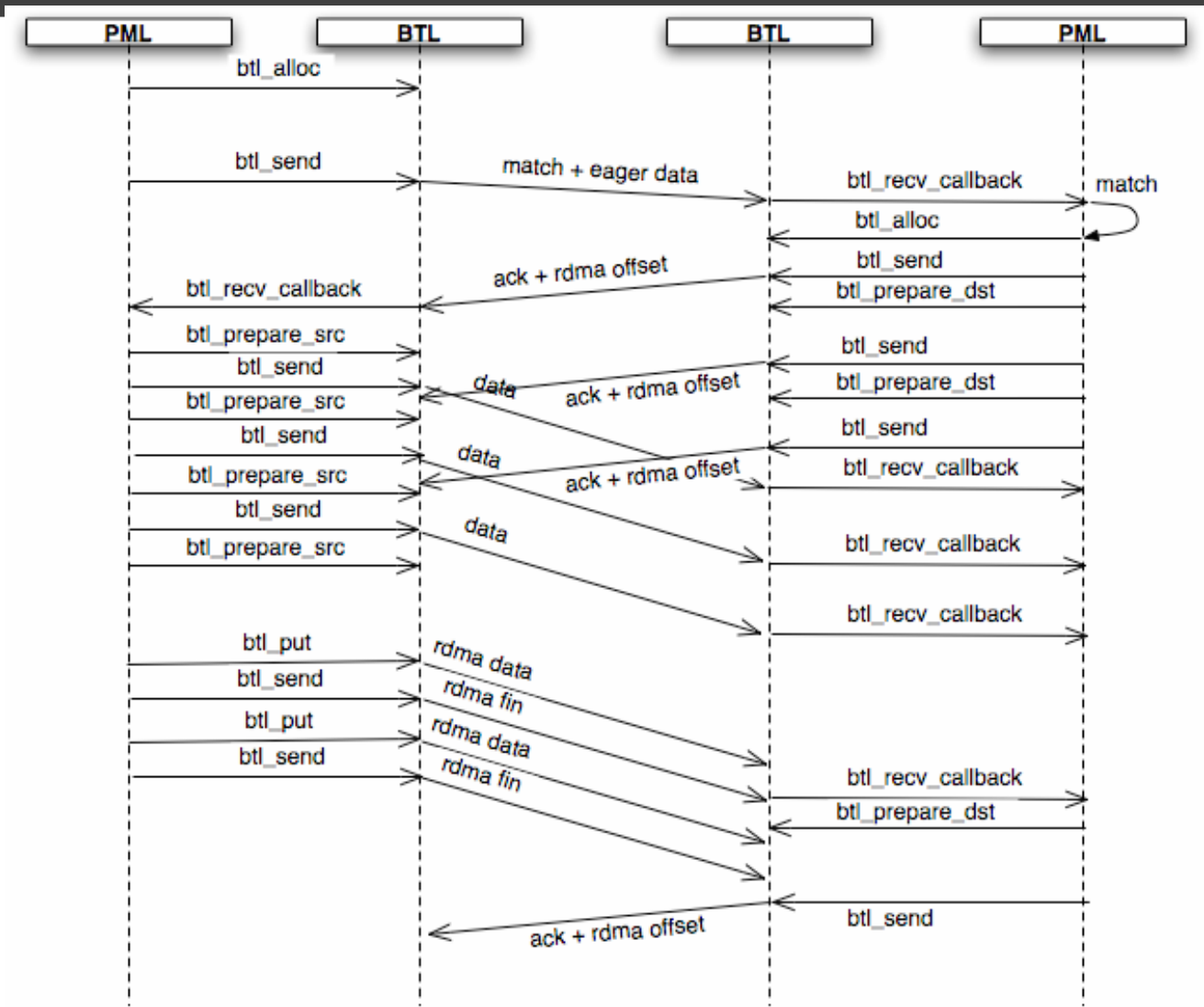
Pipeline Protocol - Register Receiver Side “chunks”



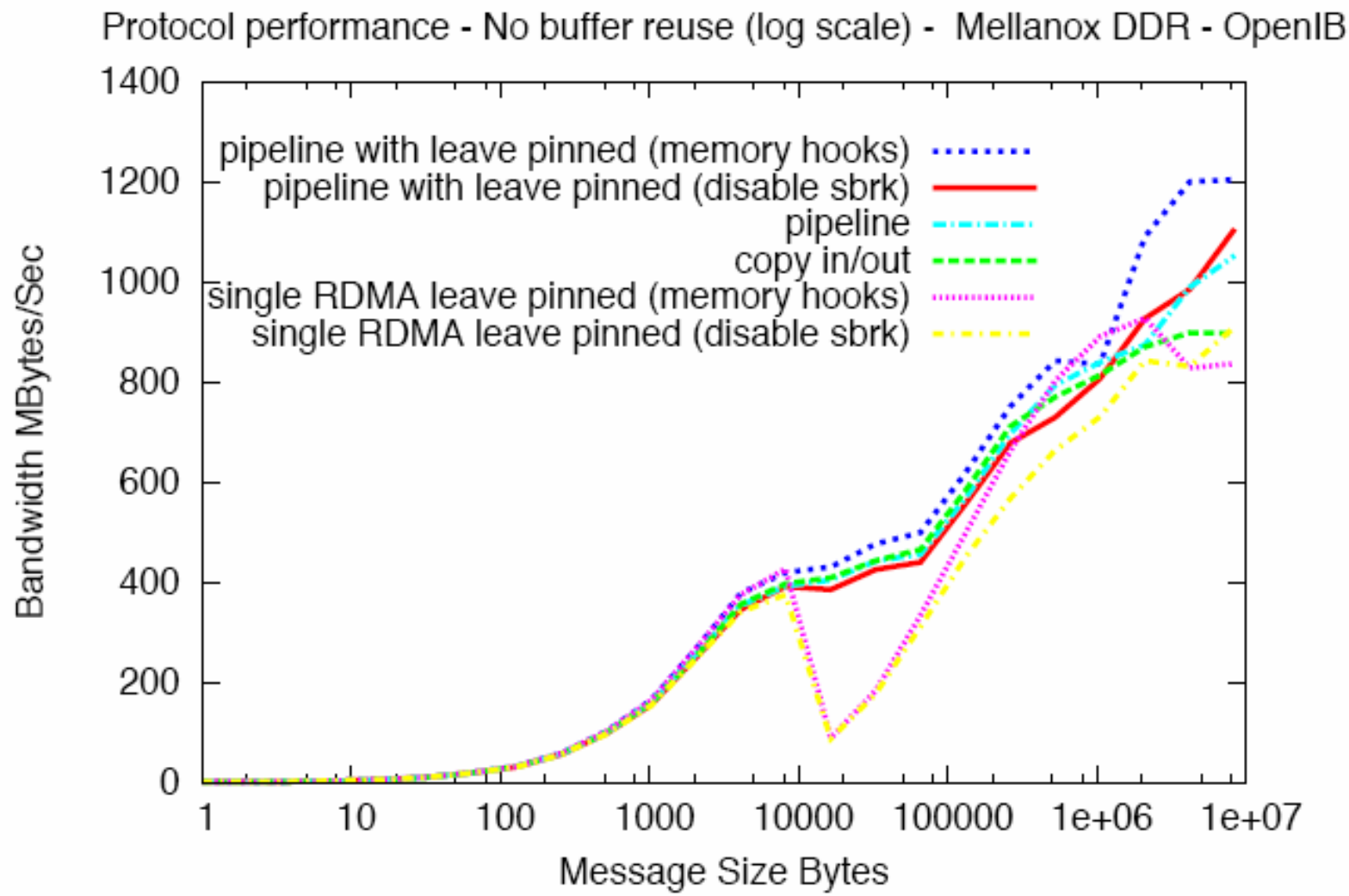
Pipeline Protocol - Receiver Register and RDMA



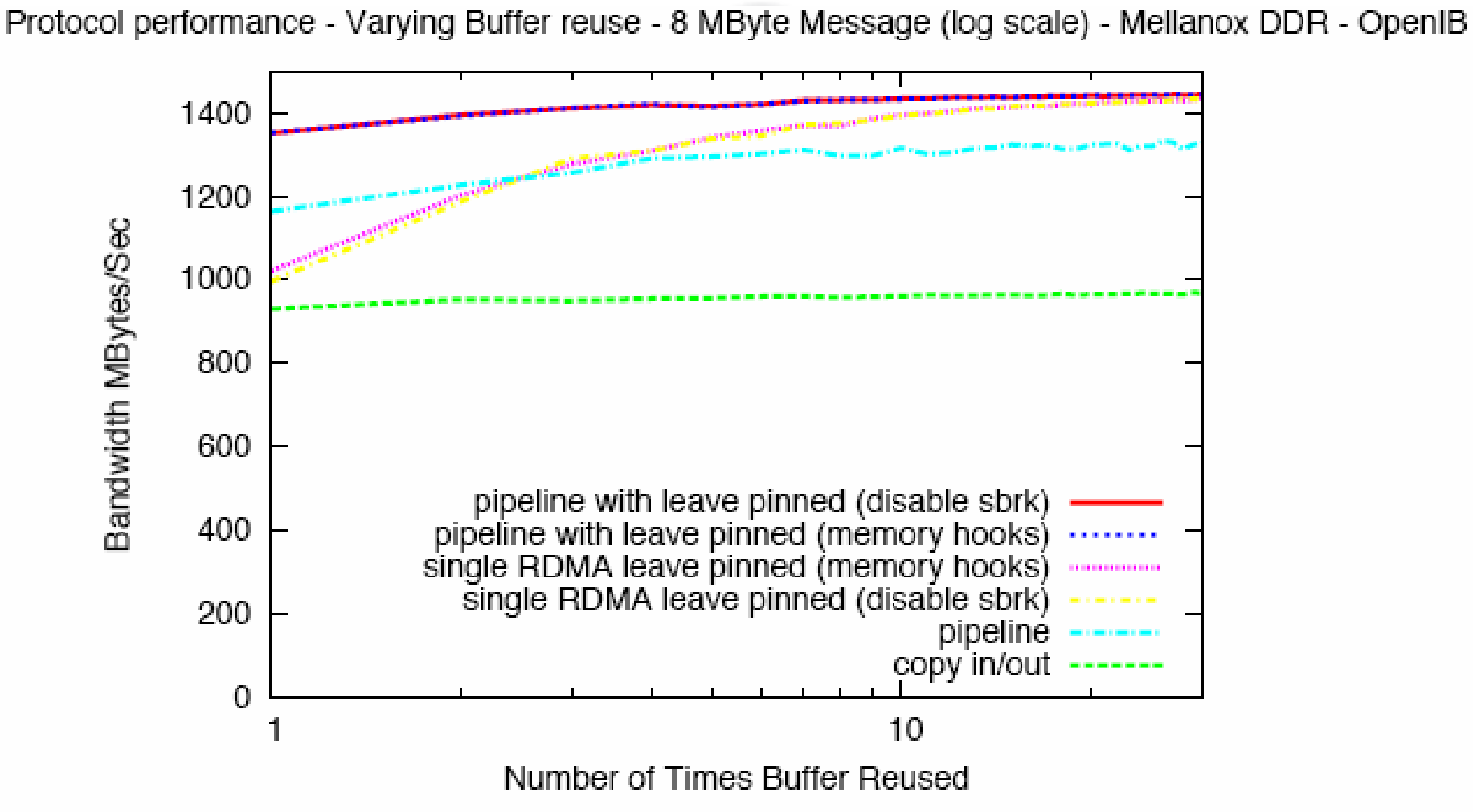
Pipeline Protocol - Timing Diagram



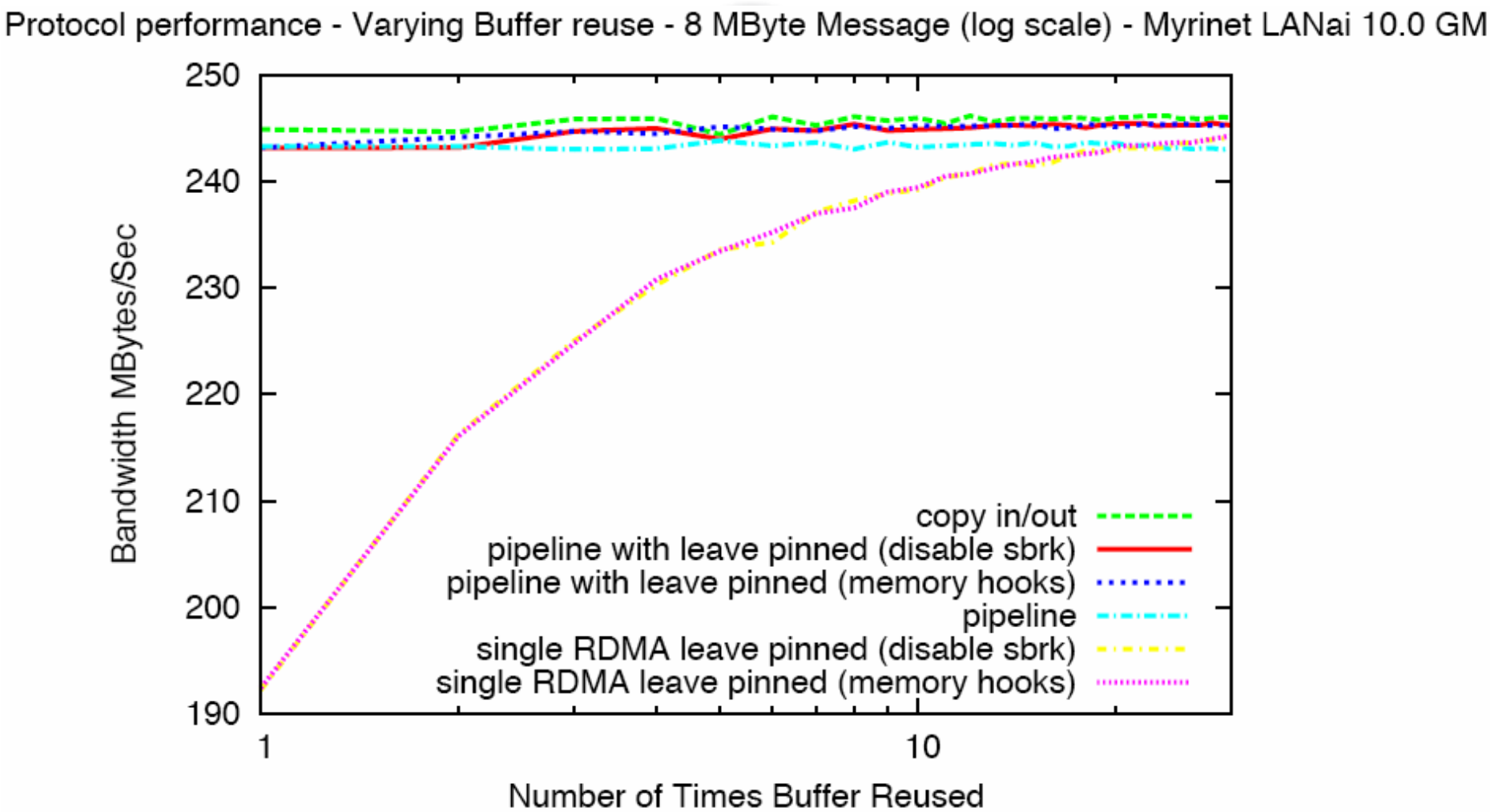
No Buffer Reuse - Mellanox OFED/OpenIB



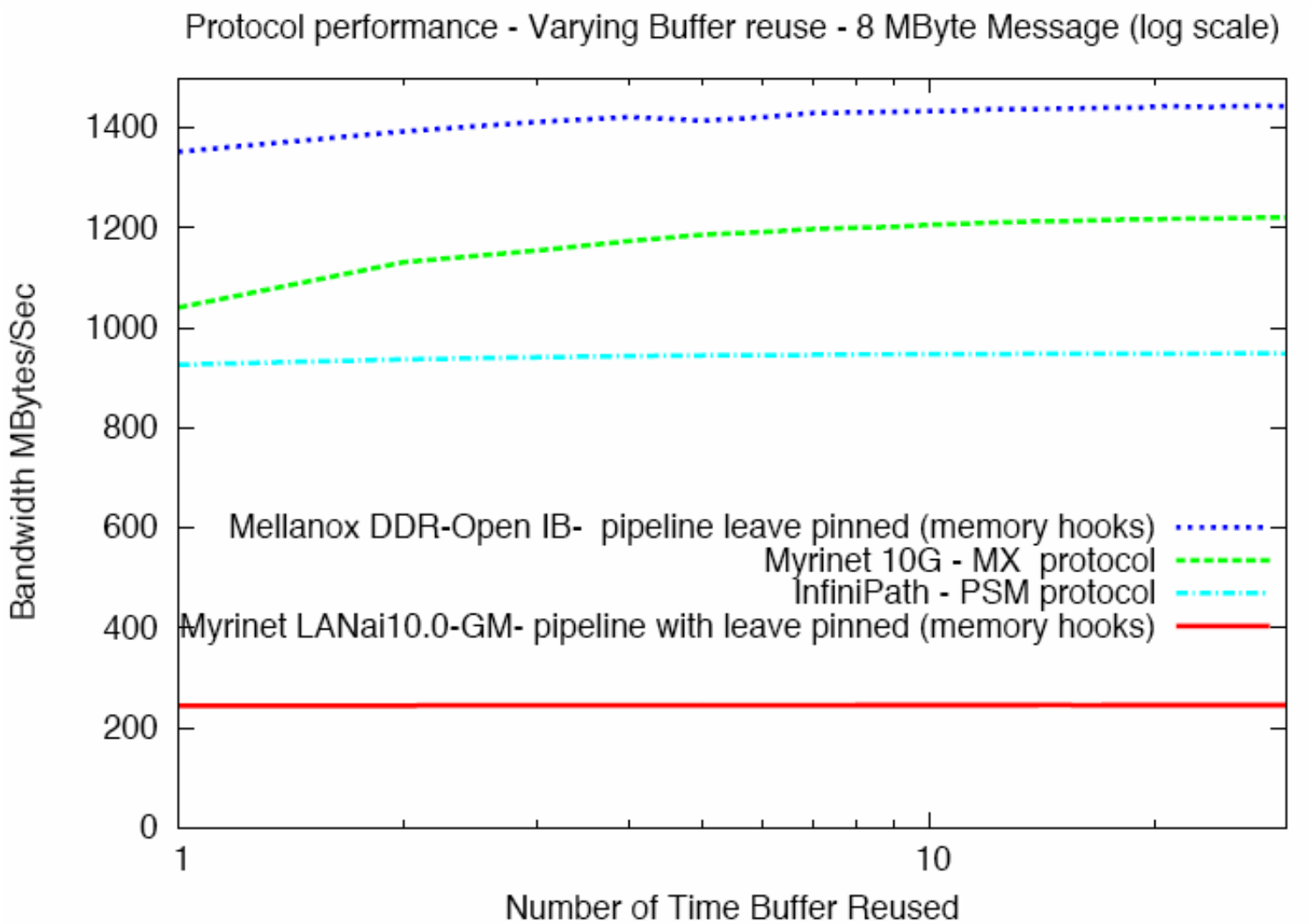
Pipeline Performance - Mellanox OFED/OpenIB



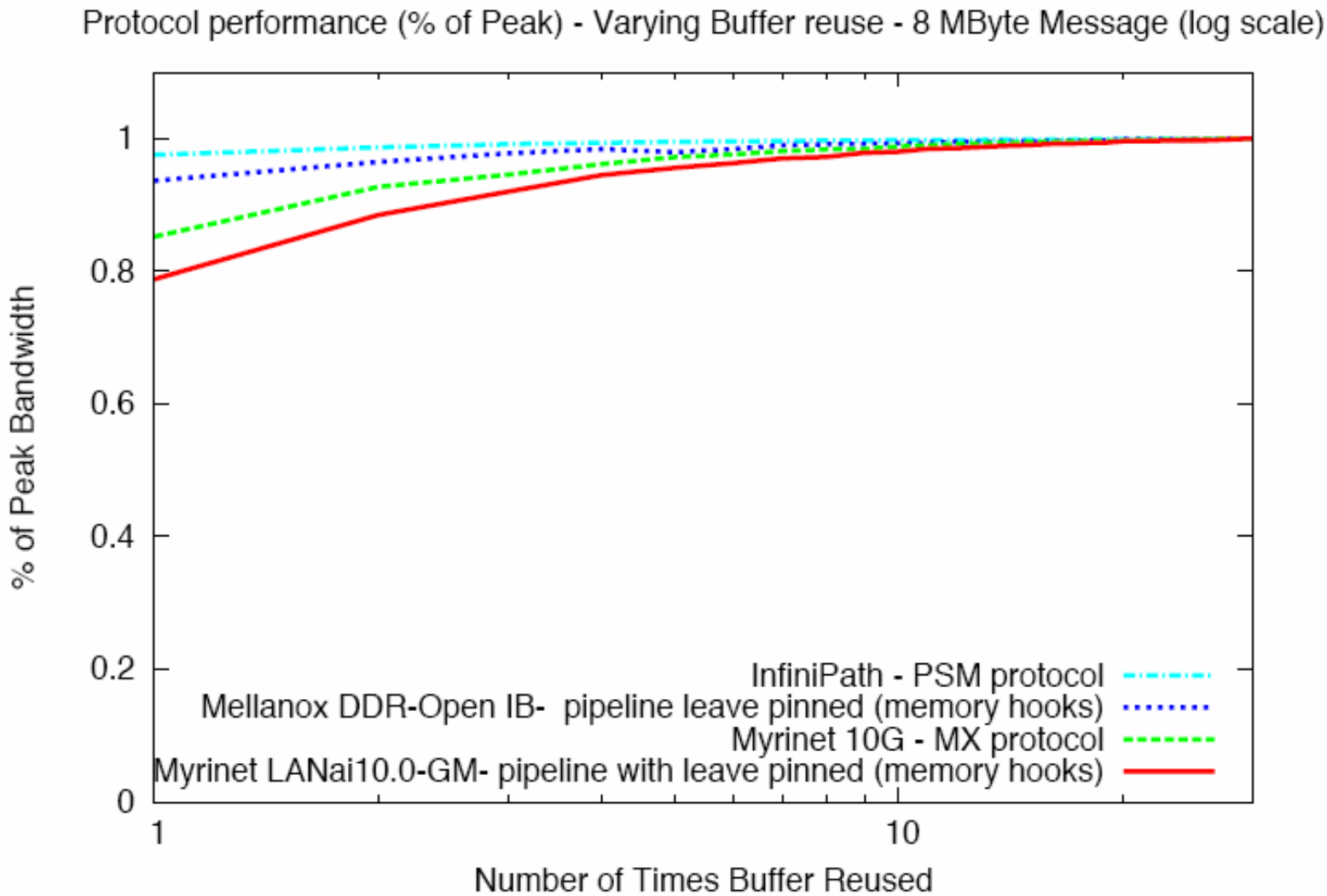
Pipeline Performance - Myrinet GM



Best of OMPI Comparison



Best of OMPI Comparison - % of Peak



Conclusions

- Care must be taken in mapping RDMA semantics to MPI-1 Send/Receive
  - Registration costs may not be effectively amortized by buffer reuse
  - Transitioning to rendezvous can have high costs if single RDMA is used
- A hybrid (pipeline) approach may address these issues
  - Good performance is not reliant on buffer reuse
  - Transition to rendezvous may avoid high cost of initializing RDMA operations.
  - Good performance may be realized without disabling sbrk() or use of memory hooks to intercept free().

Questions?