Exploiting the ExaNeSt Communication Primitives for a High Performance MPI Library

A. Psistakis, M. Asiminakis, P. Xirouchakis, M. Gianioudis, P. Peristerakis,
F. Chaix, M. Ploumidis, V. Papaefstathiou, N. Chrysos, M. Katevenis

Foundation for Research and Technology - Hellas (*FORTH*)

Goals

- Functionality contributed by FORTH within ExaNeSt
  - Communication primitives
    - Hw + sw
- Functionality employed by ExaNeSt, contributed by ExaNoDe
- Demonstrate above functionality
  - MPI library
    - Partial implementation of the MPI standard
    - Contributed by FORTH

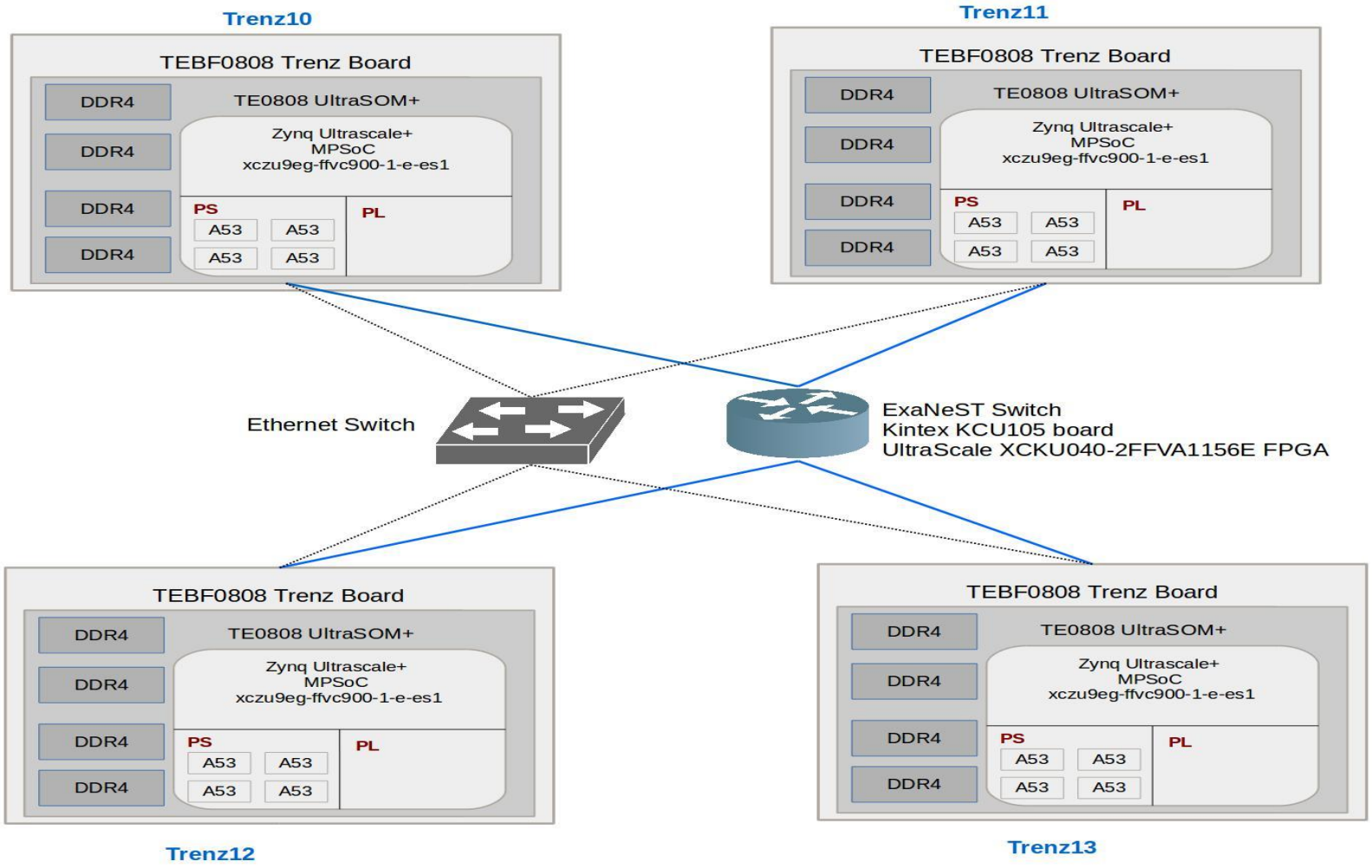
Overview

- Prototype employed
- Transfers between local and remote virtual addresses
- User-level zero-copy RDMA
  - Virtual addresses
- User-level low-latency atomic message delivery
- UnimemMPI

Overview

- Prototype employed
- Transfers between local and remote virtual addresse4s
- User-level zero-copy RDMA
  - Virtual addresses
- User-level low-latency atomic message delivery
- UnimemMPI

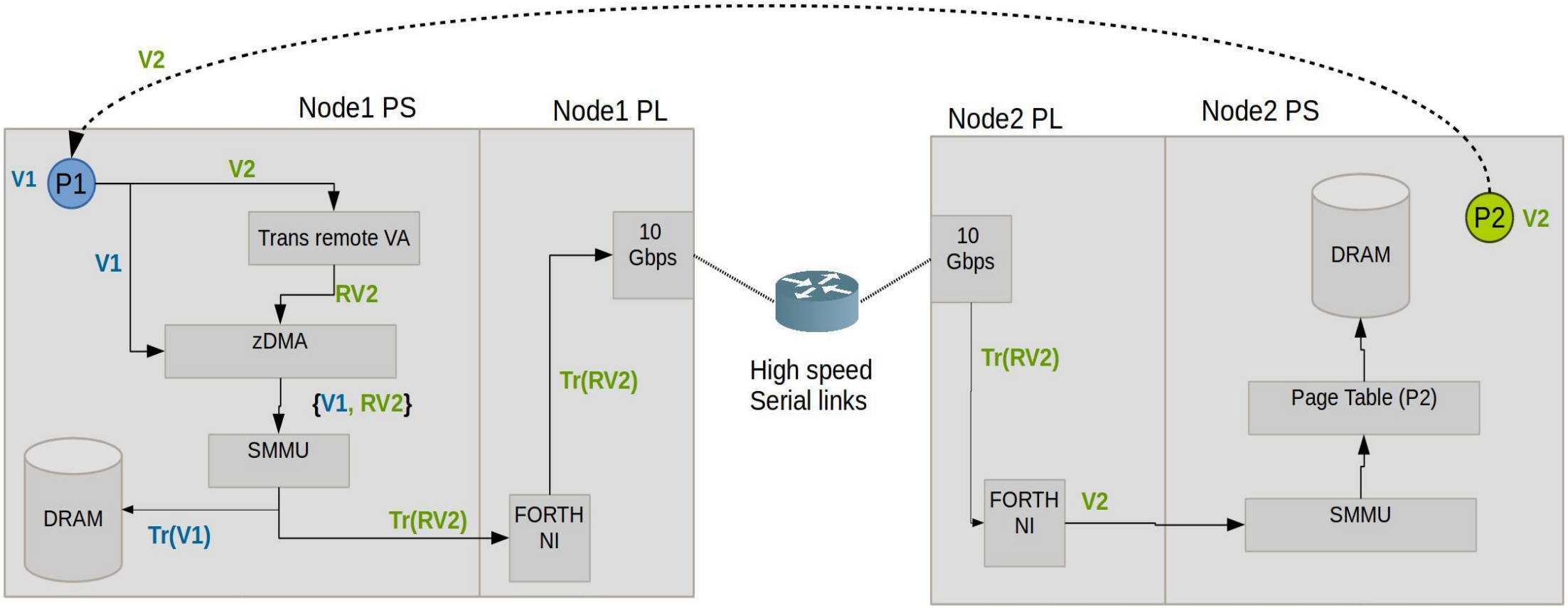
Development Prototype



Overview

- Prototype employed
- Transfers between local and remote virtual addresses
- User-level zero-copy RDMA
  - Virtual addresses
- User-level low-latency atomic message delivery
- UnimemMPI

Transfers between local and remote VAs



Using remote virtual addresses

- Access to remote virtual addresses
  - Unimem architecture
- Virtual address in Xilinx Linux Kernel = 39 bits
- Routing data add to remote VAs = 8 bits
- Restriction (due to Xilinx Kernel, not Unimem)
  - ∀ Process Virtual address ∈ $[0, 2^{31} - 1]$
- Workaround
  - Patch mmap() system call
  - Addresses in stack
    - Mackecontext, swapcontext, getcontext

Exploiting System MMU 1/2

- Configuration and programming of system MMU (a.k.a IOMMU)
  - zDMA, PL
- 16 contexts banks
- Kernel module patch
  - Context: process page table
- Local and remote virtual address translation

SMMU: protected access to remote VAs

- Processes of the same application
  - Same protection domain
    - Protection domain ID
- Sender side:
  - FORTH's NI at Programmable logic (PL)
    - Add PDID information to transaction
- Receiver side:
  - FORTH's NI: extract PDID
  - PDID -> stream ID
  - Stream ID -> SMMU context bank
  - Context bank = translation context (process page table)

Overview

- Prototype employed
- Transfers between local and remote virtual addresse4s
- User-level zero-copy RDMA
  - Virtual addresses
- User-level low-latency atomic message delivery
- UnimemMPI

User level zero copy RDMA transfers

- Employ low power domain zDMA in PS
  - Cache coherent memory accesses
  - Eight independent channels
- Kernel module by FORTH
  - Each process: exclusive access to single zDMA channel
  - Associate PDID to zDMA channel
  - Control of zDMA from user space
    - Kernel space involved only at init time
    - Not involved in actual transfers
      - Zero-copy transfers
  - Virtual addresses
    - No memory pinning

Overview

- Prototype employed
- Transfers between local and remote virtual addresse4s
- User-level zero-copy RDMA
  - Virtual addresses
- User-level low-latency atomic message delivery
- UnimemMPI

User-level low-latency atomic message delivery 1/2

- FORTH's contribution in ExaNoDe
- Two hardware blocks
  - Virtualized mailbox
  - Virtualized packetizer
- Kernel modules
- User-space library
  - Expose to user-space code
- Virtualized mailbox
  - 256bits messages, 64 interfaces
- Virtualized packetizer
  - 256 bits messages, 64 interfaces, exploit AXI Burst capability

User-level low-latency atomic message delivery 2/2

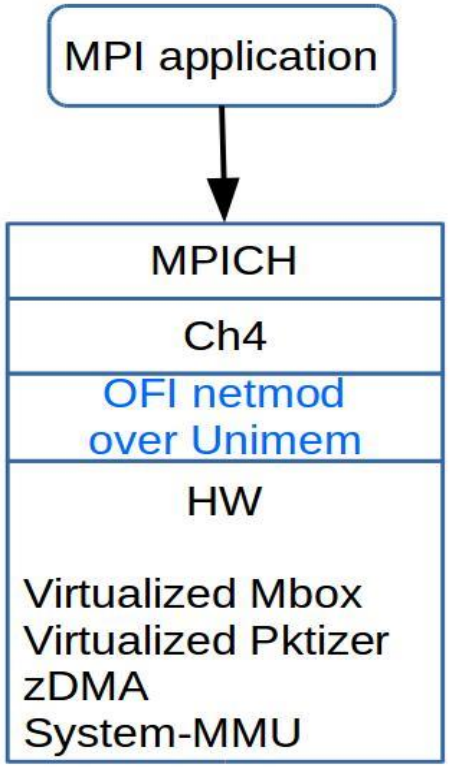
- Kernel module
  - Configure virtualized packetizer/mailbox
  - Process/thread: attach one/64 virtual packetizer/mailbox
- User-level atomic message send/recv
  - Packetizer 256 bits message
  - Destined to remote process's mailbox

Overview

- Prototype employed
- Transfers between local and remote virtual addresse4s
- User-level zero-copy RDMA
  - Virtual addresses
- User-level low-latency atomic message delivery
- UnimemMPI

MPI libraries in ExaNeSt

- MPI From BSC
  - MoU with ExaNoDe

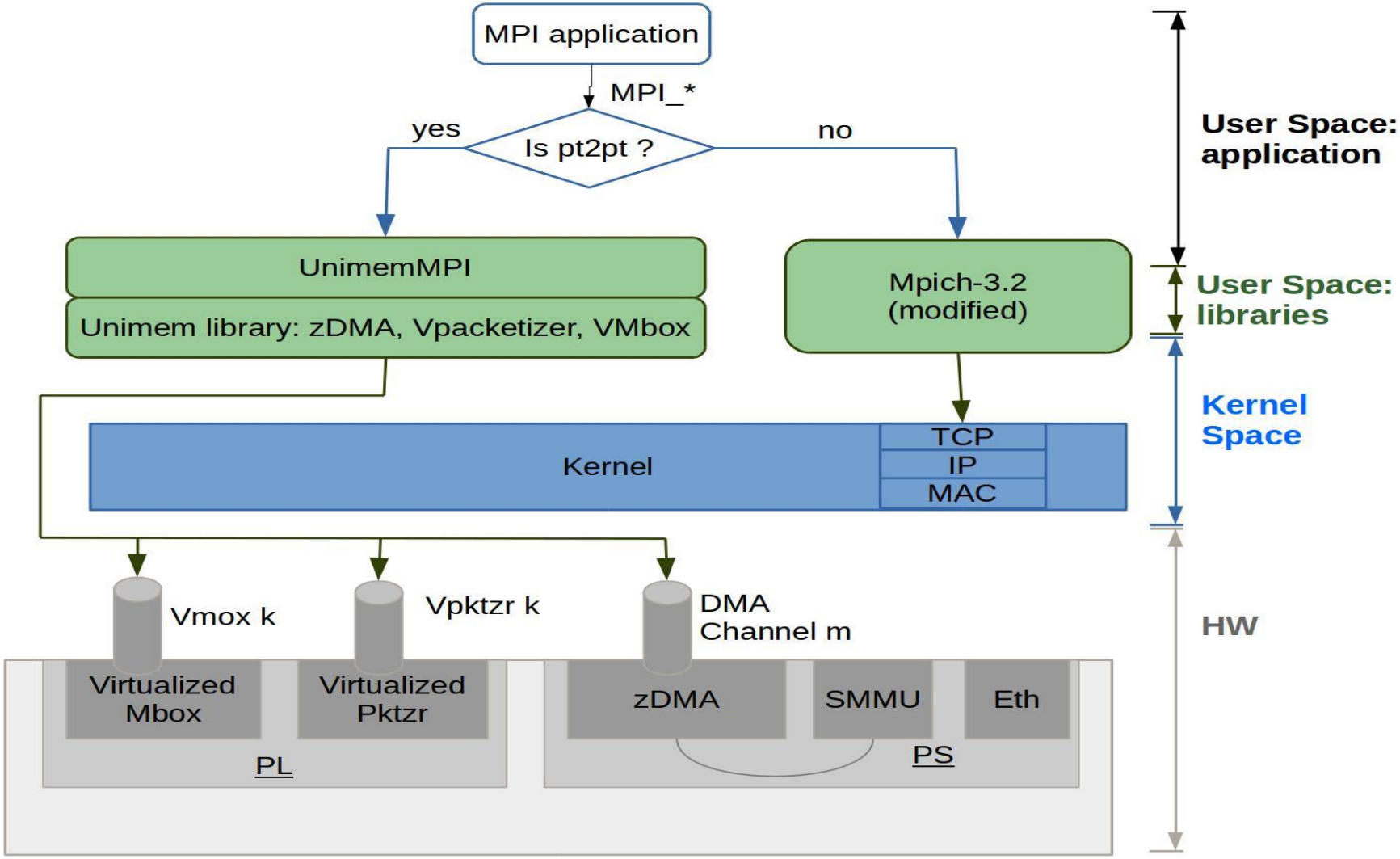


- UnimemMPI
  - Contributed by FORTH
  - Debug/optimize performance
    - Software libraries
    - HW
  - Preliminary performance results
    - Other than micro-benchmarks

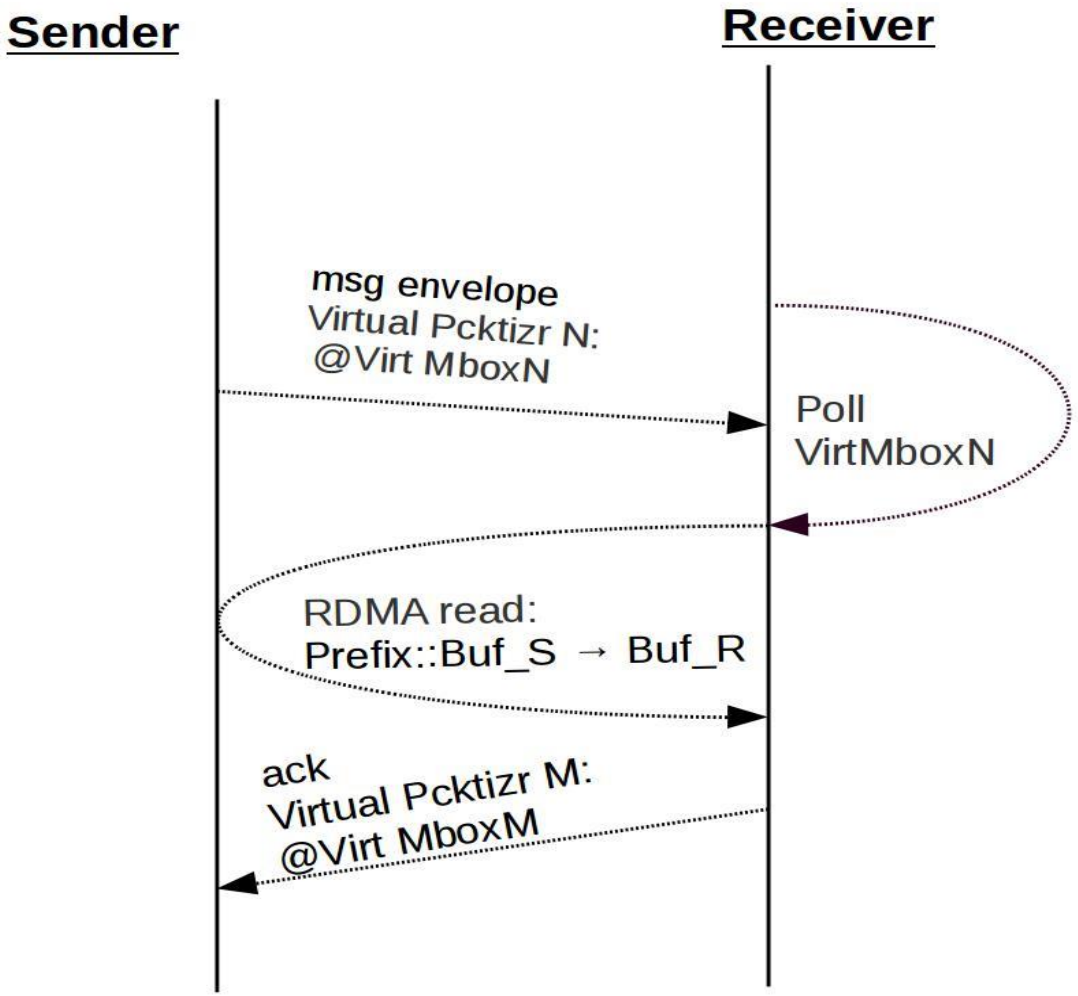
UnimemMPI overview 1/2

- Partial implementation of the MPI standard
  - Almost all point-to-point primitives
  - Collectives
    - Delegated to slightly modified MPICH library
- Point-to-point related functionality under-testing
  - Traffic through derived data types
    - Copy….
  - Persistent requests
  - MPI_Cancel, MPI_Mprobe, MPI_Improbe, MPI_Mrecv, MPI_Imrecv

UnimemMPI overview 2/2



UnimemMPI messaging protocol



UnimemMPI preliminary results 1/2

- Micro-benchmarks
  - Packetizer to remote mailbox message ~= 2 usec
  - zDMA-read (8 bytes) ~= 3 usec
  - zDMA-read (4096 bytes) ~= 18.3 usec
- MPI ping-pong test
  - UnimemMPI ping-pong = 4 mailbox messages + 2 zDMA-read ops

Ping, pong msg size	UnimemMPI	MPICH(TCP sockets)
8 bytes	12.5 usec	279.6 usec
4096	46.0 usec	1207.1 usec

UnimemMPI preliminary results 2/2

- Preliminary results
  - LAMMPS application
    - State of the art molecular dynamics code
    - Cooperation with eXact lab
    - Rhodopsin problem
    - Three OMP threads per run

Num of nodes	Timesteps	Unimem Wall Timesteps/s	MPICH (TCP sockets) Timesteps/s	Gain
2	600	1.599	1.542	3.6%
3	900	2.103	1.982	6.1%
4	1200	2.575	2.409	6.9%
6	1800	3.205	2.889	10.9%

Ongoing and future work

- Ongoing
  - Move from AXI-based prototype to ExaNet
  - Support missing pt2pt primitives
  - Allow remote page faults
  - Employ new mailbox
    - Larger messages, higher performance
- Future work
  - MPI messaging protocol = rdma-write based