Toward development of OFI provider to support MPICH over Unimem

Kyunghun Kim[1], Polydoros Petrakis[2], Manolis Ploumidis[2], Yanfei Guo[3], Kenneth Raffenetti[3], Paul Carpenter[1], Nikolaos Dimou[2], Pavan Balaji[3], Antonio J. Peña[1]

Barcelona Supercomputing Center (BSC)[1], Foundation for Research and Technology[2], Argonne National Laboratory[3]

Introduction

Unimem

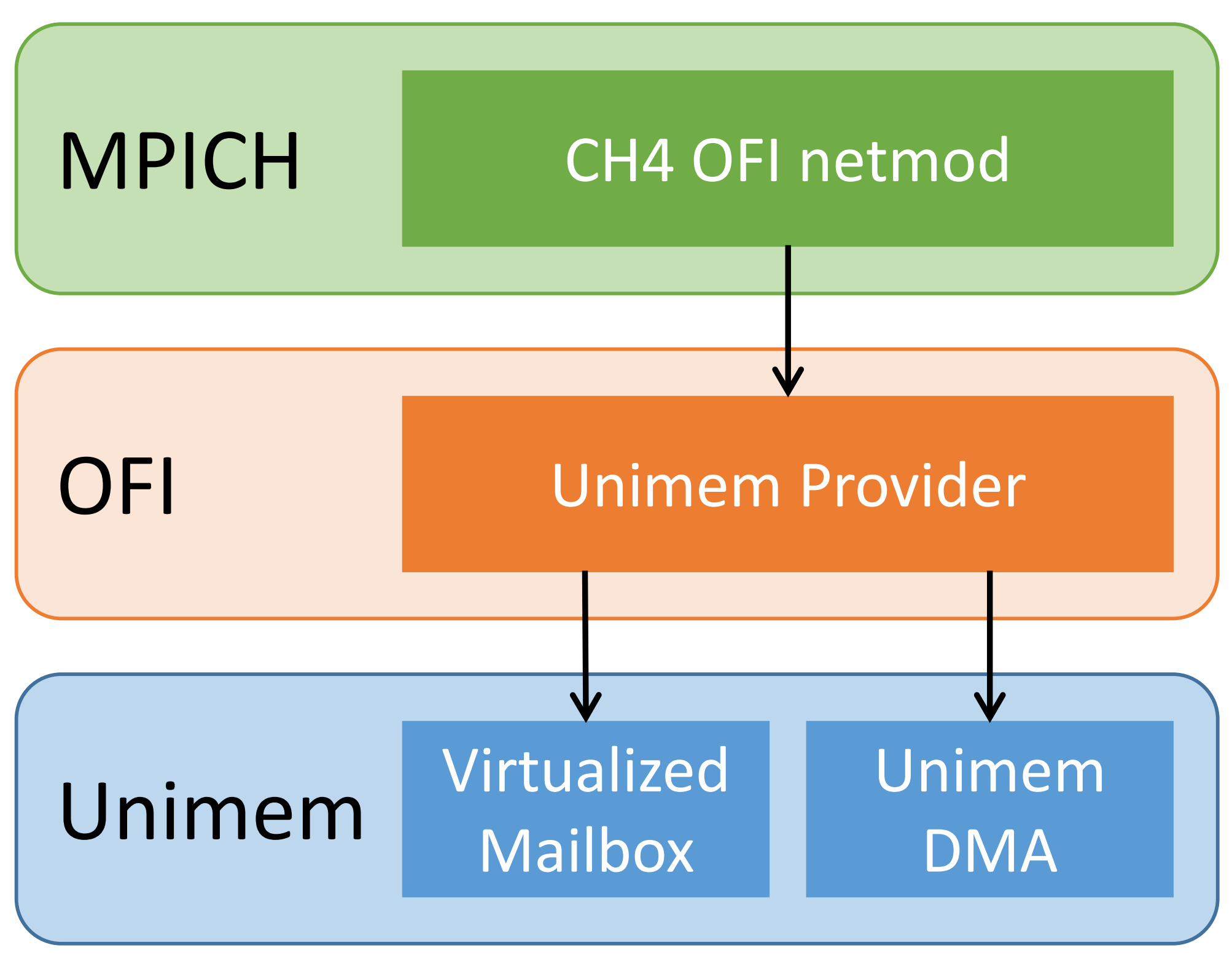
Unimem is ARM-based exascale architecture for High Performance Computing. Unimem provides unified memory address across multiple nodes and follows PGAS (Partitioned Global Address Space) model. In this project, we will develop MPI for Unimem.

OFI libfabric

OFI is a framework to export fabric communication services and libfabric provides a unified user-space API to communication runtimes such as MPI or SHMEM. Each hardware vendor implements its OFI provider to support the functionality of specific hardware.

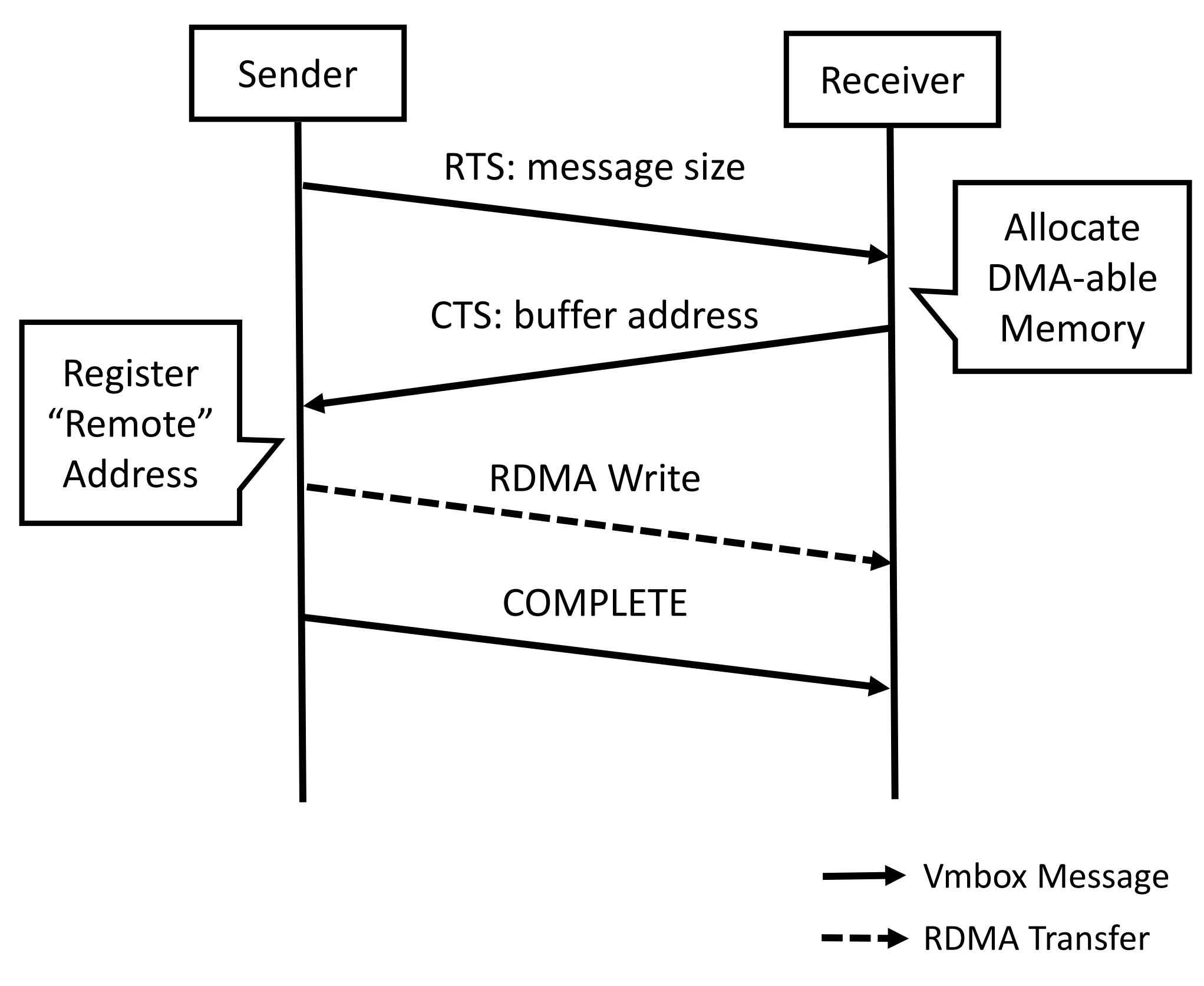
MPICH+OFI+Unimem

One of the most leveraged state–of–the–art approaches to develop MPI runtimes for new hardware is developing an OFI provider. We developed an OFI provider for Unimem and applied it under MPICH CH4. We implemented the OFI provider features mandatory to operate MPICH CH4 first, then optimized our OFI provider for better performance.



Unimem APIs

Virtualized Mailbox (vmbox) is a simple hardware queue for control messages with 192-bit fixed size. **Unimem DMA** is an RDMA (Remote Direct Memory Access) engine based on FPGA hardware.



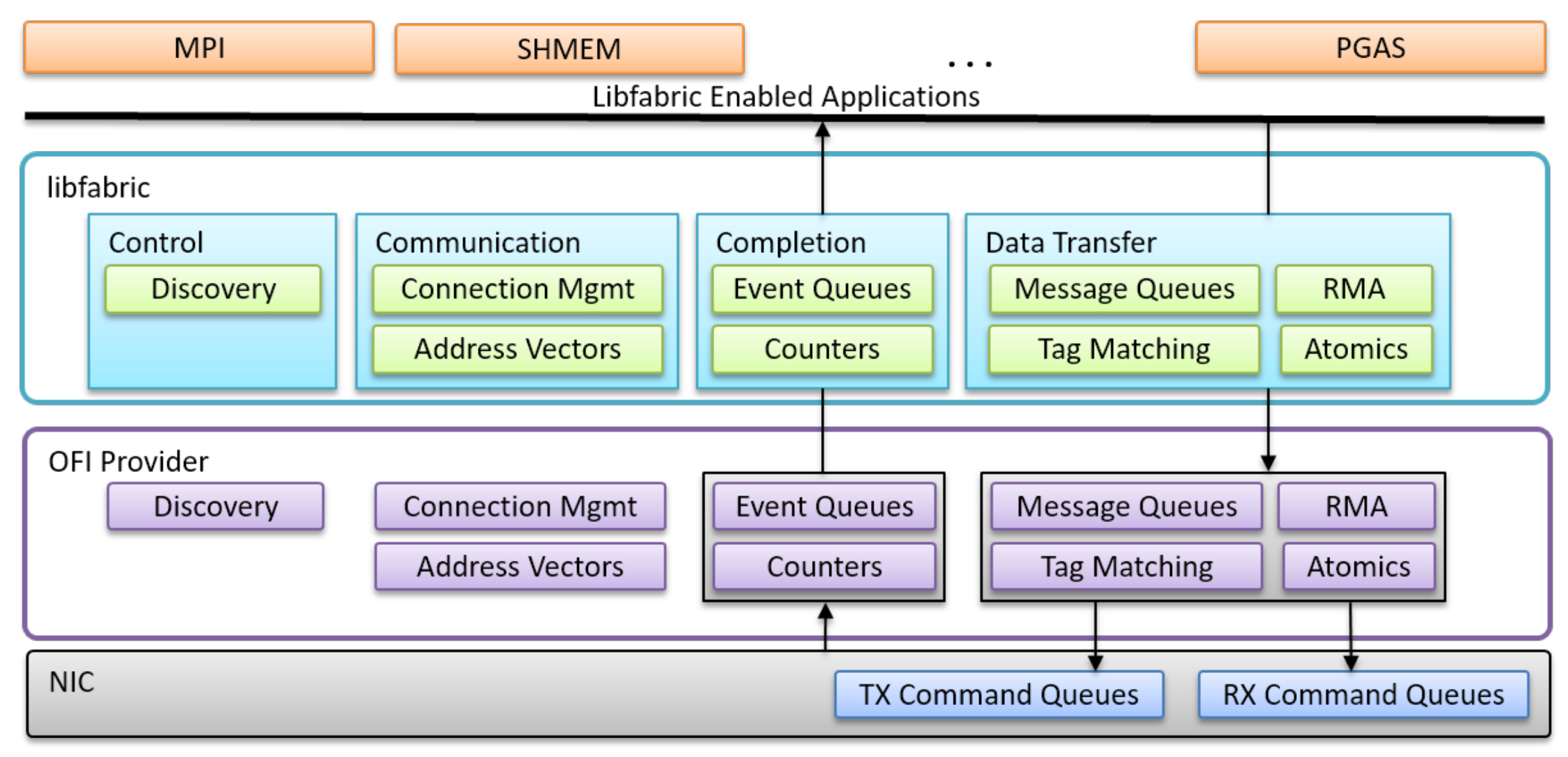
Allocate DMA-able Memory: Unimem architecture supports RDMA only for a specific memory address range.

Register Remote Virtual Address: The remote address should be registered to the local RDMA engine to prepare the route.

libfabric for MPICH

To operate MPICH CH4 with libfabric, the fundamental OFI provider features are:

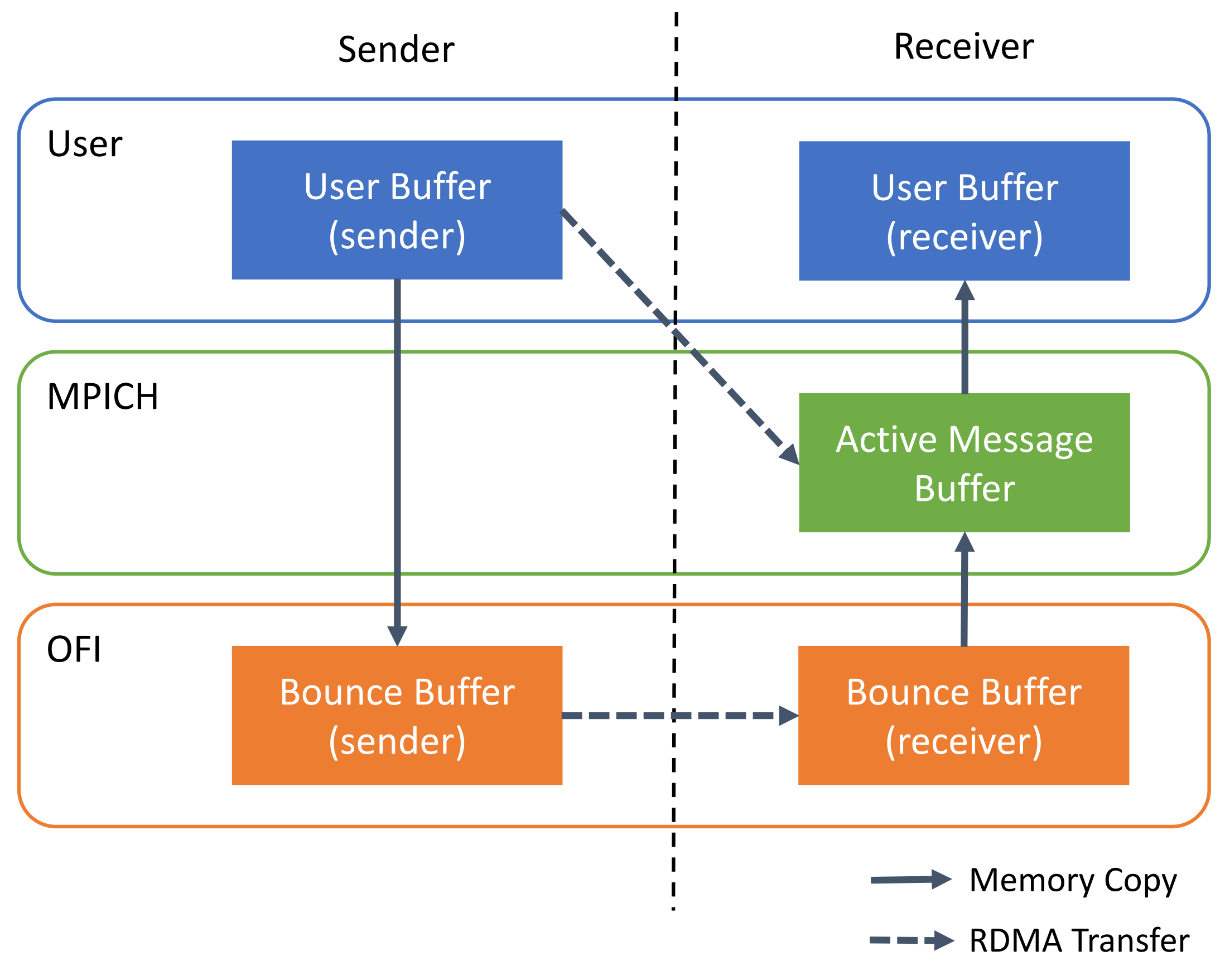
- Basic libfabric features: fabric, domain, endpoint, and completion queue.
- `FI_RDM`: Connectionless endpoint and connection mngmt. for remote node.
- `FI_MSG`, `FI_MULTI_RECV`: Basic message passing and receiving multiple messages by posting a single large receive buffer.
- `FI_RMA` was also required for long message transfer but we implemented a temporary patch in MPICH.



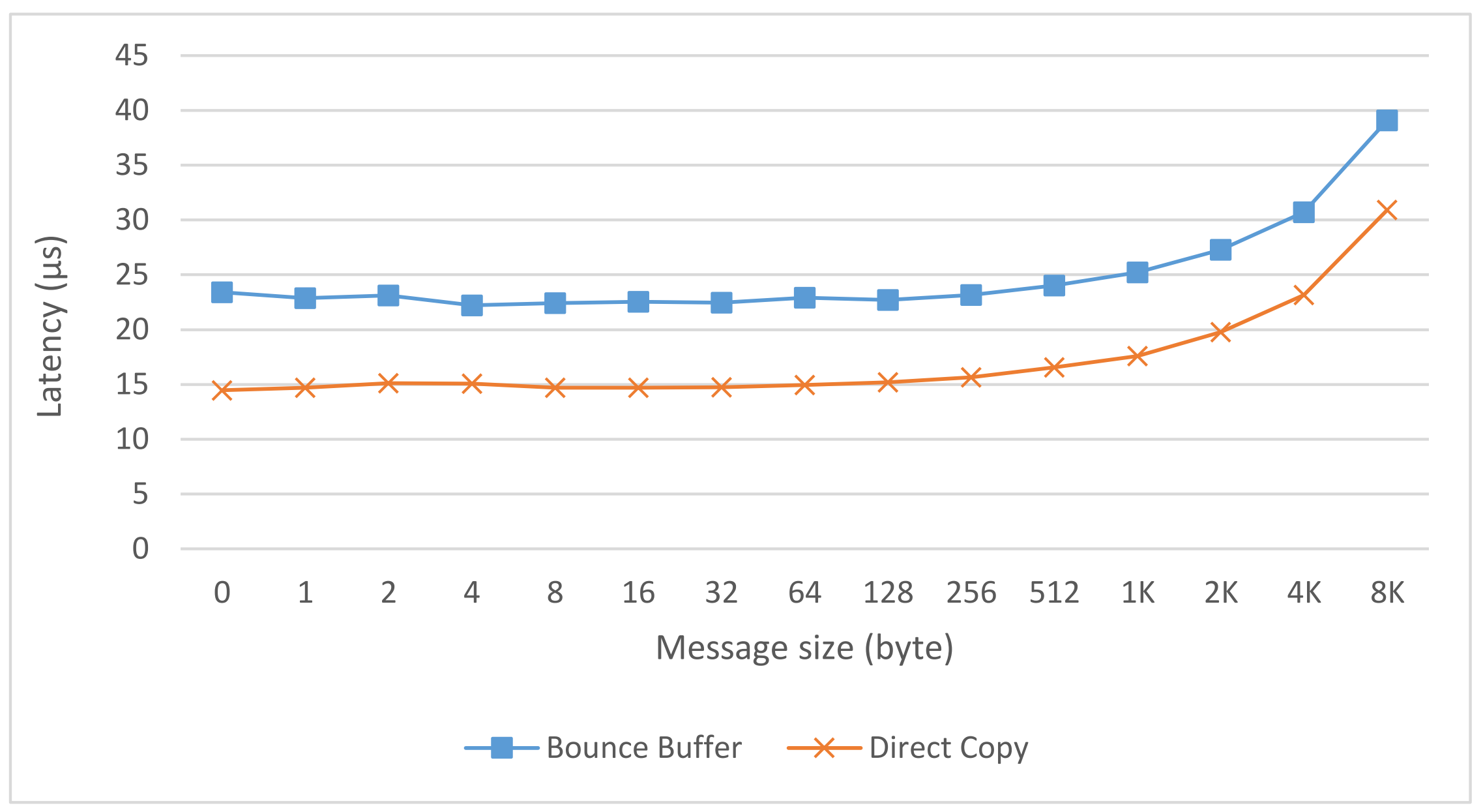
Design and Implementation

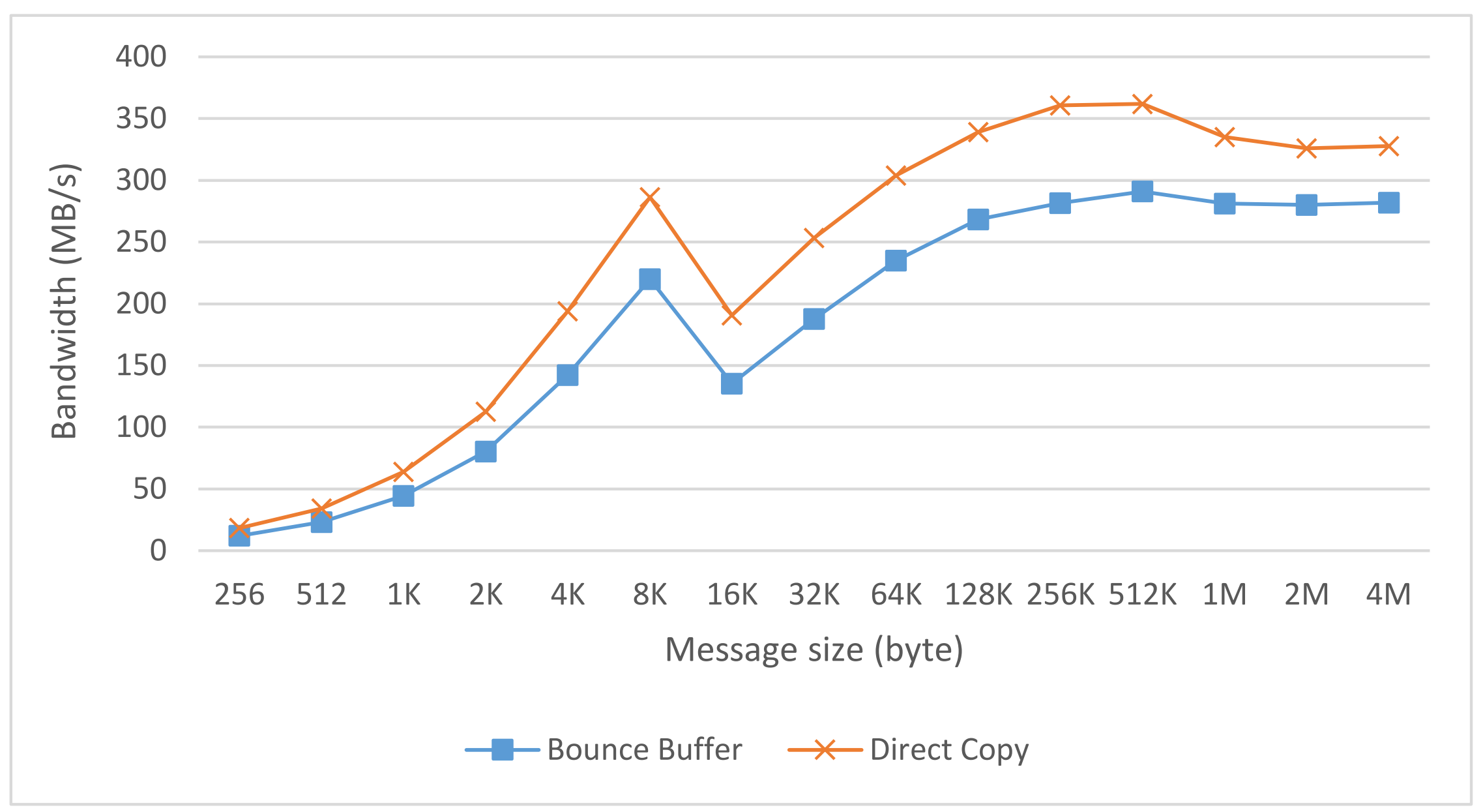
Bounce Buffers: Unimem DMA only supports RMA operations from a specific memory area, as provided by the `alloc_dmable_buf()` function from the Unimem DMA API. User buffers residing outside this memory region musts be copied to intermediate bounce buffers in our OFI provider.

Direct Copy: If the user buffer is *DMA-able*, we can directly transfer from the user buffer to an Active Message buffer in MPICH. This route reduces 2 copies while transferring the message, so it is necessary to provide proper buffer allocation function to users.



Results





Latency performance (`osu_latency`) and bandwidth performance (`osu_bw` benchmark)

We measured the performance of our implementation with OSU micro-benchmarks. Our implementation could run the whole OSU micro-benchmarks with no problem.

- `osu_latency` : 23.41 µs (with bounce buffer) →14.47 µs (direct copy to user buffer)
- `osu_bw` : 360.59 MB/s peak bandwidth

(1) Without copying to intermediate bounce buffers, there was significant latency performance improvement. (2) There was bandwidth degradation especially for 16KB size messages because MPICH changes send message protocol to long message transfer (lmt) at this size, which requires more control messages for transfers.

Conclusion

We developed an OFI provider for the Unimem architecture and applied our OFI provider to the MPICH CH4 runtime.

For future work, we need to implement additional optimizations to improve latency. For example, small messages can be transferred by directly writing to preallocated buffers in the receiver side (*eager* transfers).

References

Jiuxing Liu, Jiesheng Wu, and Dhabaleswar K Panda. High performance RDMA-based MPI implementation over InfiniBand. *International Journal of Parallel Programming*, 32(3):167–198, 2004

Sung-Eun Choi, Howard Pritchard, James Shimek, James Swaro, Zachary Tiffany, and Ben Turrubiates. An implementation of OFI libfabric in support of multi-threaded PGAS solutions. In *9th International Conference on Partitioned Global Address Space Programming Models (PGAS)*, pages 59–69. IEEE, 2015