OPENFABRICS ALLIANCE

13th ANNUAL WORKSHOP 2017

URDMA: RDMA VERBS OVER DPDK

Patrick MacArthur, Ph.D. Candidate

University of New Hampshire

March 28, 2017

ACKNOWLEDGEMENTS

- urdma was initially developed during an internship with the IBM Zurich Research Laboratory. The author would like to thank Dr. Bernard Metzler for the opportunity as well as Jonas Pfefferle, Patrick Stuedi, and Animesh Trivedi for their advice and critique on urdma.
- The author would like to thank the University of New Hampshire InterOperability Laboratory for the use of their RDMA cluster for the development, maintenance, and testing of urdma and UNH EXS.
- This material is based upon work supported by the National Science Foundation under Grant No. OCI-1127228 and under the National Science Foundation Graduate Research Fellowship Program under award number DGE-0913620.

BACKGROUND

EXISTING SOFTWARE RDMA DRIVERS

- **softiwarp and rxe**
  - Implement iWARP over TCP and RoCEv2, respectively
  - Data transfer in kernel space
  - Run unmodified verbs applications
  - Designed with performance in mind
- **libfabric sockets provider**
  - Implements private protocol
  - Userspace implementation using TCP/IP sockets
  - Cannot run verbs applications
  - High performance explicitly not a goal

SOFTWARE VERBS DRIVERS: KERNEL VS. USER SPACE

Why not implement a verbs driver using sockets API from userspace?

- **Userspace verbs API design choices**
  - Verbs will not load a userspace driver without a corresponding uverbs device exposed by the kernel
  - Connection management deferred to kernel by librdmacm
  - CQ events delivered from kernel
- **Using userspace sockets API requires both userspace and kernel involvement**
- **Using kernel sockets API**
  - Incoming RDMA READ and RDMA WRITE can be handled entirely in kernel without waking user thread
  - Can use tricks like sendpage() to send TCP segments with zero-copy
- **Path of least resistance has been implementation using sockets API in kernel**

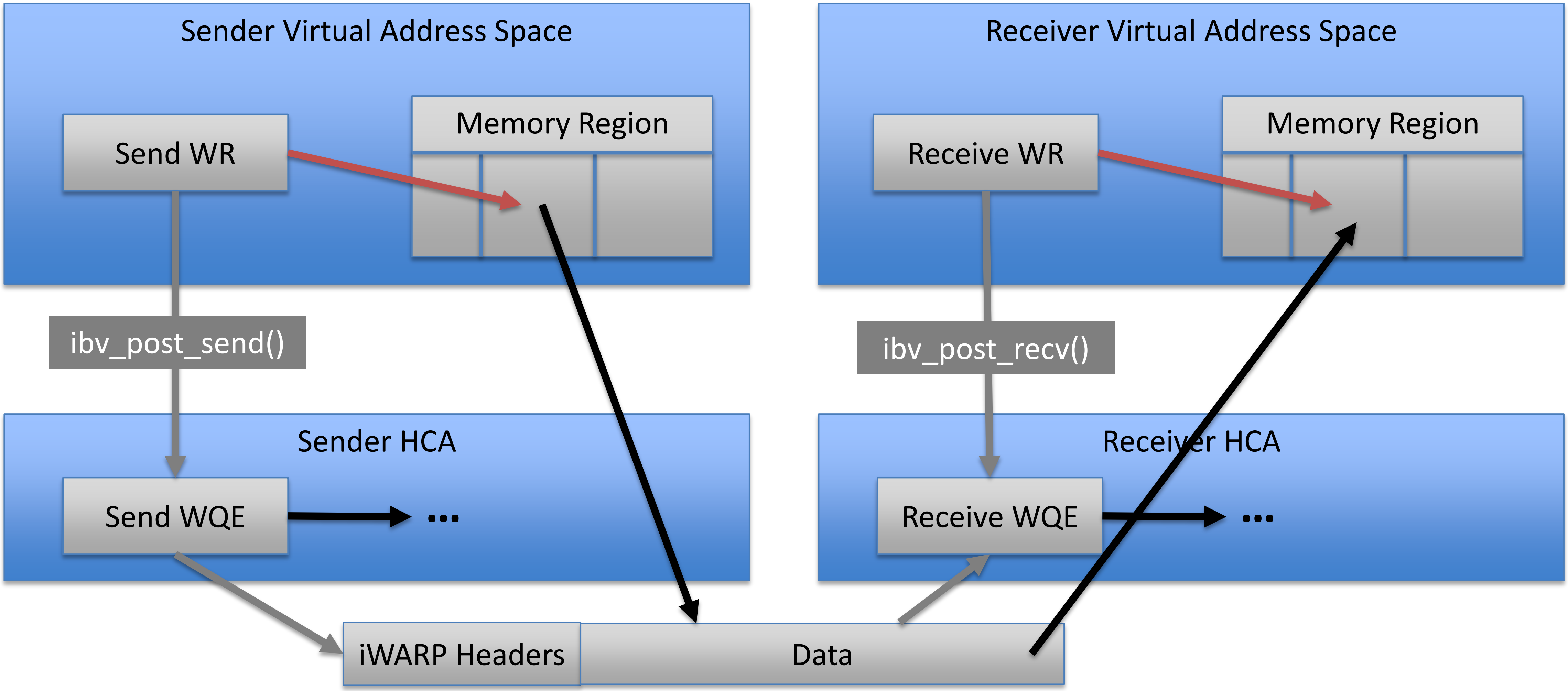
URDMA: USERSPACE RDMA

- **Goals**
  - Prototype software RDMA driver with data transfer entirely in userspace
  - Run unmodified verbs applications
  - High performance
- **Why a userspace implementation?**
  - Ease of development, makes it easy to use as a development vehicle for new RDMA features
  - Avoid context switches between kernel and userspace (especially for small SENDs)
- **Implementation uses DPDK (Data Plane Development Kit)**

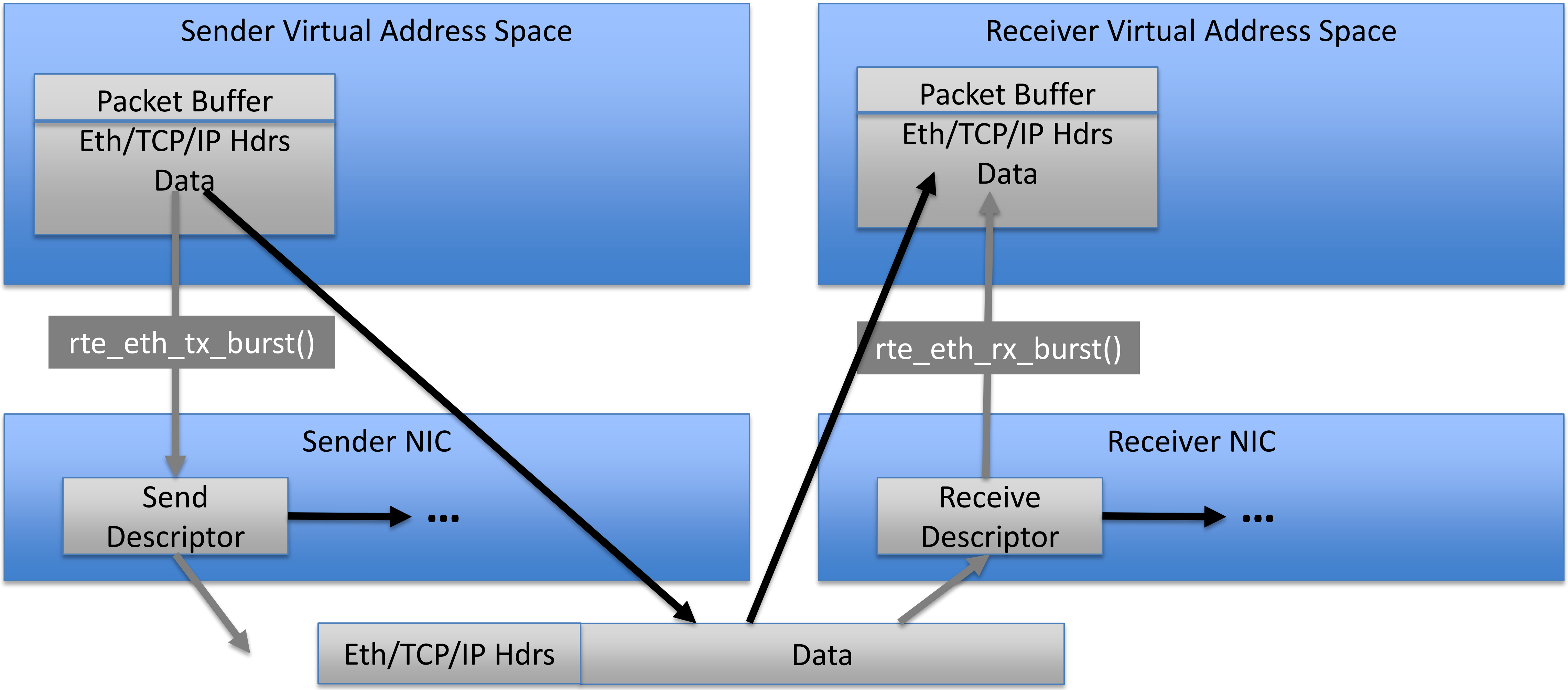
BACKGROUND: DPDK (DATA PLANE DEVELOPMENT KIT)

- **DPDK leverages Linux UIO/VFIO to map Ethernet NICs into userspace**
- **Features:**
  - Bulk packet transmit/receive to/from hardware NIC queues
  - NUMA-aware memory buffer pool allocation using hugepages
  - High performance multi-core data structures
  - Hardware packet filtering
  - TCP/UDP offloads, including checksum calculation
- **Does not provide:**
  - RDMA functionality
  - Network-layer or transport-layer protocol logic
- **Using DPDK for userspace RDMA verbs eliminates kernel from data transfer path**

RDMA SEND/RECV MESSAGE TRANSFER



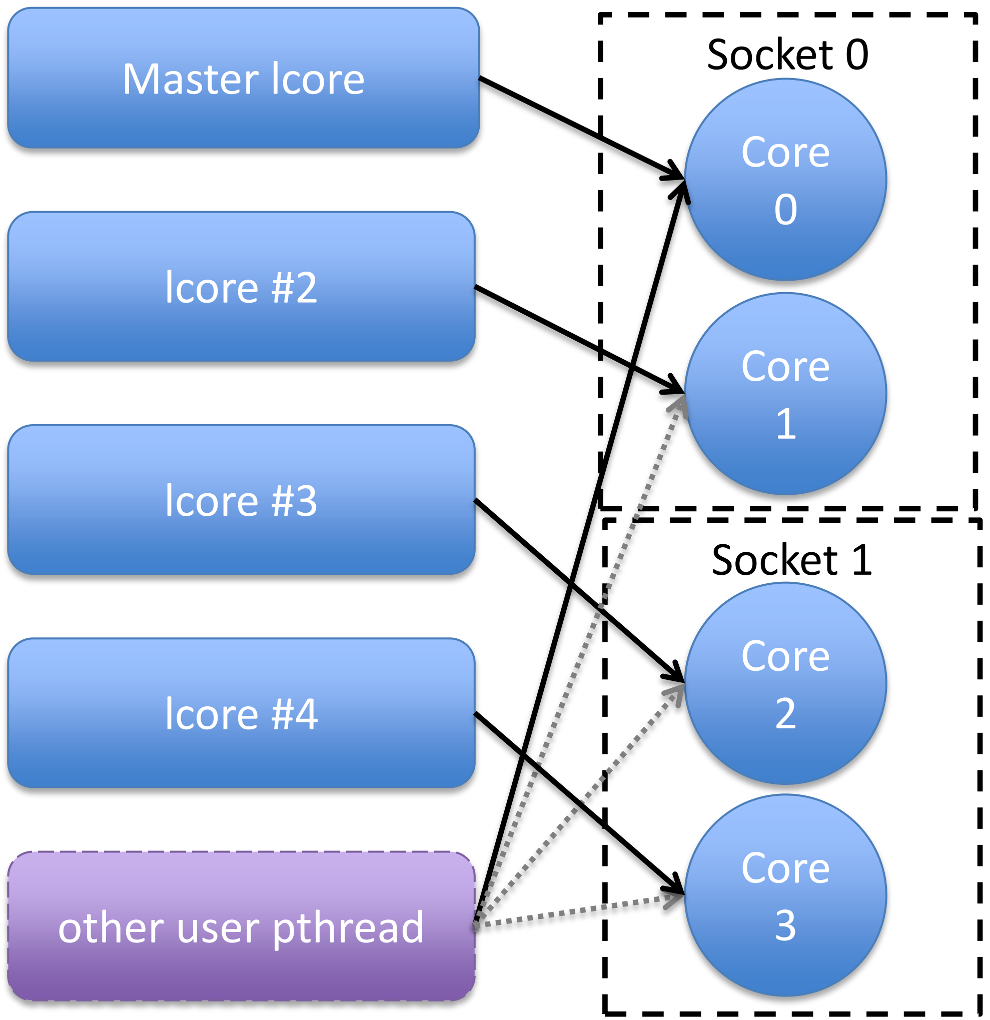
DPDK PACKET TRANSFER



BACKGROUND: DPDK THREAD MODEL

DPDK process consists of threads called "logical cores" or "lcores"

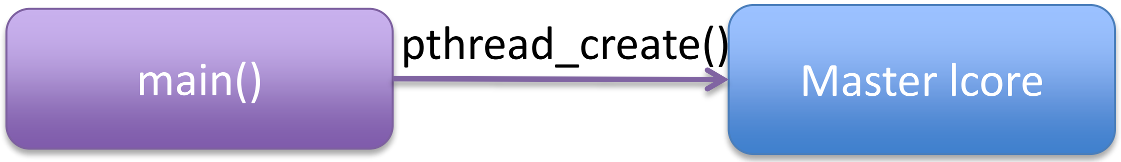
- DPDK creates 1 "lcore" thread per CPU core by default
- Thread which initializes DPDK is "master" lcore
- CPU affinity of each thread, including master, is set to run on a specific CPU core
- API allows launching tasks on other logical cores
- DPDK API expected to be called from lcores, in particular ring queues and memory pools rely on this
- We tell DPDK *not* to create lcores other than the master lcore



BACKGROUND: DPDK THREADS AND LIBRARIES

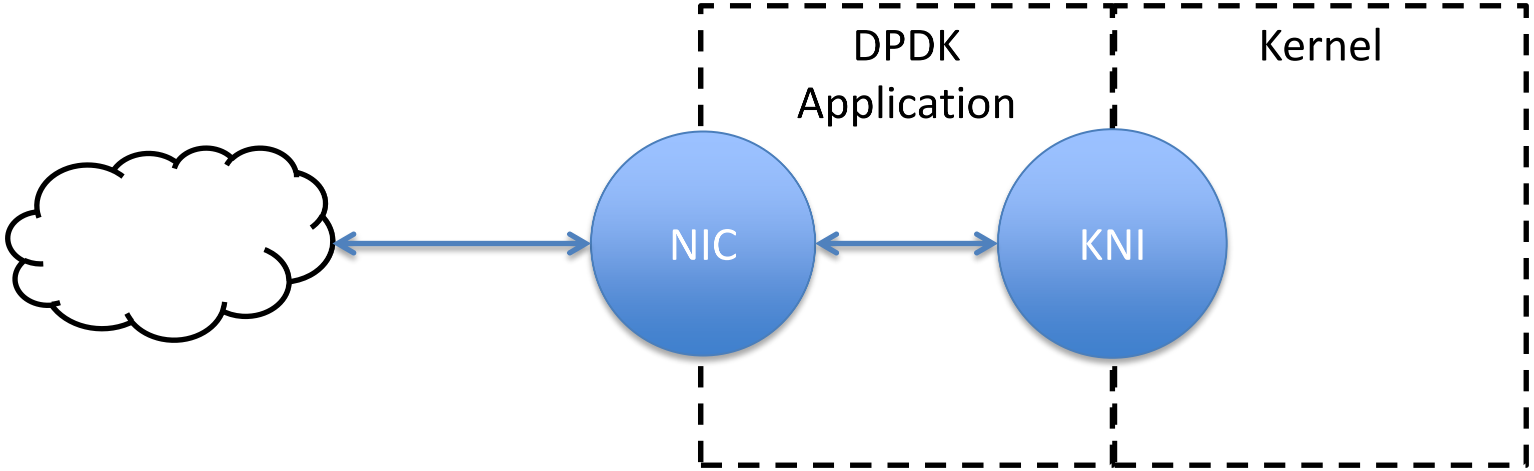
DPDK is more of an application framework than a library

- **DPDK initialization function**
  - Takes command-line arguments
  - Consumes all available hugepages by default
  - Changes CPU affinity of calling thread
- **To use DPDK from library, we create a thread and call DPDK initialization from there**
  - Pass parameter to not create further lcores
  - Separate DPDK thread from user threads
  - We do not affect CPU affinity of user threads



BACKGROUND: DPDK KNI

- **KNI (Kernel Network Interface)**
  - Creates a virtual network interface in the kernel
  - Loosely associated with a DPDK Ethernet hardware NIC
  - Can exchange packets between kernel and userspace
  - Useful for small interactions between kernel service and DPDK application



URDMA: DESIGN AND IMPLEMENTATION

URDMA: DESIGN

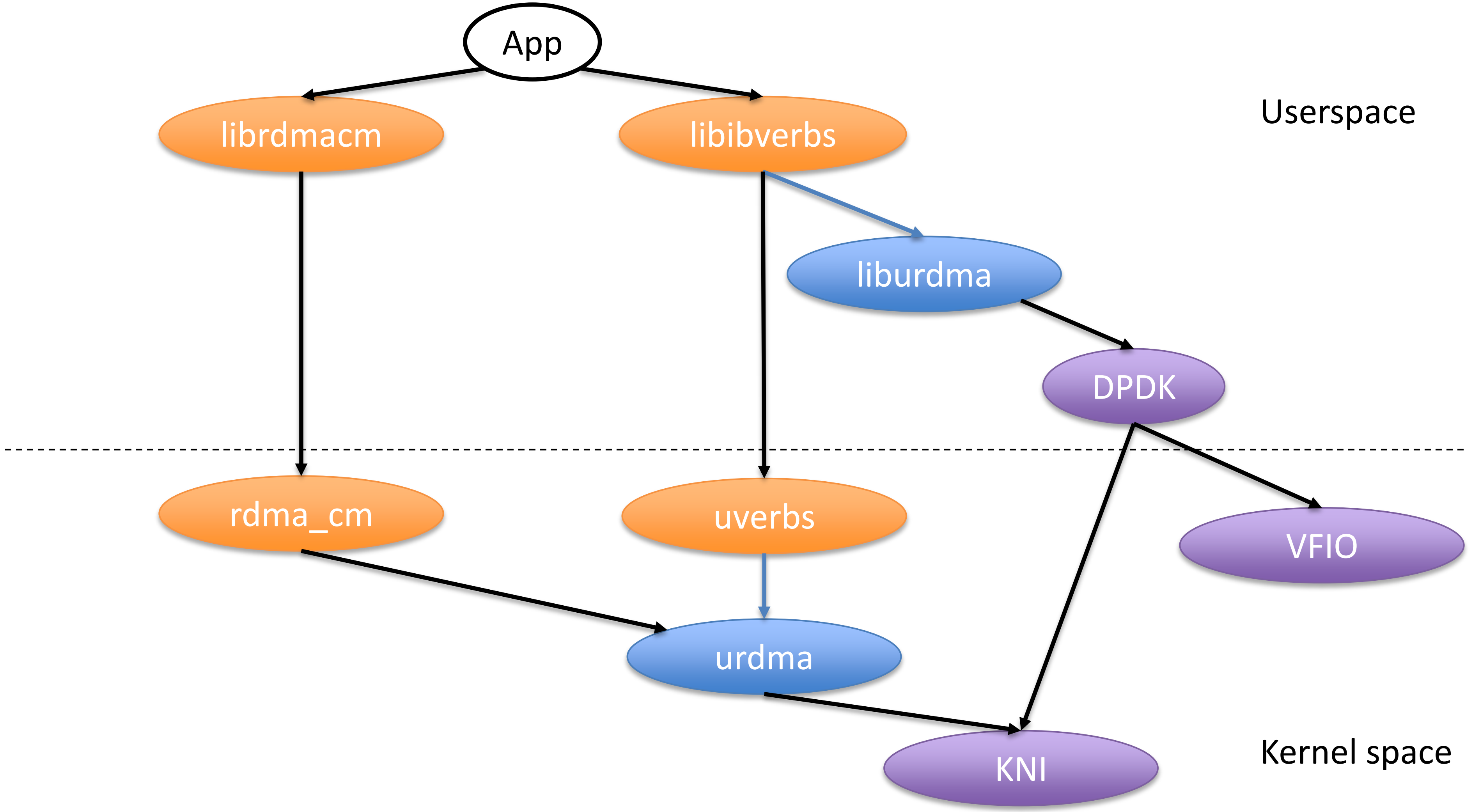
- **Implements iWARP DDP and RDMAP protocols**
- **Runs over UDP transport protocol**
  - TRP (Trivial Reliability Protocol) provides a thin shim for reliability
  - Simplifies implementation considerably
- **Small kernel component**
  - Required for libibverbs initialization, RDMA CM, and CQ events
  - Performs connection establishment before ceding control of UDP “connection” to liburdma
  - Uses KNI to send/receive packets to/from userspace
- **Packet processing done in background thread**
  - Ensure quick response to RDMA packets and KNI events
- **Hardware receive filter used to assign queue pairs to NIC receive queues**

URDMA: MULTI-PROCESS SUPPORT

- **DPDK maps Ethernet NIC hardware into userspace → owned by that process**
  - Can delegate to secondary processes that explicitly cooperate
  - DPDK considers primary + secondary processes as one combined application
  - DPDK threads in combined application cannot share the same lcore identifier
- **In urdma, primary process is a user daemon urdmad**
  - Initializes DPDK
  - Registers secondary processes with separate core mask
  - Assigns Ethernet NIC hardware RX/TX queues to urdma processes
  - Sets up Ethernet NIC hardware filtering rules
- **liburdma verbs provider**
  - Sets up process as secondary DPDK process
  - DPDK “master” lcore acts as background progress thread
- **Each liburdma process has direct access to its Ethernet NIC hardware queues**

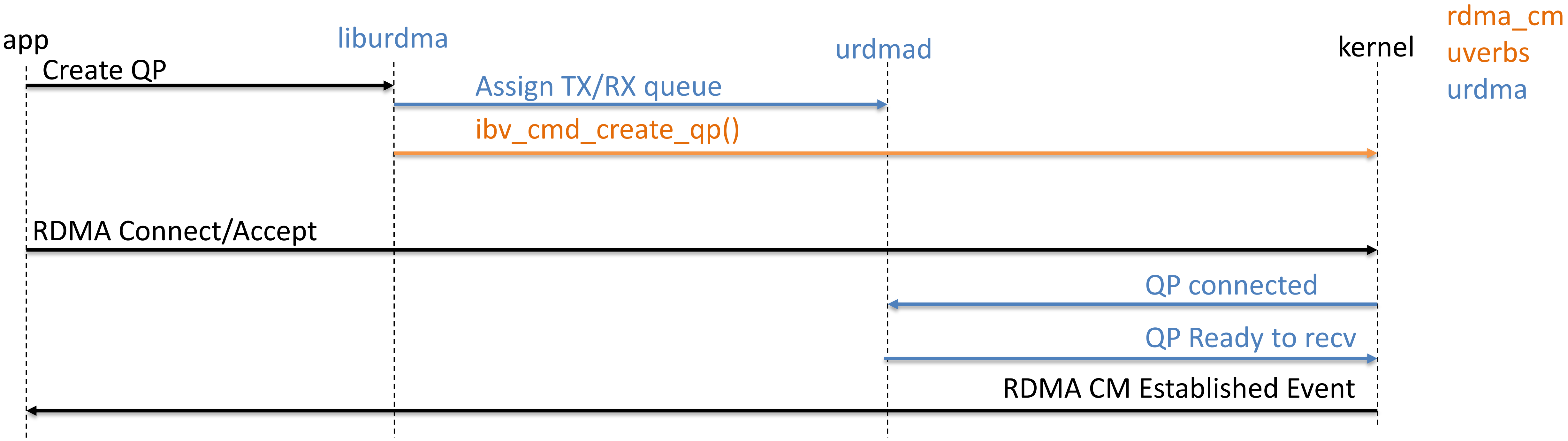
URDMA: COMPONENTS

Legend:
urdma
Verbs-related
DPDK-related

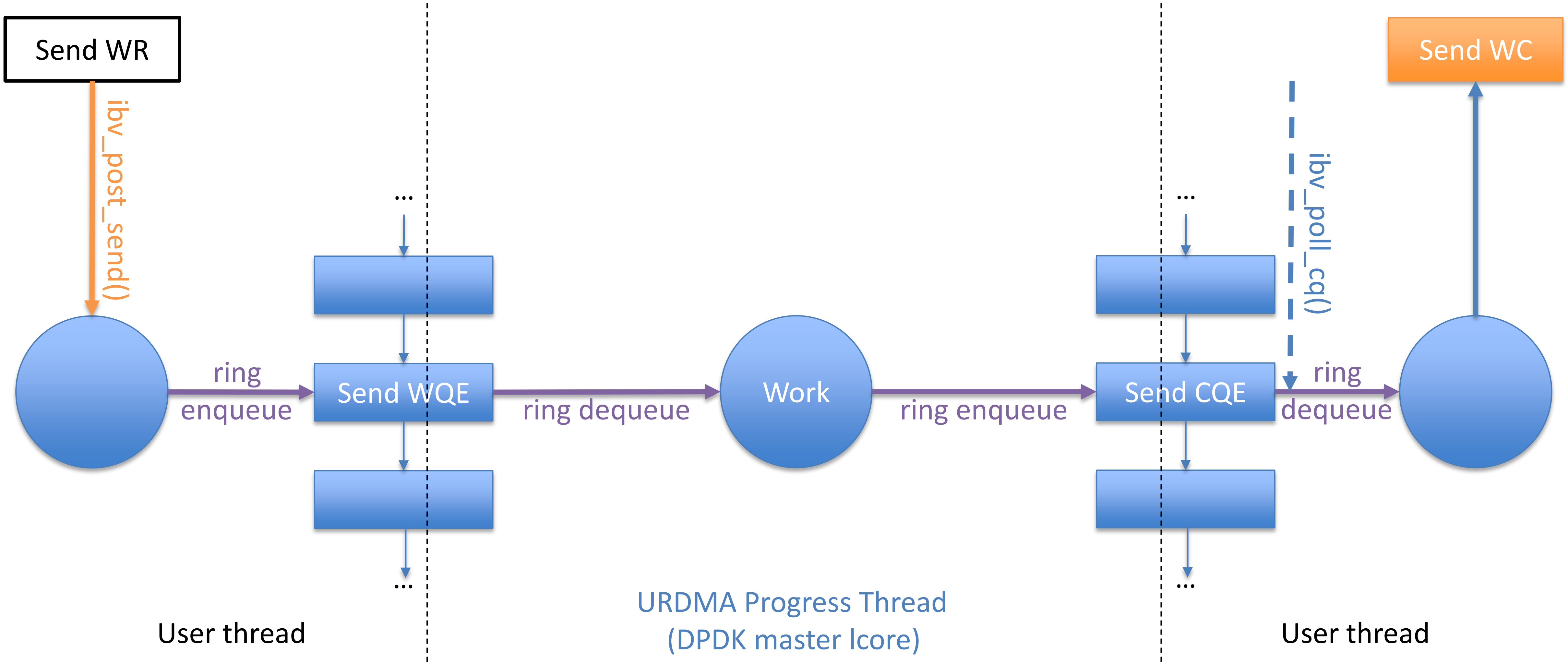


URDMA CONNECTION ESTABLISHMENT

- **Connection establishment done in kernel space**
- **In userspace:**
  - Each queue pair must be assigned a Ethernet NIC hardware send and receive queue
  - Hardware receive filtering rules must be assigned before first packet arrives
  - Private character device used to communicate connection establishment



URDMA DATA TRANSFER

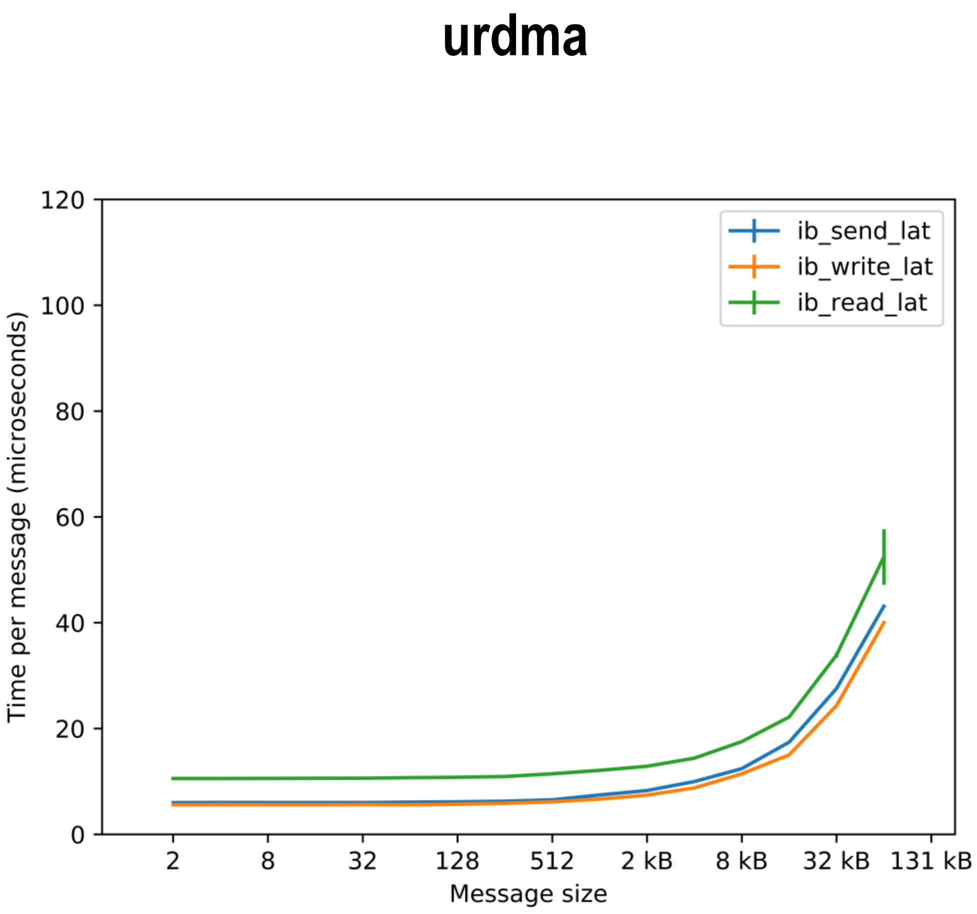


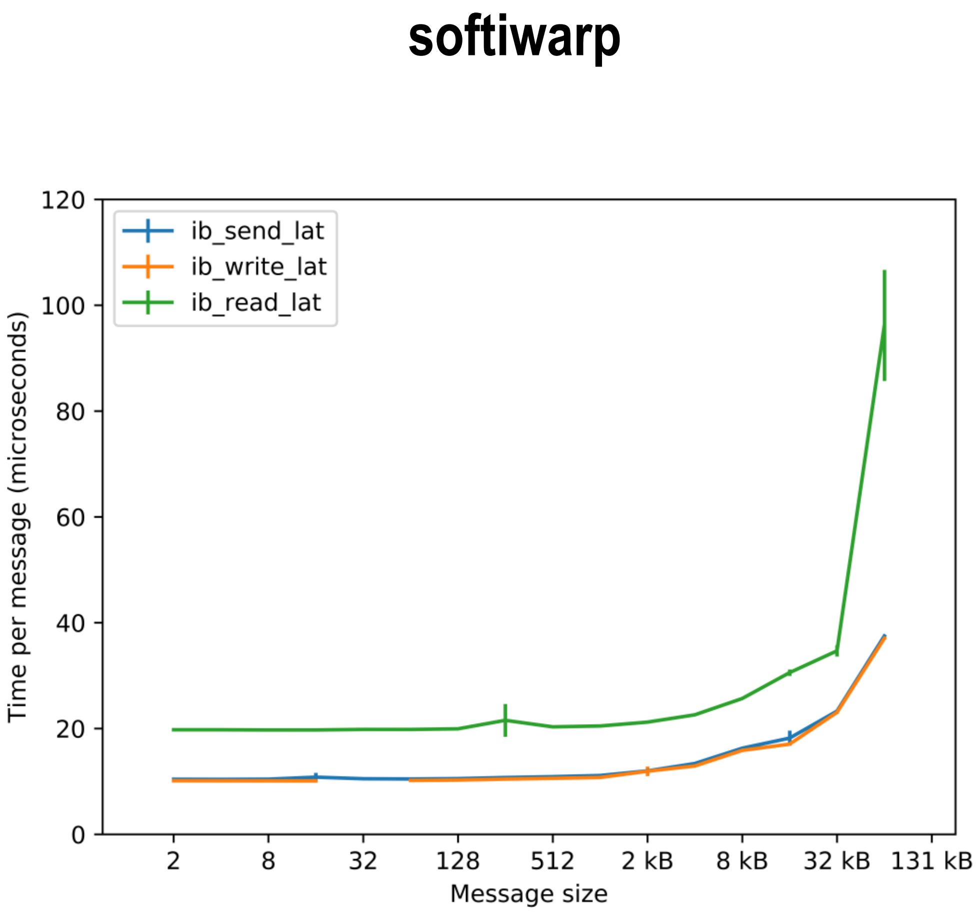
PERFORMANCE

PERFORMANCE: OVERVIEW

- **Two identical systems:**
  - Supermicro SYS-6028R-T
  - 2 Intel Xeon ES-2630 v4 CPU @ 2.20GHz
  - 64 GB DDR4 RAM
  - PCIe generation 3
  - Ubuntu 16.10 with inbox 4.8 kernel
  - Intel XL710 40GbE NIC
  - Verbs and RDMA CM as supplied with Ubuntu 16.10
- **Applications used**
  - perftest version 3.0+0.18.gb464d59-1
  - UNH EXS (Extended Sockets) 1.4.1 (https://www.iol.unh.edu/expertise/unh-exs)

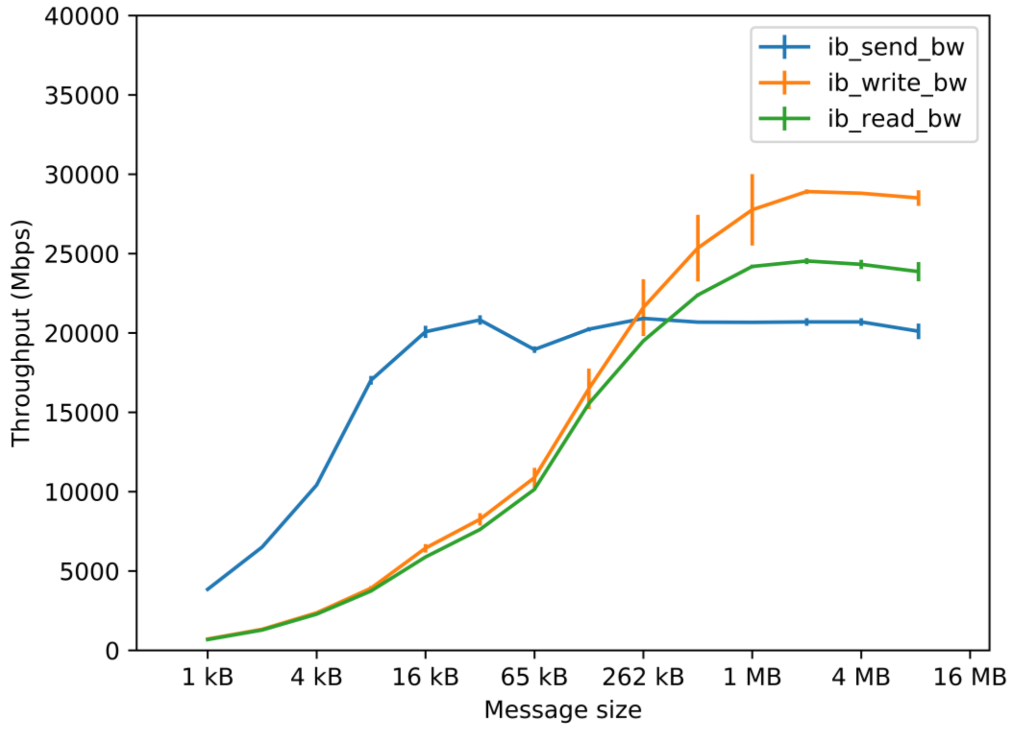
RAW VERBS: LATENCY



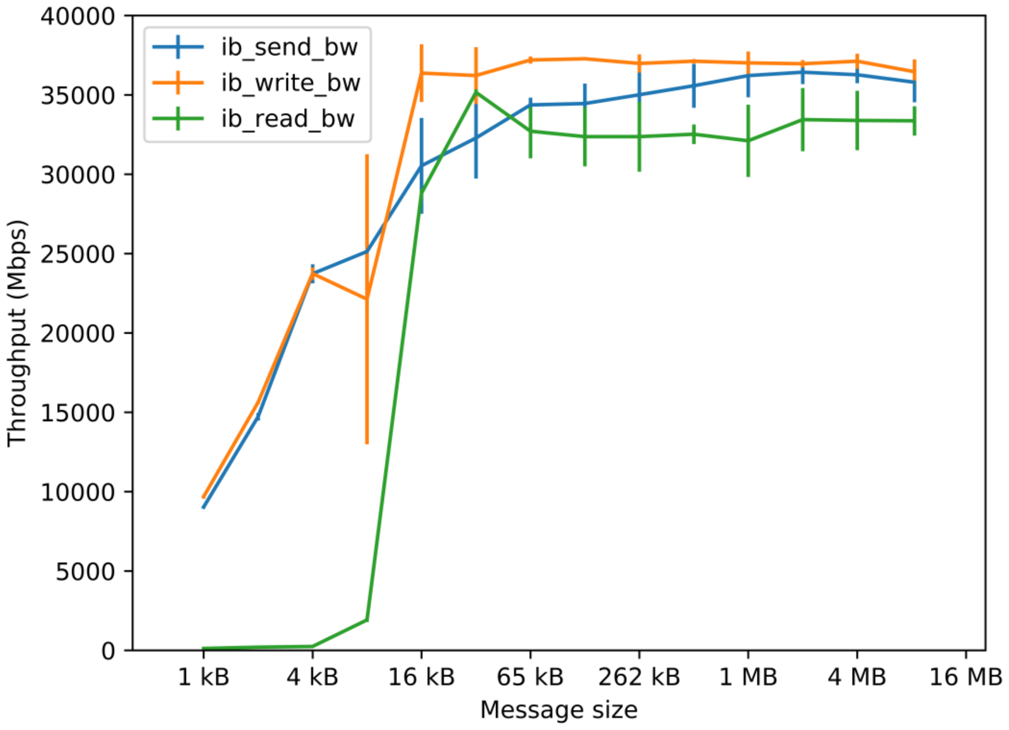


RAW VERBS: THROUGHPUT

urdma

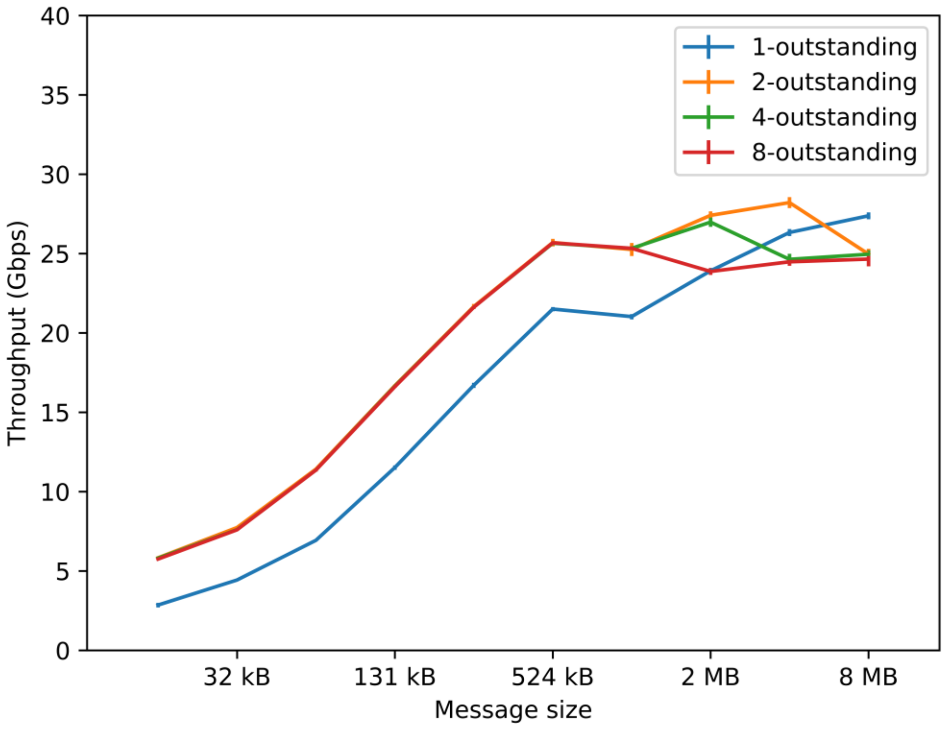


softiwarp

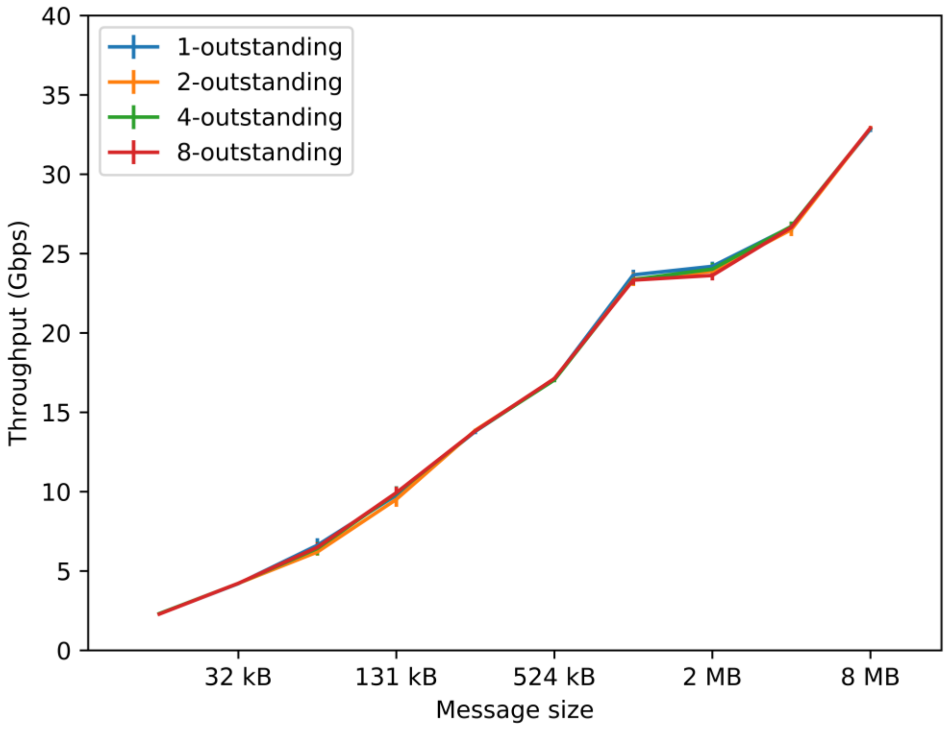


UNH EXS: THROUGHPUT

urdma



softiwarp



CONCLUSION

URDMA SUMMARY

- **Existing software RDMA implementations done in kernel space**
- **DPDK allows us to implement RDMA verbs data transfer in userspace**
  - Eliminates all kernel involvement in data transfer path
  - Small kernel module for connection management
- **Runs unmodified verbs applications**
- **Designed with performance in mind**
  - Good small message latency
  - Needs tuning for throughput
- **Future work**
  - Investigate zero-copy transmit support
  - libfabric provider implementation
  - Reliable datagram support

URDMA DOWNLOAD AND STATUS

- urdma development done on GitHub
  - https://github.com/zrlio/urdma
- No formal release as of yet
- Not integrated into rdma-core
- Tested on Ubuntu 16.10 and DPDK 16.07

OPENFABRICS ALLIANCE

13th ANNUAL WORKSHOP 2017

THANK YOU

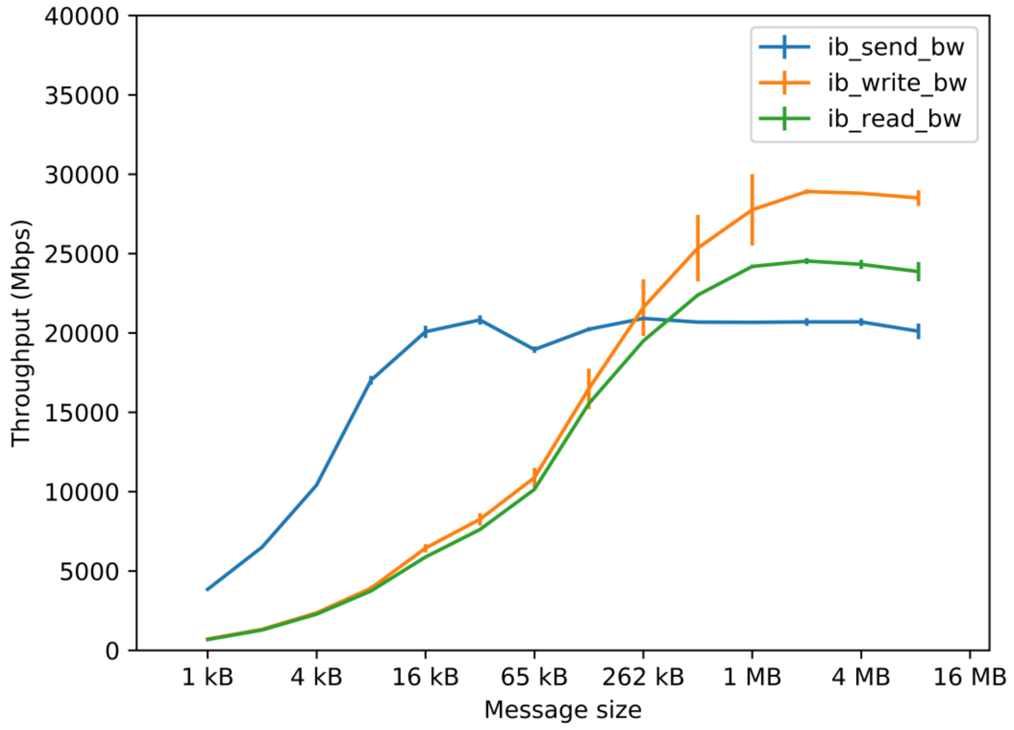
Patrick MacArthur, Ph.D. Candidate

University of New Hampshire

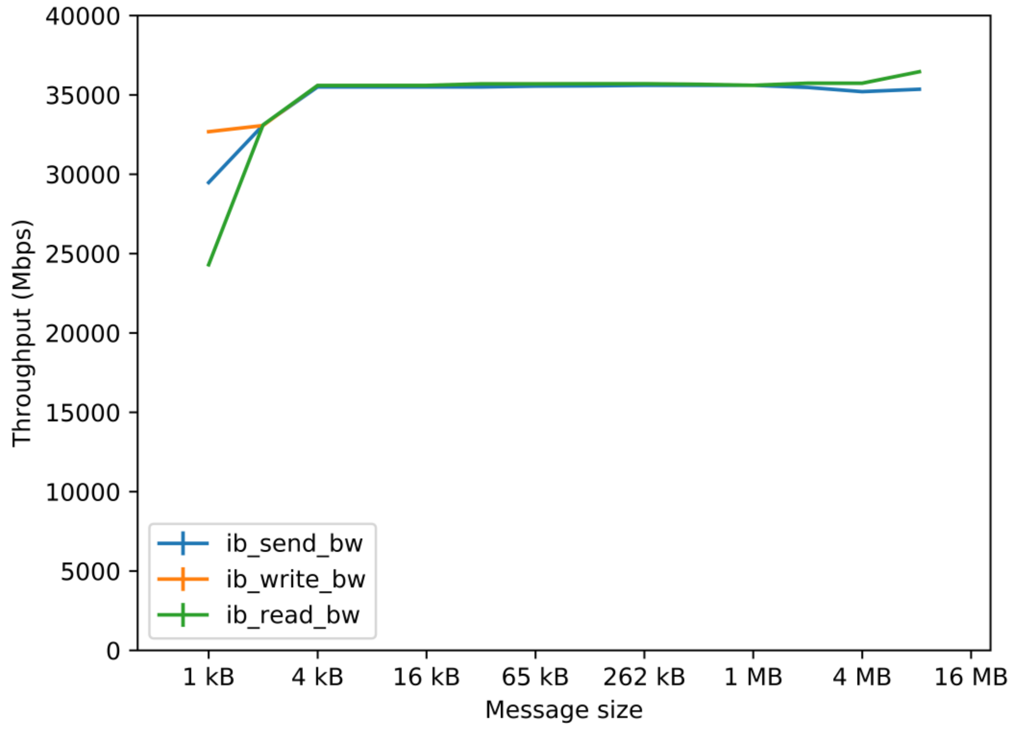
BACKUP

RAW VERBS: THROUGHPUT VS. HARDWARE RNIC

urdma on Intel XL710

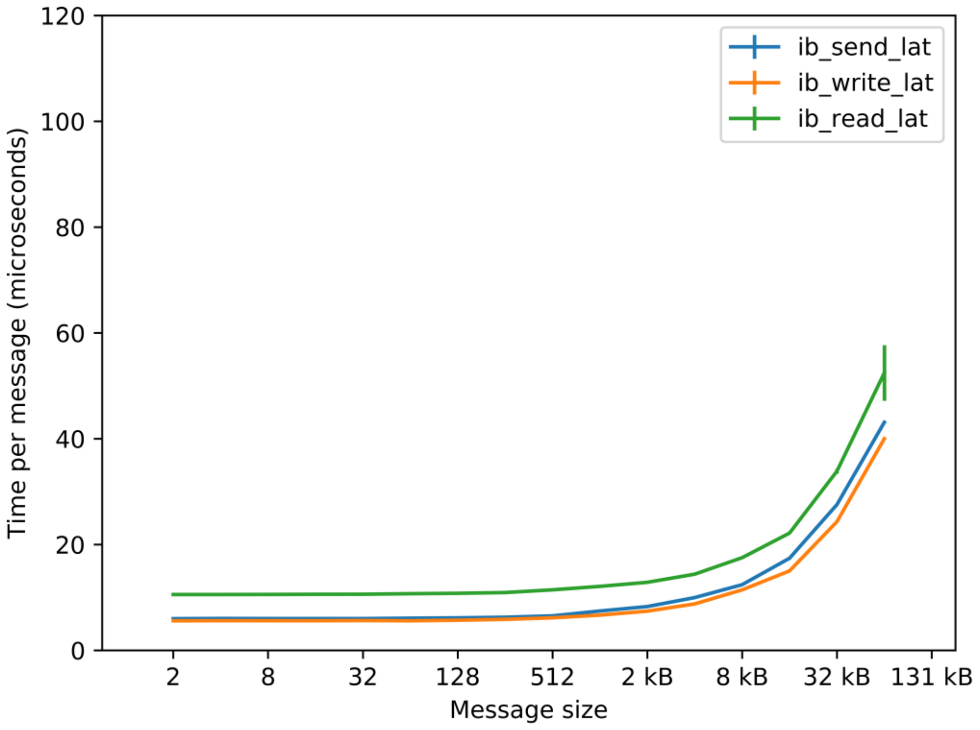


Chelsio T580-LP-CR iWARP



RAW VERBS: LATENCY VS. HARDWARE RNIC

urdma on Intel XL710



Chelsio T580-LP-CR iWARP