



A General-purpose API for iWARP and InfiniBand

Robert D. Russell

*InterOperability Laboratory
University of New Hampshire*

Outline

- Remote Direct Memory Access (**RDMA**)
- Open Fabrics Alliance (**OFA**)
- InterOperability Laboratory (**IOL**)
- Extended Sockets API (**EXS**-API)
- UNH-EXS in user space
- Mapping UNH-EXS onto RDMA
- Performance

RDMA

Remote Direct Memory Access

- Designed for “zero-copy” transmission
- Designed for low CPU overhead
- Several types available
 - **IB** - InfiniBand
 - Switched-fabric interconnect for clusters
 - **iWARP** – Internet Wide Area RDMA Protocol
 - IETF standard
 - Operates over legacy TCP/IP
 - **CEE/RDMAoEE, Myrinet, QsNet II**

iWARP

- A protocol stack for RDMA over TCP/IP
- Not limited to clusters – goes over Internet
- Three standards defined by IETF Remote Direct Data Placement Working Group
 - **RDMAP** (RFC5040) to control DDP coherently
 - **DDP** (RFC5041) to segment and place data
 - **MPA** (RFC5044) to align frames in TCP stream
- Hardware **RNIC** – RDMA aware NIC

OpenFabrics

- OpenFabrics Alliance (**OFA**)
 - www.openfabrics.org
- OpenFabrics Enterprise Distribution (**OFED**)
 - www.openfabrics.org/downloads/OFED/
 - Software interface to **IB** and **iWARP**
 - Open-source software for Linux and Windows
 - Linux user-space and kernel-space interfaces
 - Officially part of Linux kernel 2.6.20+ distributions

University of New Hampshire InterOperability Lab

Neutral 3rd party testing
Staffed largely by students

www.iol.unh.edu

20+ networking and storage
technologies

- OFA-UNH-IOL Logo Program

• www.iol.unh.edu/services/testing/ofa/

Extended Sockets (EXS-API)

- Published by the Open Group
- Extensions to “Standard” Sockets
- Two Major New Features
 - **Memory Registration** for “zero-copy” I/O
 - Event-based **Asynchronous I/O**
- Useful for high-level access to RDMA

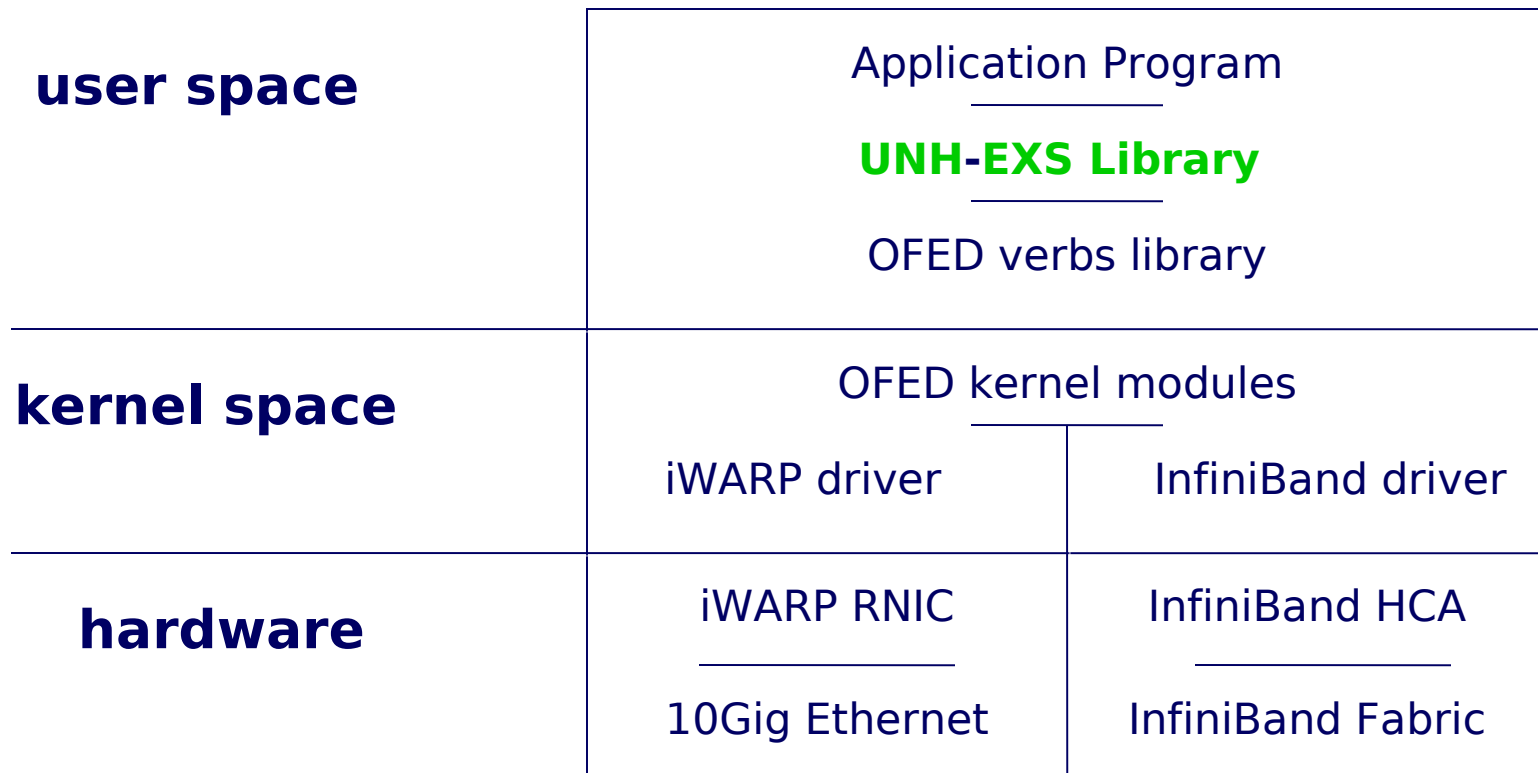
UNH-EXS Overview

- A general-purpose user-space interface to
 - InfiniBand
 - iWARP
 - CEE/RDMAoEE ??
- Extends widely-known socket concepts with
 - “zero-copy” memory-to-memory transfers
 - asynchronous threaded operation
- Provides SOCK_STREAM and SOCK_SEQPACKET RDMA transfers

UNH-EXS Library

- “thin layer” running entirely in **user space**
- Designed for use by user threads in Linux
- **Utilizes OFED** verbs library to access RDMA
- Requires **no change** to OFED or Linux
- **Extends** Open Group ES-API specifications
 - Why?
 - ES-API designed to run in kernel
 - ES-API requires modification to existing socket functions in the kernel

UNH-EXS Stack



EXS Programming

Two Major differences from “normal” sockets
(otherwise, just like “normal” sockets)

- **Memory registration**
 - to lock memory regions for “zero-copy” I/O
- **Event queues**
 - to determine asynchronous I/O completion

Memory Registration

- `exs_mregister()`

Registers a region of user virtual memory for “zero-copy” I/O

- `exs_mderegister()`

Unregisters previously registered memory region

- Memory regions specified in I/O operations

- `exs_send()`

- `exs_recv()`

Asynchronous Event Queues

- `Exs_qcreate()`
 - Creates a new event queue object
- `Exs_qdelete()`
 - Deletes an existing event queue object
- `Exs_qdequeue()`
 - Blocks until event is posted to event queue object
- Event posted to a queue when an I/O completes
`exs_connect()`, `exs_close()`, `exs_send()`, `exs_rcv()`

Parameters to `exs_send()`

1. Socket file descriptor - **normal**
2. Buffer containing data to send - **normal**
3. Number of bytes of data to send - **normal**
4. Flags - **normal**
5. Event queue for posting completion event - **new**
6. User-defined request identification - **new**
7. Registered memory region for data buffer - **new**

Parameters to `exs_recv()`

1. Socket file descriptor - **normal**
2. Buffer into which data is received - **normal**
3. Max number of bytes of data to read - **normal**
4. Flags - **normal**
5. Event queue for posting completion event - **new**
6. User-defined request identification - **new**
7. Registered memory region for data buffer - **new**

Other UNH-EXS functions

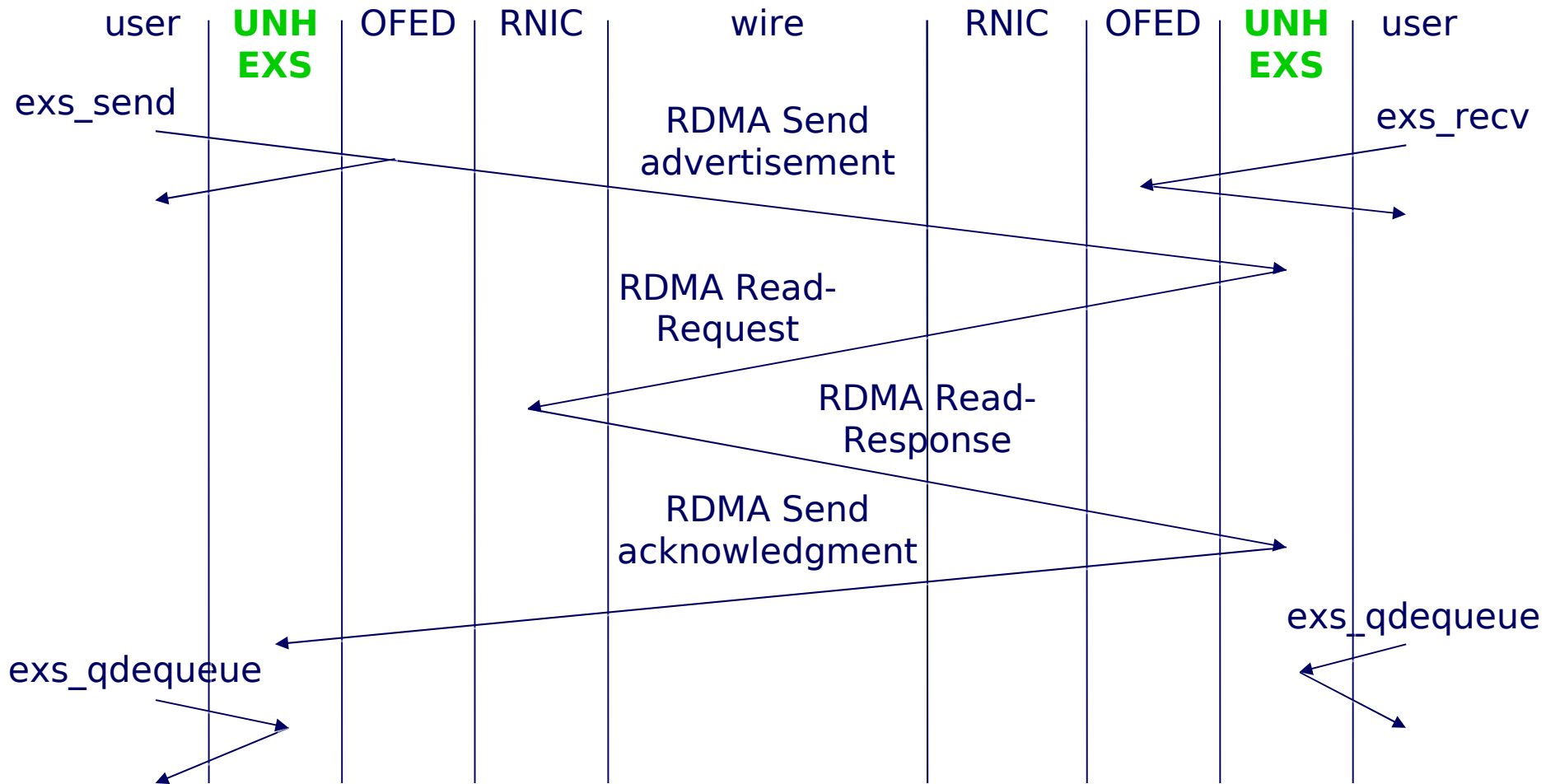
- `exs_accept()` - ES-API standard
- `exs_bind()` - UNH extension
- `exs_close()` - UNH extension
- `exs_connect()` - ES-API standard
- `exs_fcntl()` - UNH extension
- `exs_init()` - ES-API standard
- `exs_listen()` - UNH extension
- `exs_socket()` - UNH extension

Mapping UNH-EXS onto RDMA

Transfer controlled by **recipient** of data

- `exs_send()` translates into short RDMA “send” to **advertise** sender’s data buffer to receiver
- `exs_rcv()` sets up asynchronous wait for advertisement, then issues RDMA “read_request” to **asynchronously transfer data directly** from sender’s memory into receiver’s memory
- asynchronous completion of transfer translates into short RDMA “send” of **ACK** back to sender

Typical EXS Data Transfer



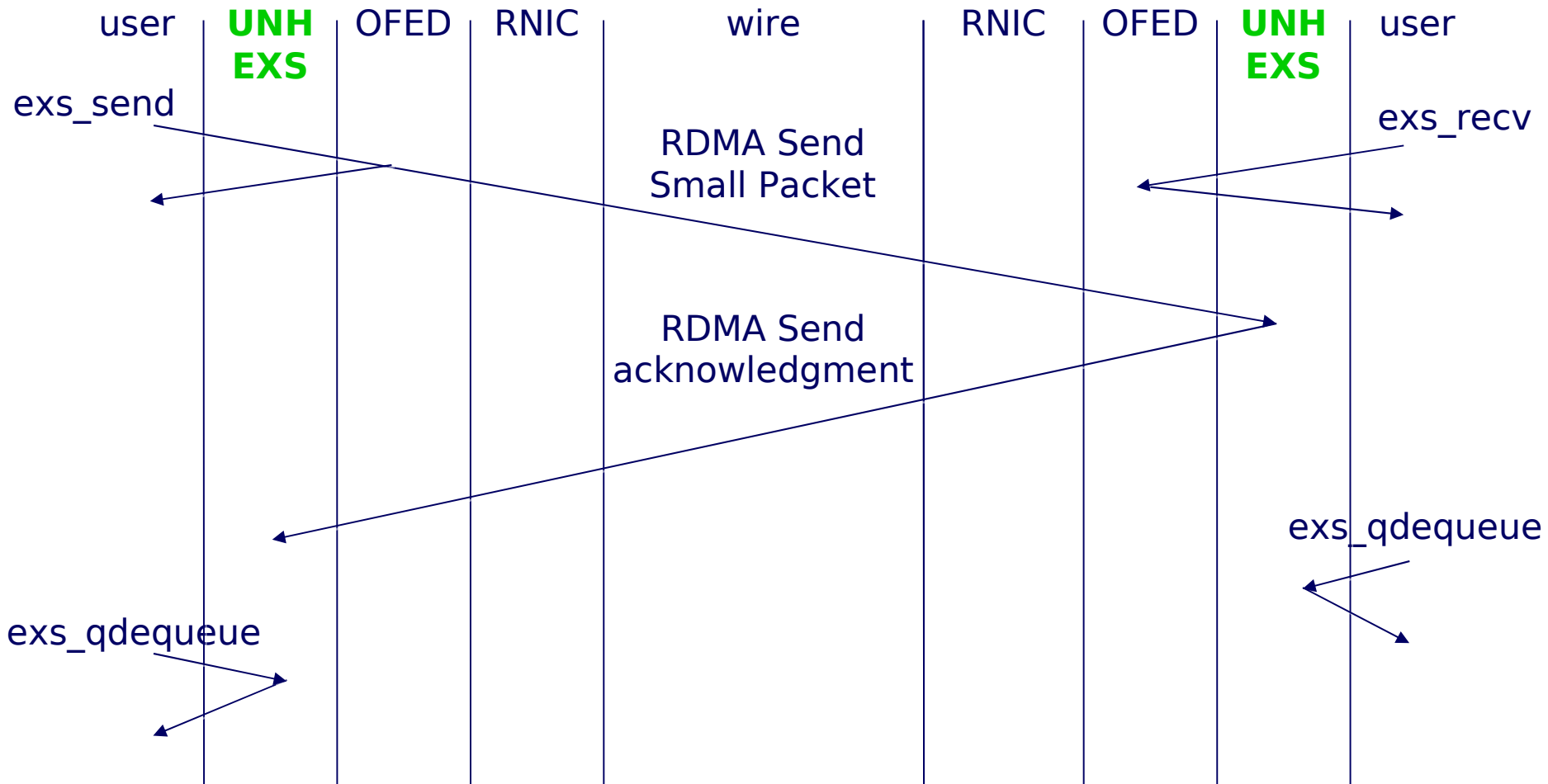
UNH-EXS internal flow control

- Each side maintains local “send credit” value
- Initial credits negotiated when connection established (hidden from user)
- Exs_send() decrements local “send credit” by 1, blocks if none remaining
- Receipt of ACK increments local “send credit” by 1, unblocks any waiting exs_send()

UNH-EXS small packets

- Size limit **configured by user** with `exs_fcntl()` prior to connection establishment
- Causes “small” amounts of data in `exs_send()` to travel as “**immediate data**” in the advertisement
- **Saves** extra RDMA “`read_request`” / “`read_response`” exchange (hidden from user)
- **Costs** extra data copy on receiver when data in advertisement is matched with `exs_rcv()`

“Immediate” Data Transfer



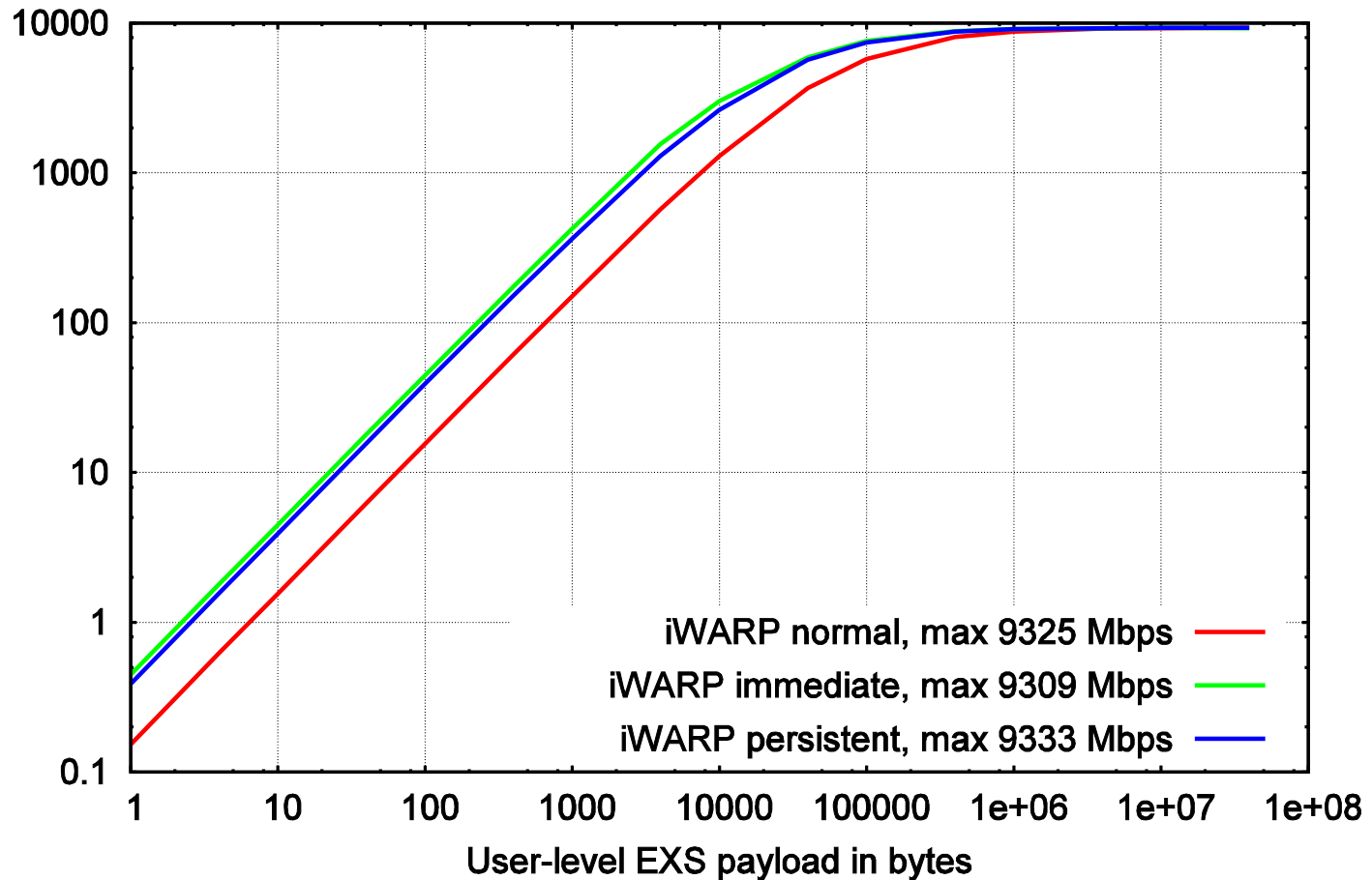
UNH-EXS persistent sends

- Normally bytes sent by `exs_send()` are consumed by `exs_rcv()`
- `EXS_PERSISTENT` flag in `exs_send()` allows repeated `exs_rcv()` with no ACK back to sender
- Uses:
 - Real-time sampling of sender's memory
 - Remote debugging of sender's memory
 - Transaction exchanges

Performance Measurements

- Platform configuration
 - 2 dual-core 64-bit Intel 2.66 GHz processors
 - 4 Gigabytes memory
 - 1 NetEffect (Intel) 10 Gigabit/second RNIC
 - 1 Mellanox 8 Gigabit/second Lion Mini SDR HCA
- Metrics measured
 - User-level throughput in Megabits/second
 - CPU utilization as a percentage of one CPU
 - One-way latency in microseconds

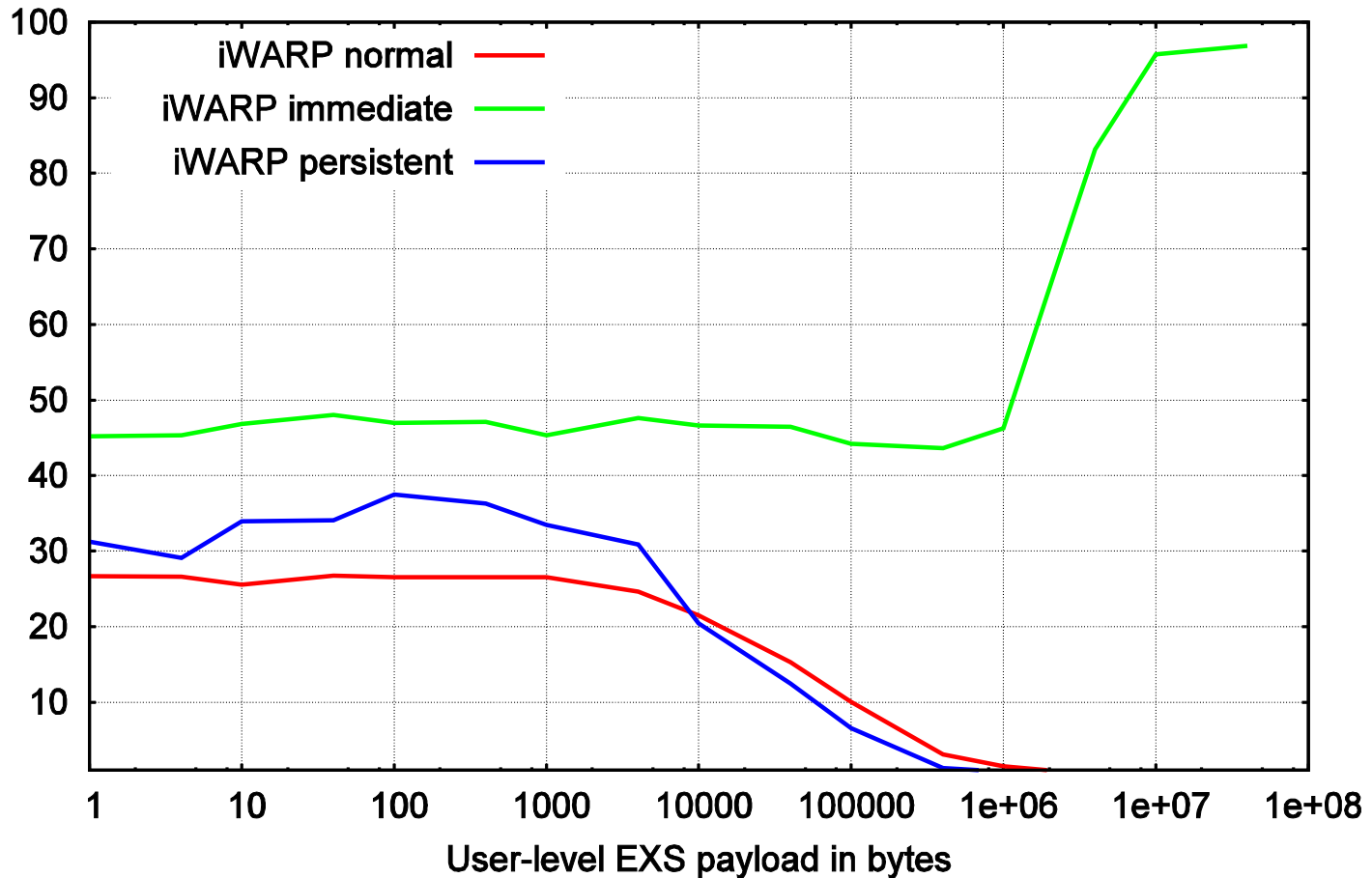
EXS/iWARP throughput in Mbps



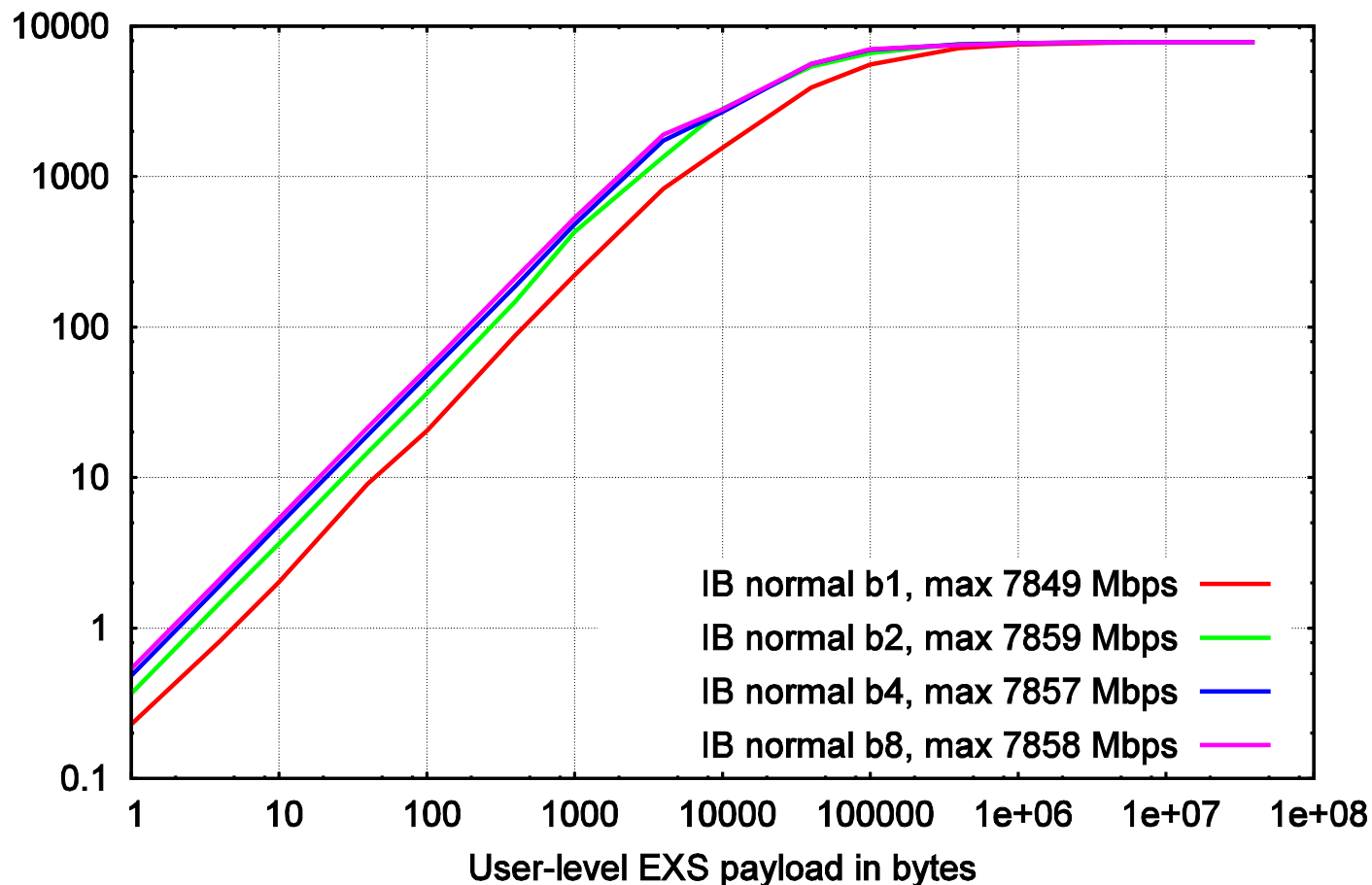
Bandwidth Utilization

• Standard Ethernet Frame	1538 bytes
_ Ethernet gap, preamble, header and CRC	38 bytes
_ IP and TCP headers	40 bytes
_ MPA, DDP, RDMAP headers and MPA CRC	20 bytes
• Available user payload data	1440 bytes
• Percentage available for data	93.63%
• Percentage actually utilized	93.33%

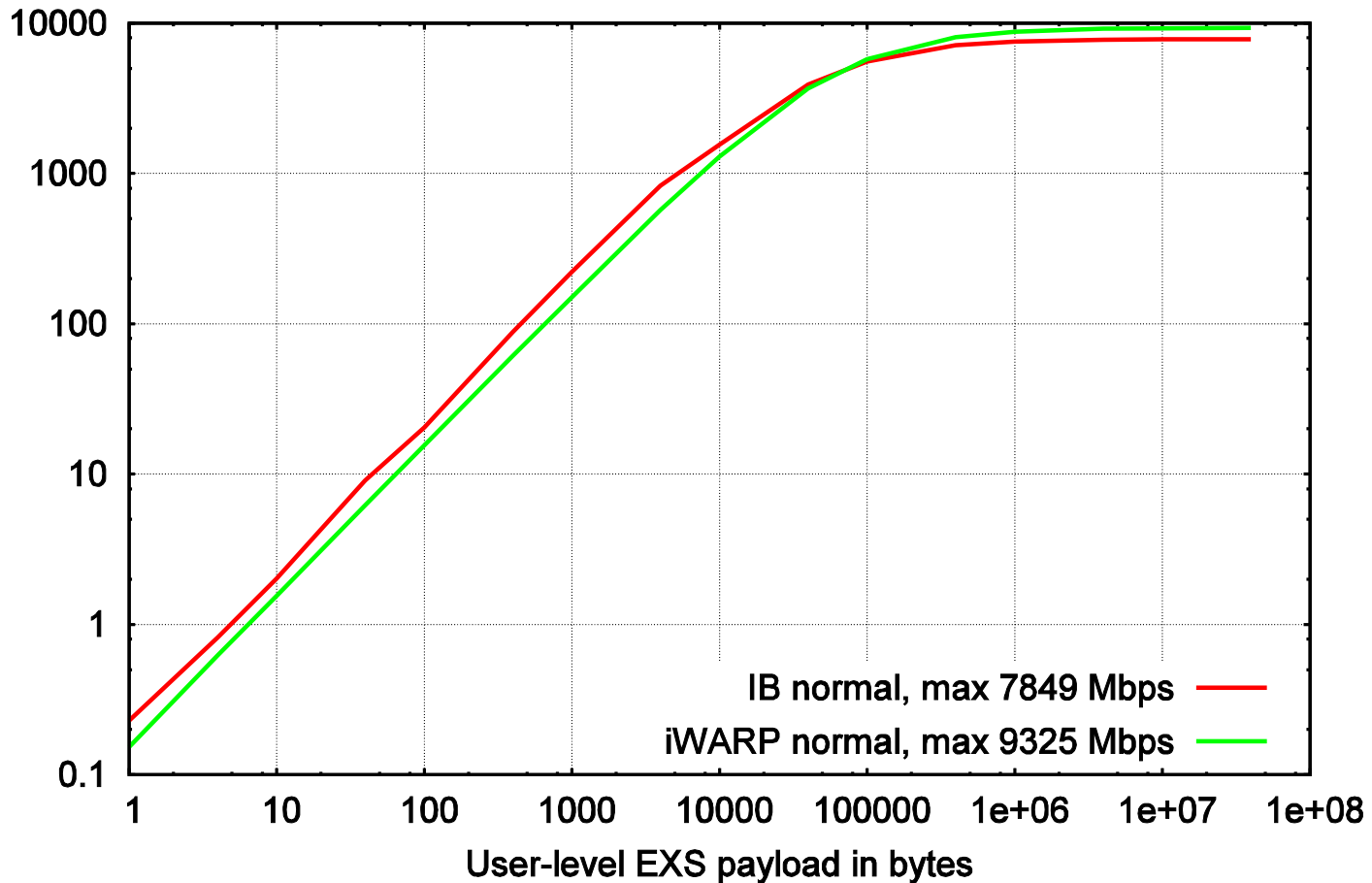
EXS/iWARP Percent Use of 1 CPU



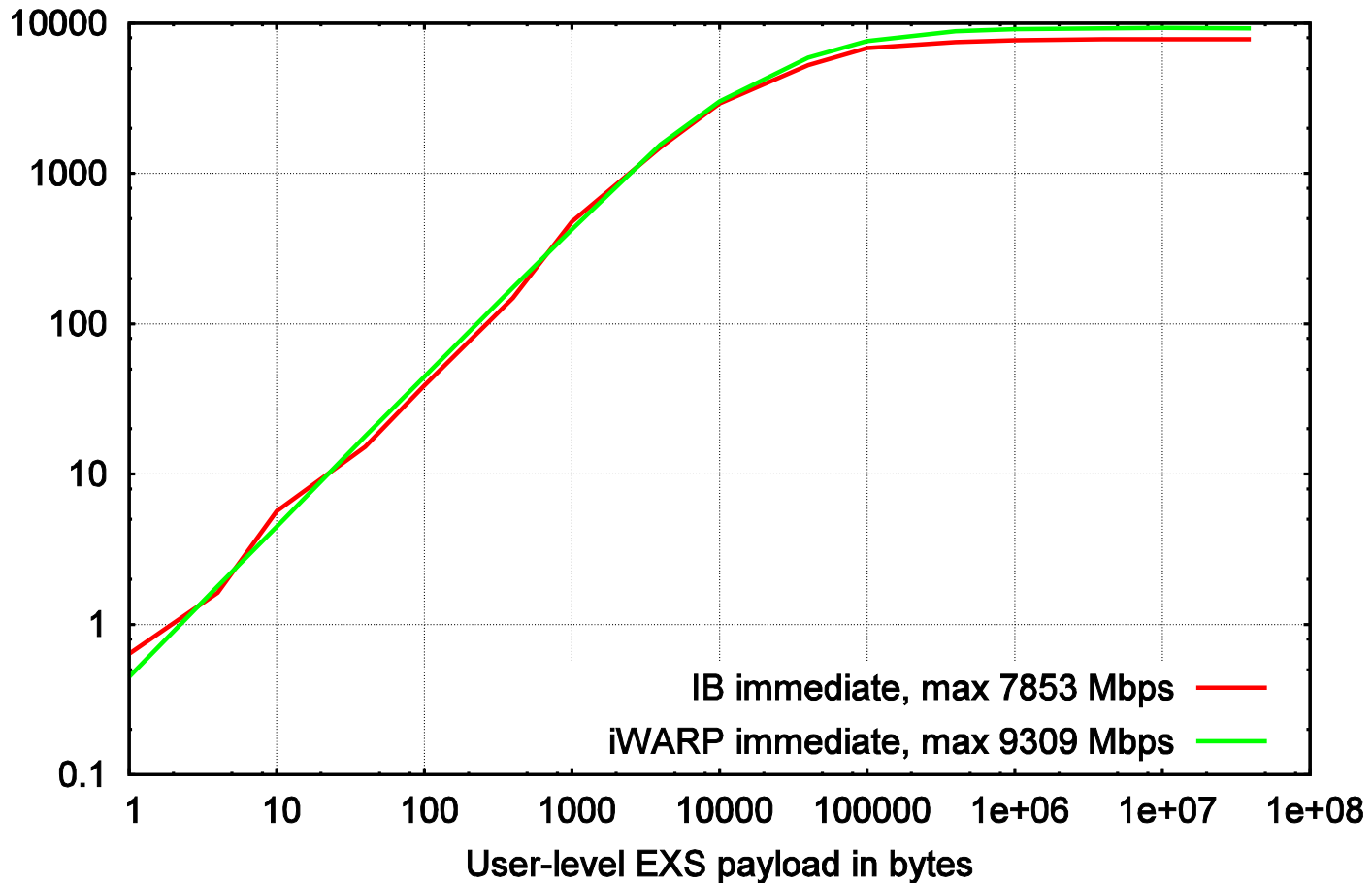
EXS/IB throughput in Mbps



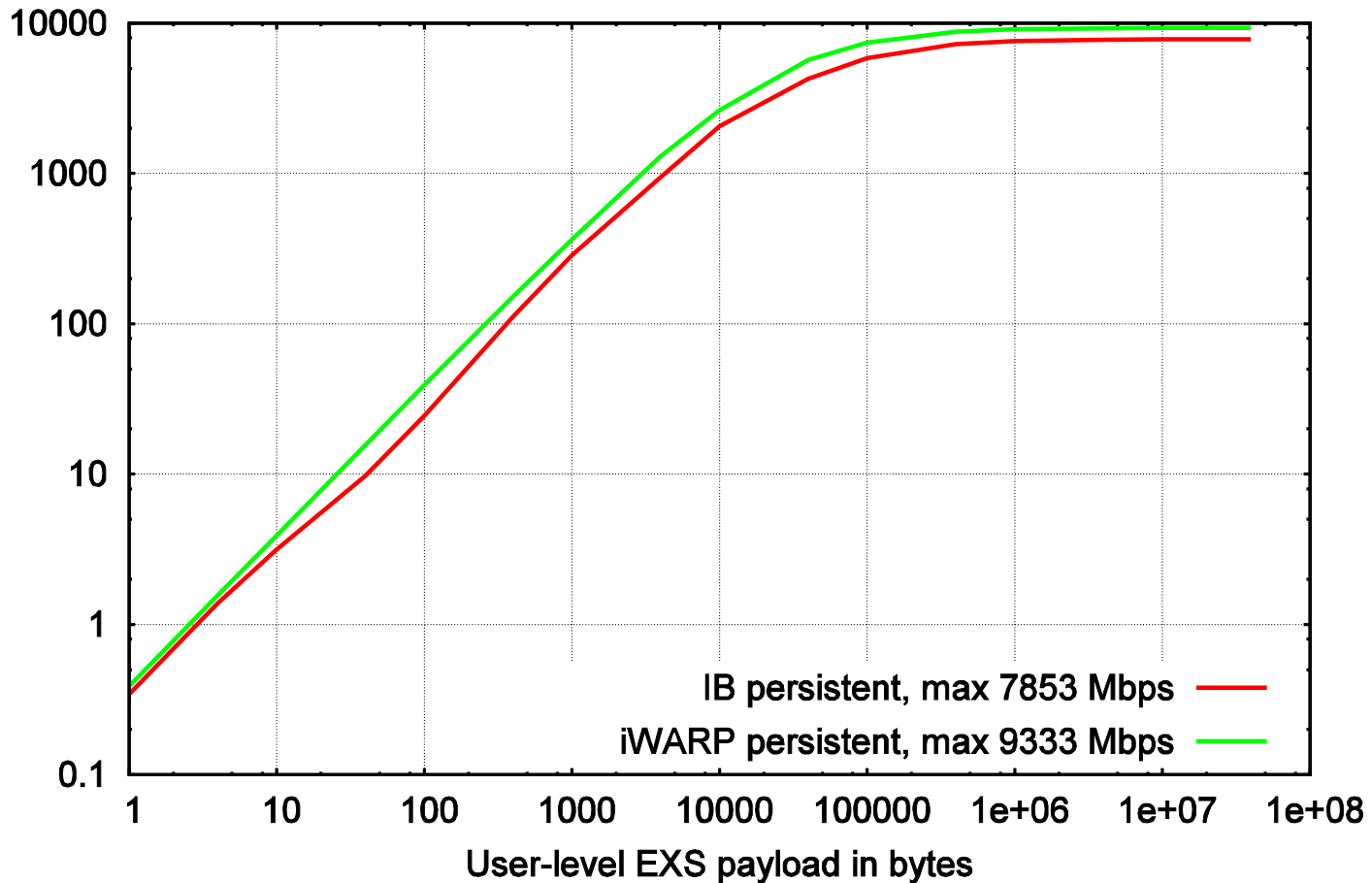
IB vs iWARP throughput with normal sends



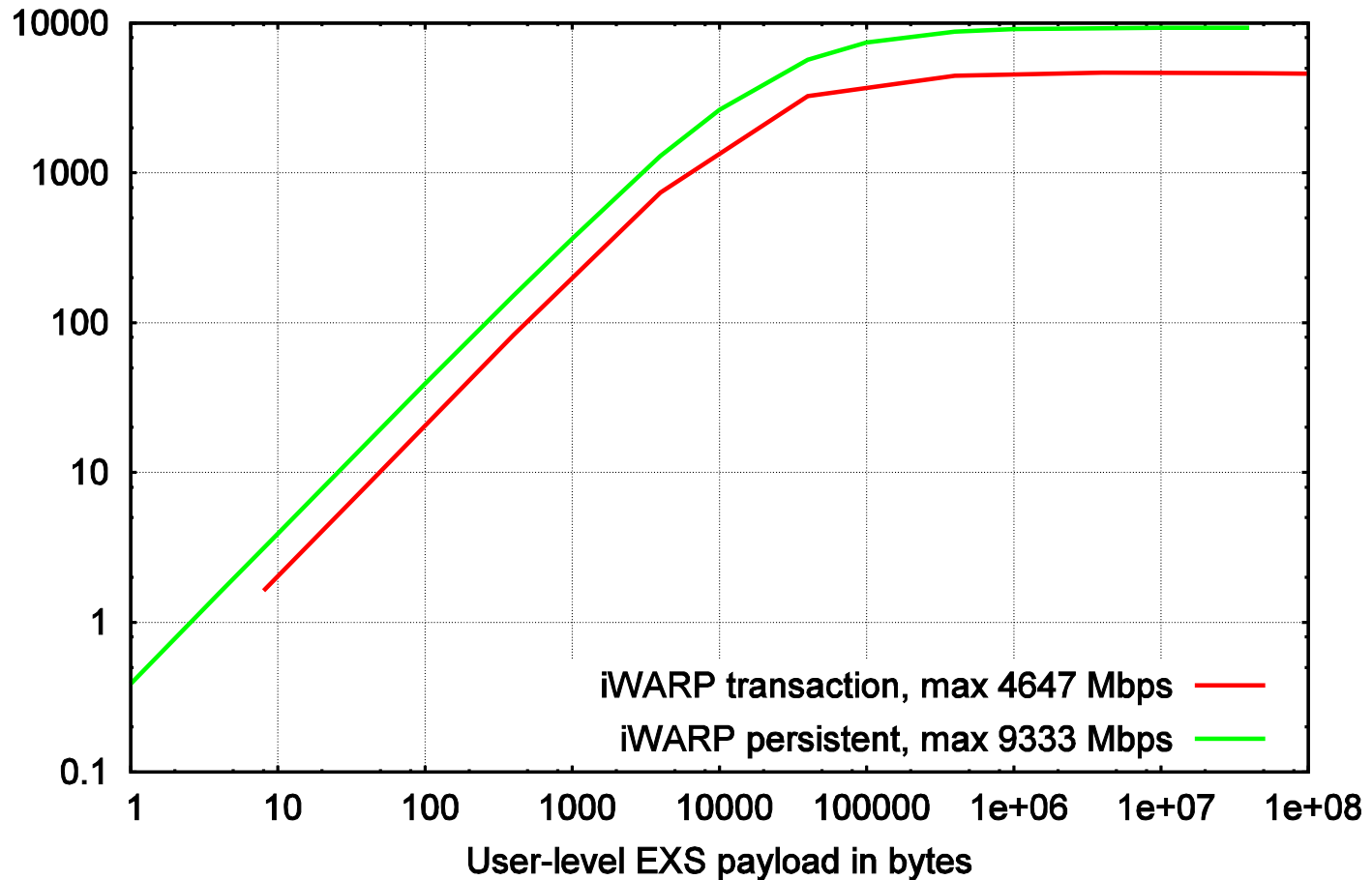
IB vs iWARP throughput with immediate sends



IB vs iWARP throughput with persistent sends



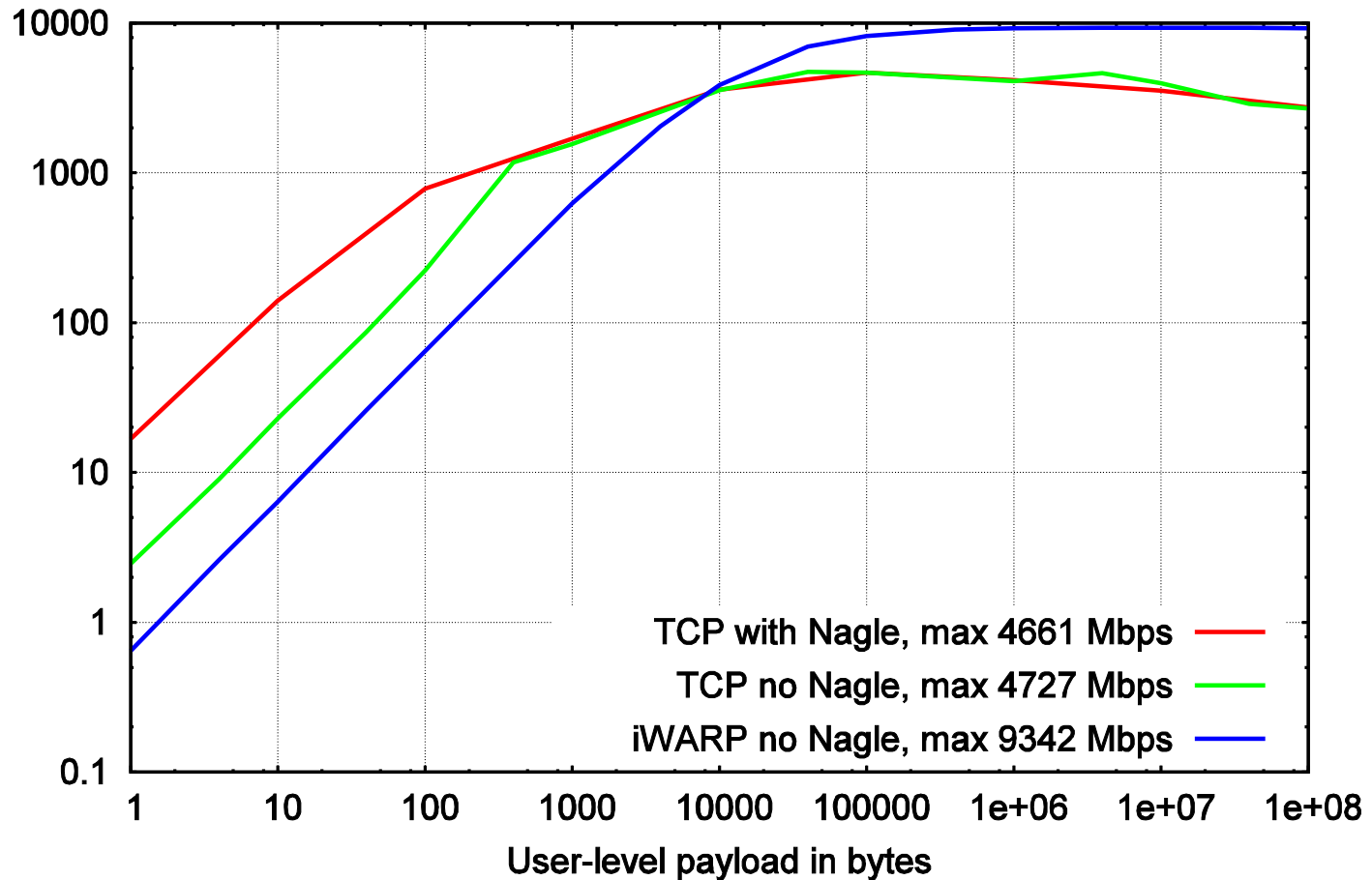
iWARP throughput with transaction vs persistent sends



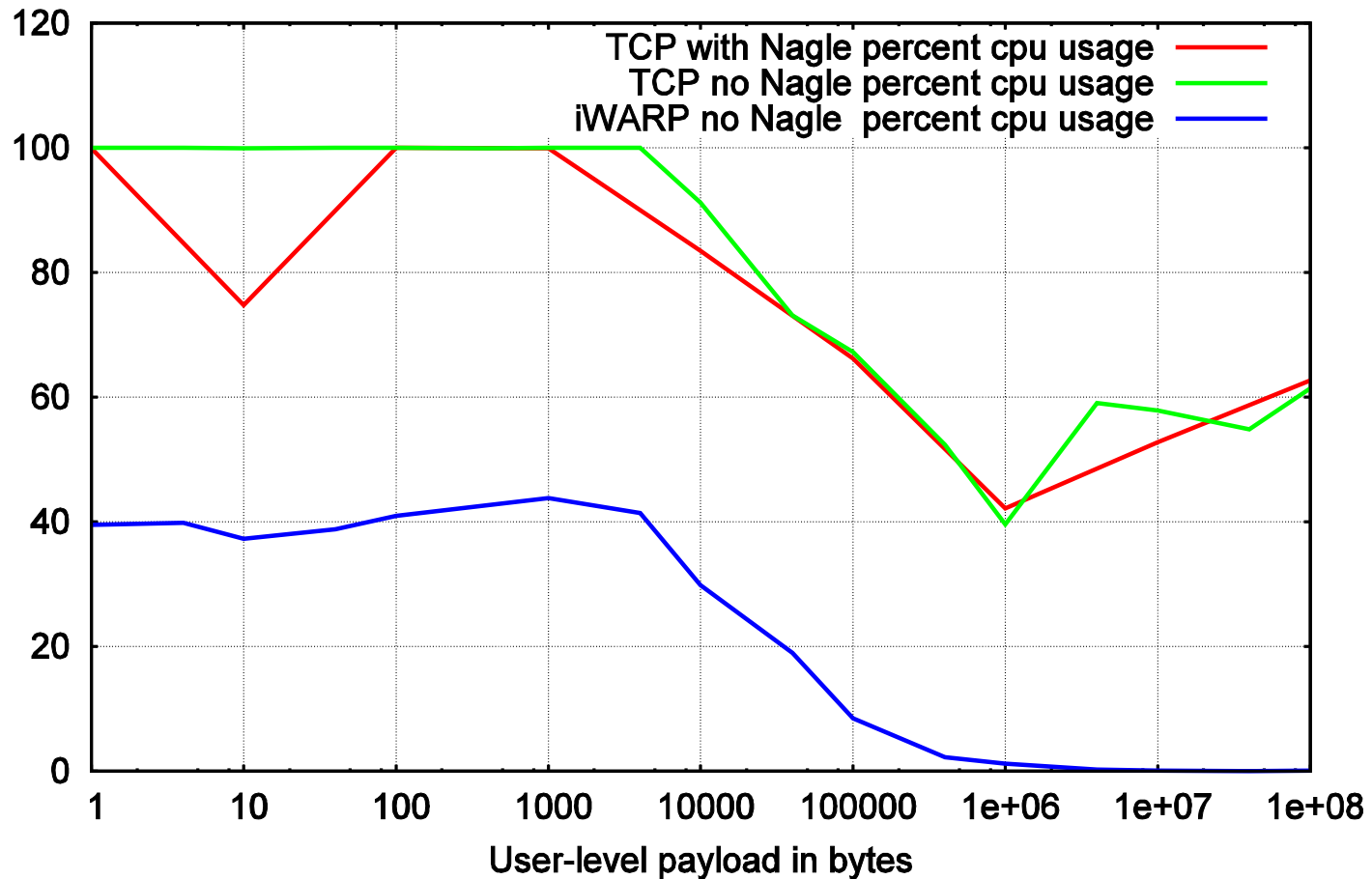
1-way latency in microseconds

payload bytes	normal iWarp	normal IB	immed iWarp	immed IB	persist iWARP	persist IB
1	33	33	17	19	10	16
10	33	33	17	19	10	15
100	34	33	17	18	10	15

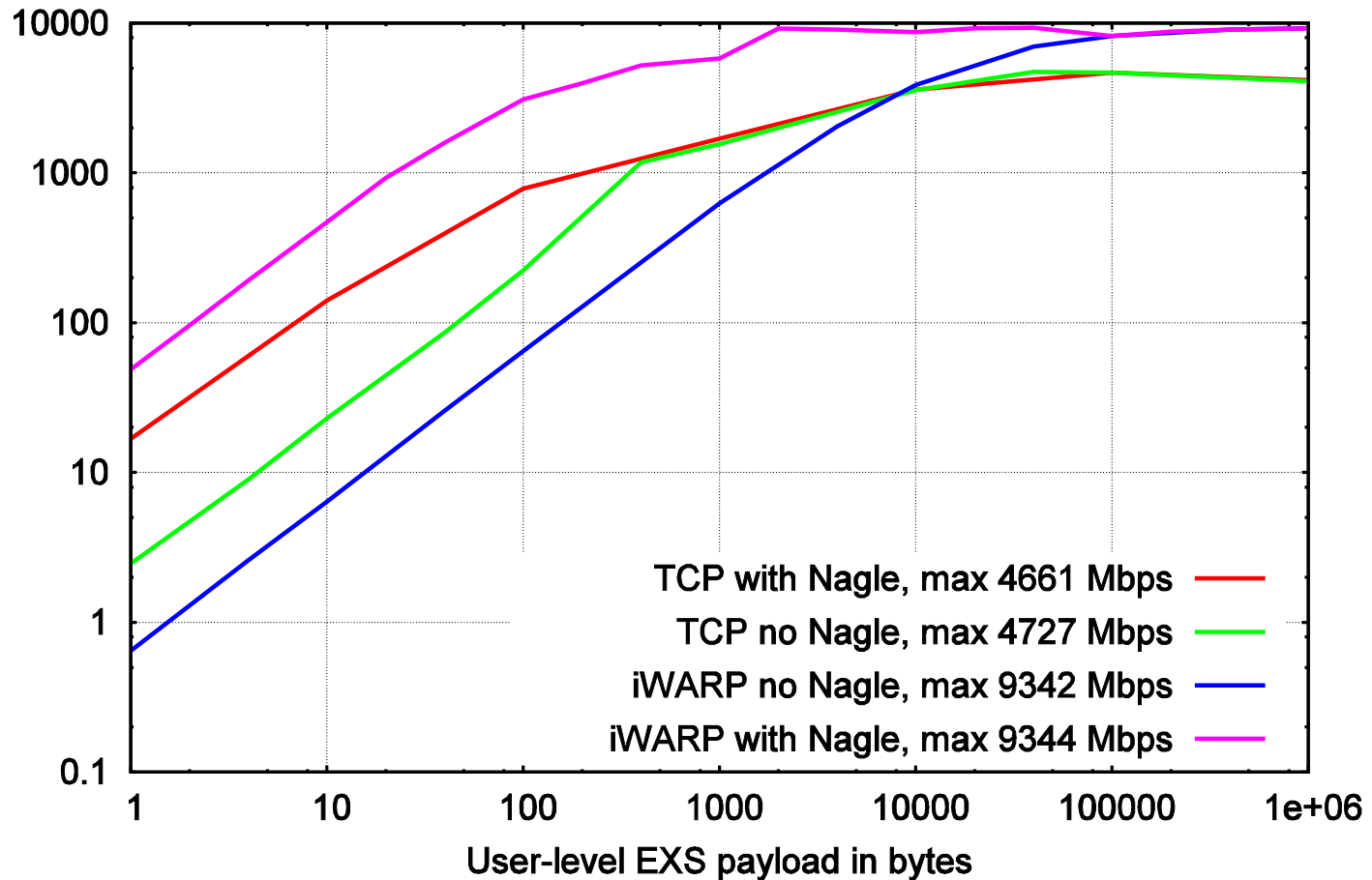
iWARP vs TCP throughput in Mbps



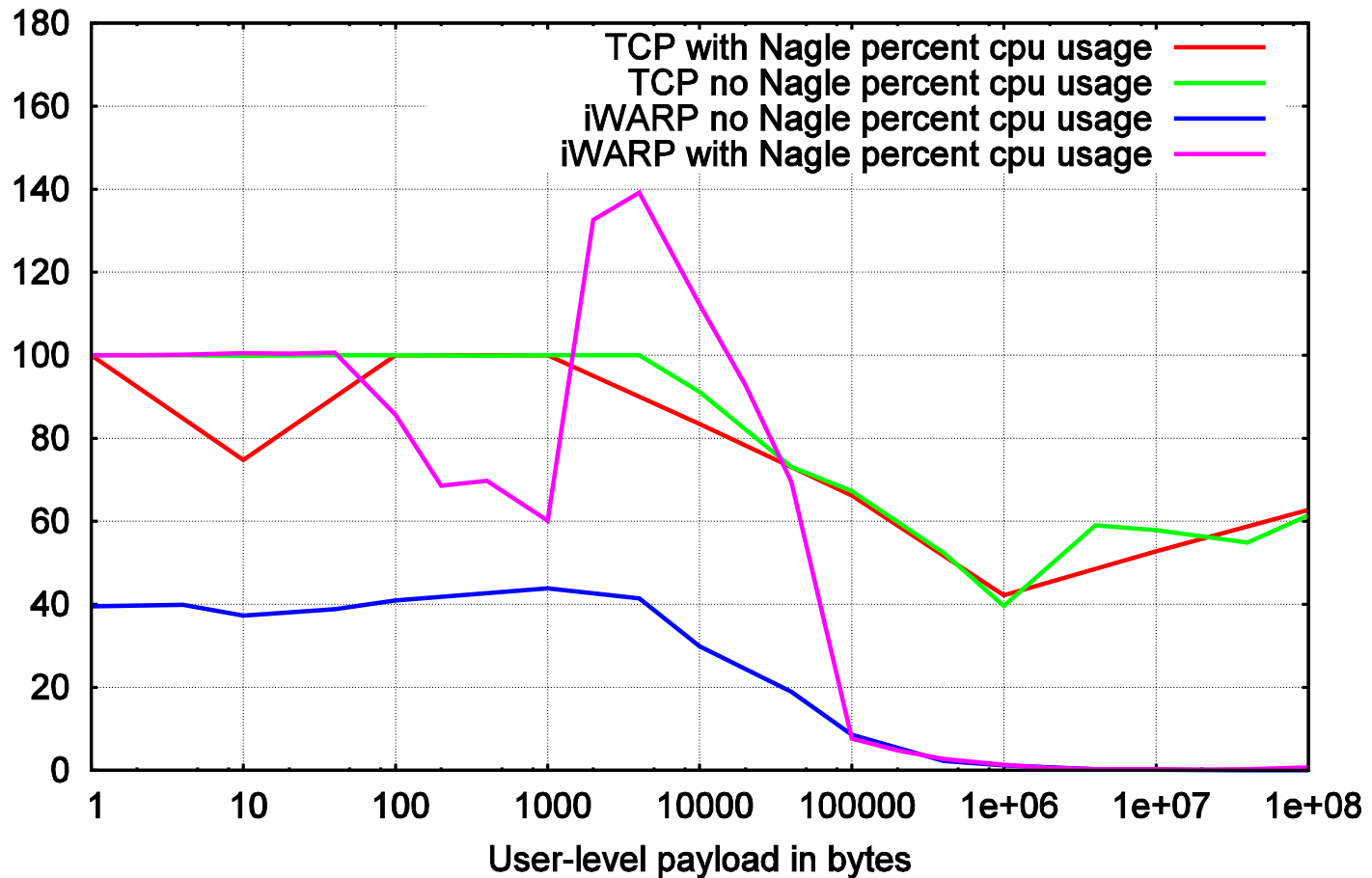
iWARP vs TCP cpu usage



iWARP vs TCP throughput in Mbps



iWARP vs TCP cpu usage



Conclusions

- Application programs using UNH-EXS do attain
 - High bandwidth utilization
 - Low CPU overhead
- Good performance is attainable with user- space UNH-EXS library
- Runs transparently over iWARP and InfiniBand
- Application programs using UNH-EXS can tune their performance

Continuing Work on UNH-EXS

- Finish implementing and tuning
- Compare standard and jumbo Ethernet frames
- Look at multicast transmission
- Look at richer set of select/sync transmissions
- Look at multiple simultaneous connections
- Look at CEE/RDMAoEE
- Look at porting to Windows

UNH-EXS Summary

- A general-purpose user-space interface to
 - InfiniBand
 - iWARP
 - CEE/RDMAoEE ??
- Extends widely-known socket concepts with
 - “zero-copy” memory-to-memory transfers
 - asynchronous threaded operation
- Runs entirely in user-space - easily portable
- Utilizes OFED user-space library and stack

Questions?

Thank you!

Robert D. Russell

rdr@unh.edu