EXS and RoEE from UNH-IOL

Based in part upon Dr. Robert Russell's slides

Presenter: Mikkel Hagen

Outline

- InterOperability Laboratory (IOL)
- Extended Sockets API (EXS-API)
- EXS in user space
- Mapping EXS onto RDMA
- Performance
- RDMA over Enhanced Ethernet (RoEE)

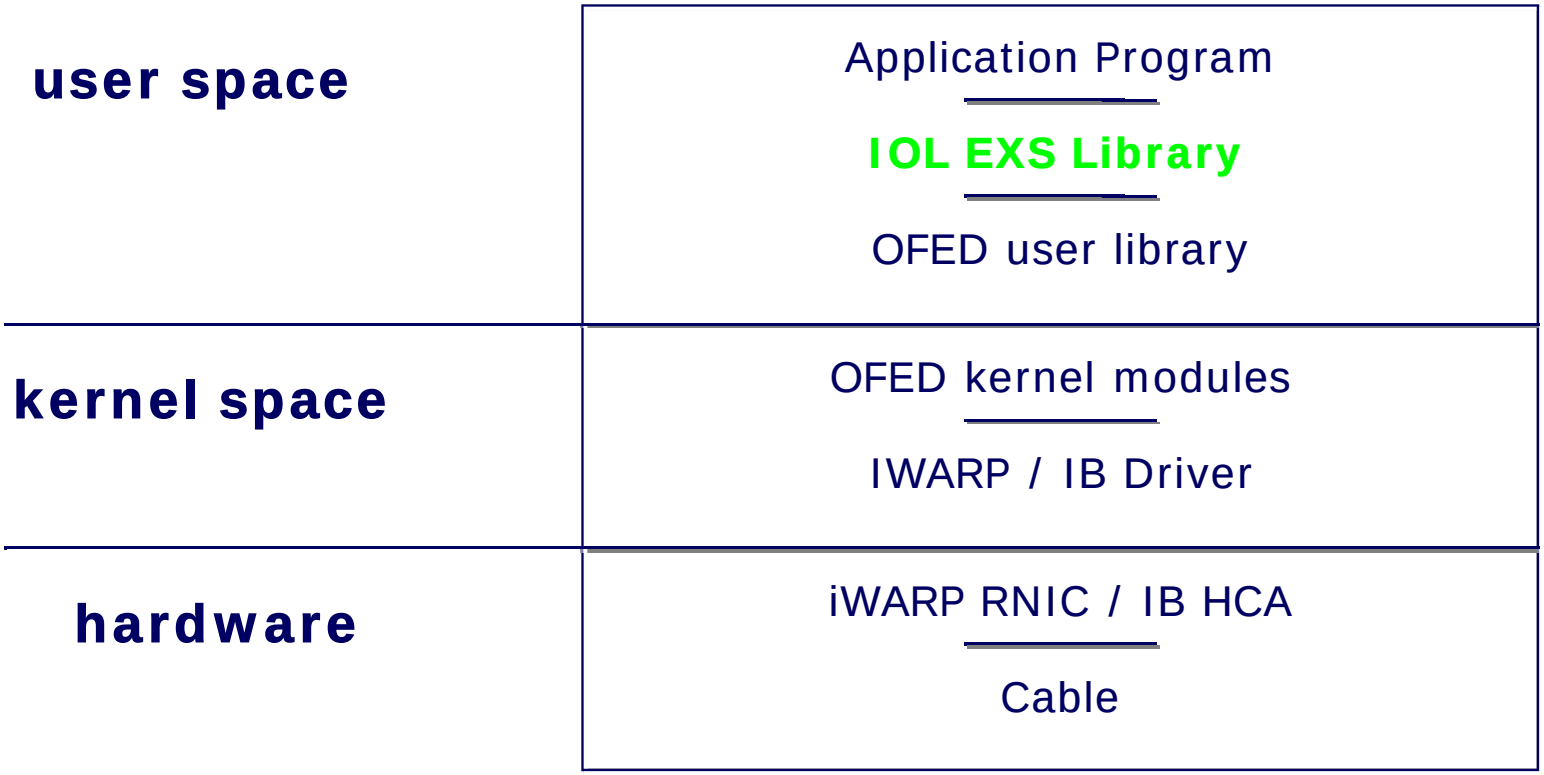
UNH-IOL

- Neutral 3rd party testing
- Hosts the OFA-IWG interoperability cluster
  - ~30 hosts, with targets and switches from many different vendors
  - Both iWARP and IB
- Maintains the OFW-IWG Logo List
- Related testing through:
  - iWARP, 10Gig, IB/10G Cable Testing and Data Center Bridging Consortia

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

IOL EXS Stack for OFA



IOL EXS

- Runs entirely in user space
- Utilizes OFED library to access RDMA
- Requires no change to OFED or Linux
- Extends Open Group ES-API specifications
  - ES-API designed to run in kernel
  - ES-API requires modifications to existing socket functions

EXS Programming

- Two major differences from “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 are used in I/O operations
  - exs_send()
  - exs_recv()

Event Queues

- exs_qcreate()
  - Creates a new queue object
- exs_qdelete()
  - Deletes an existing queue object
- exs_qdequeue()
  - Blocks until event is posted to queue object
- Event posted to a queue when an I/O completes

Parameters to exs_send()

- Socket file descriptor – normal
- Buffer containing data to send – normal
- Number of bytes of data to send – normal
- Flags – normal
- Event queue for posting completion event – new
- User-defined request identification – new
- Registered memory region for data buffer - new

Parameters to exs_recv()

- Socket file descriptor – normal
- Buffer into which data is received – normal
- Max number of bytes of data to read – normal
- Flags – normal
- Event queue for posting completion event – new
- User-defined request identification – new
- Registered memory region for data buffer - new

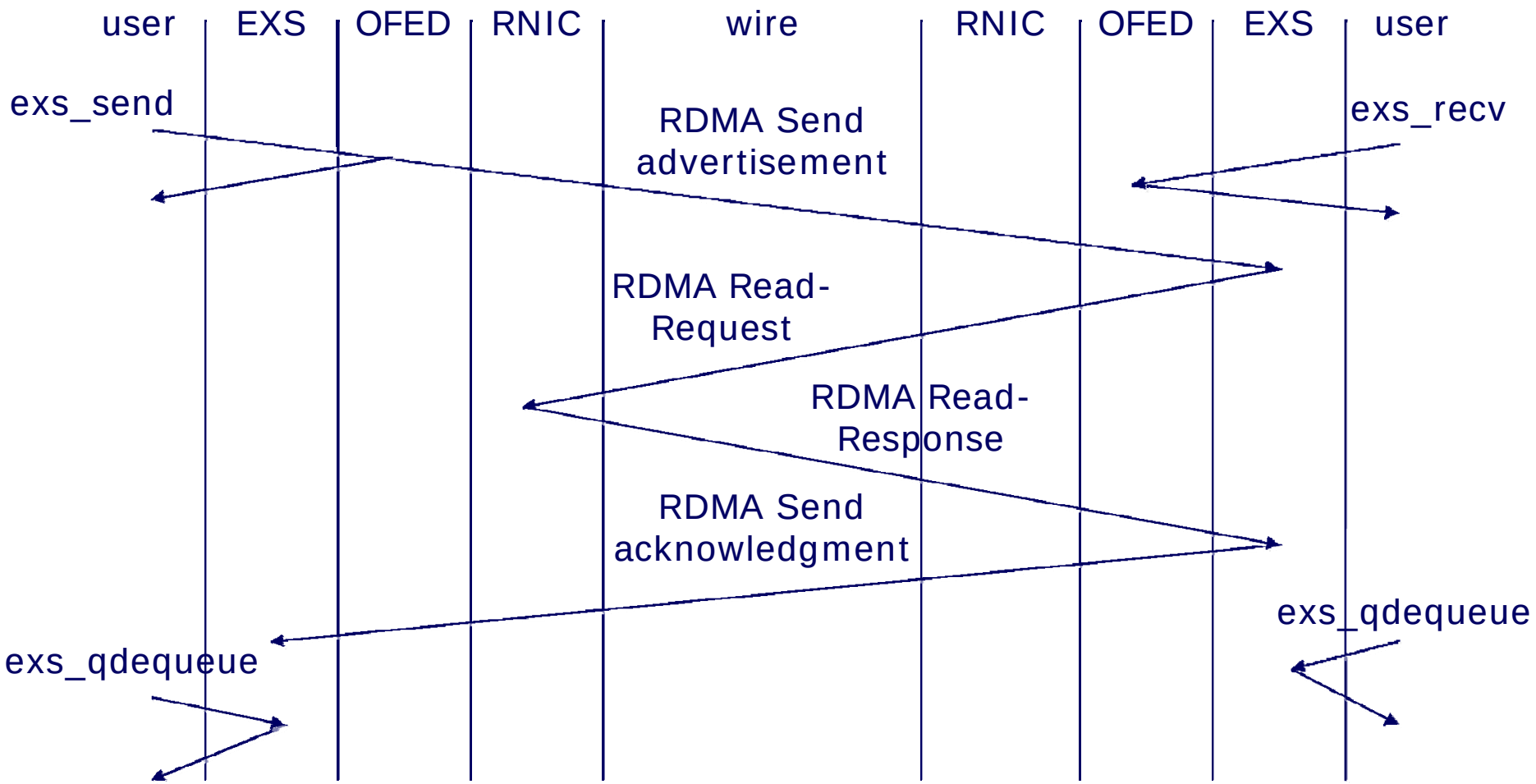
Other EXS functions

- exs_init()
- exs_fcntl() - IOL Extension
- exs_socket() - IOL Extension
- exs_bind() - IOL Extension
- exs_listen() - IOL Extension
- exs_connect()
- exs_accept()
- exs_close() - IOL Extension

Mapping IOL-EXS onto RDMA

- Transfer controlled by recipient of data
- exs_send() translates into RDMA “send” to advertise sender's data buffer to receiver
- exs_recv() 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



IOL EXS Internal Flow Control

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

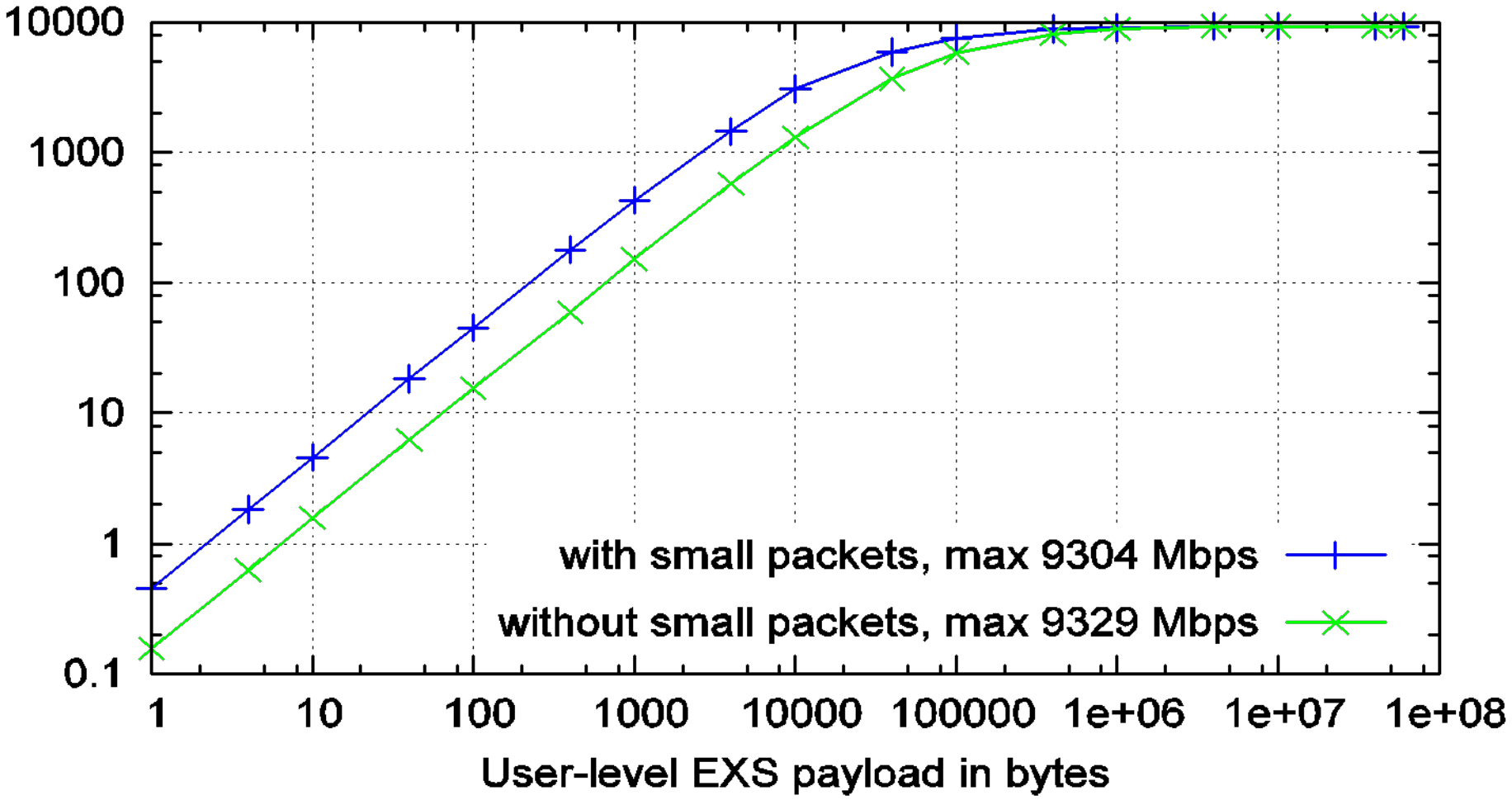
IOL-EXS Small Packets

- 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 advertisement is matched with exs_recv()

Initial Performance Measurements

- Platform configuration
  - 4 64-bit Intel 2.66 GHz processors
  - 4 Gigabytes memory
  - 1 NetEffect (Intel) 10 Gigabit/second RNIC
- Test
  - “blasting” data from one workstation to other
- Metrics measured
  - User-level throughput in Megabits/second
  - CPU utilization as a percentage of 1 CPU

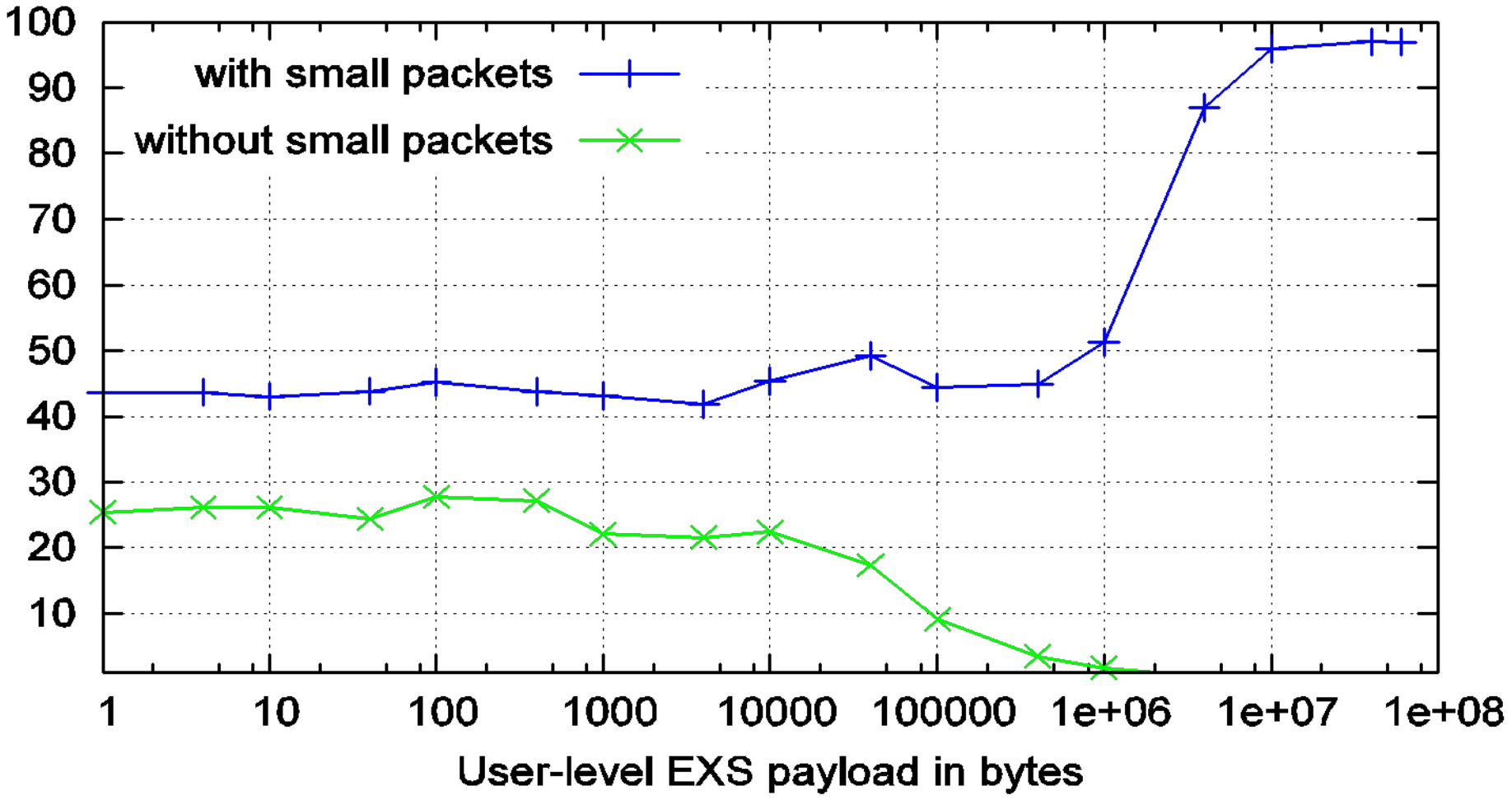
EXS Throughput in Mbits/sec



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 93.29%

Percent CPU Utilization



EXS Conclusions

- Application programs using EXS can attain
  - High bandwidth utilization
  - Low CPU overhead
- Application programs using EXS can tune their performance
- Programmers don't need to learn verbs in order to make use of RDMA technology

EXS Future Work

- Compare various RNIC hardware
- Compare standard and jumbo Ethernet frames
- Compare EXS over IB and iWARP
- Compare various applications
- Look at multiple outstanding transmissions
- Look at multiple simultaneous connections

RoEE

- With the introduction of Data Center Bridging standards into Ethernet, applications sensitive to loss can now be placed directly upon Ethernet such as Fibre Channel and RDMA
- DCB has four new technologies being introduced: PFC, ETS, CN, DCBX

Priority-based Flow Control (PFC)

- Introduces a new MAC Control Frame that extends existing PAUSE mechanism
- New frame has fields to define pause time for all eight defined priorities
- This allows end nodes to inhibit only the traffic classes that are over utilizing their fair share of bandwidth

Enhanced Transmission Selection (ETS)

- Adds another field, Priority Group ID (PGID), to frames
- Priorities are added to different groups and the bandwidth of the link is divided up between the groups.
- This allows network admins to provide minimum QoS bandwidth requirements to different traffic classes
- Rate limiters on the transmitter maintains the BW allocations

Congestion Notification (CN)

- Adds another field, FlowID, to frames
- Provides end-to-end flow control
- PFC causes back pressure congestion cascades that can pull in nodes and flows not directly involved in the congestion
- CN allows devices to limit congestion pro-actively and hopefully prevent this phenomenon

DCB Exchange (DCBX)

- Allows end nodes to communicate configuration and even actively configure each other
- Designed to provide a means of identifying config problems and limited means of fixing them

DCB and UNH-IOL

- UNH-IOL is currently testing all of the above technologies in the new Data Center Bridging Consortium
- Future events are already planned with EA and FCIA to test DCB and FCoE (FC over DCB)
- Future events may also include Enterprise iSCSI (iSCSI over DCB)
- First test plan is complete:
  - http://www.iol.unh.edu/services/testing/dcb/testsuites.php

RoEE and UNH-IOL

- UNH-IOL has over 20 years of experience working with standards bodies including IEEE, IETF, ISO, ITU, etc.
- UNH-IOL has worked with RDMA technologies including iWARP, IB and OFA for several years – testing, documenting and developing applications
- UNH-IOL has worked on DCB from the beginning and will lead the way with testing and developing this new technology
- UNH-IOL is uniquely qualified with experience in all areas touching RoEE

Contacts

- Mikkel Hagen
  - Research and Development
  - Phone: 1 (603) 862 5083
  - Email: mhagen@iol.unh.edu
- Bob Noseworthy
  - Chief Engineer
  - Phone: 1 (603) 862 0090
  - Email: ren@iol.unh.edu
- Website: http://www.iol.unh.edu