***I. J. Computer Network and Information Security*, 2016, 10, 12-22**
Published Online October 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2016.10.02

Low Level Performance Evaluation of InfiniBand with Benchmarking Tools

Eric Gamess
Central University of Venezuela, School of Computer Science, Caracas, Venezuela University of Puerto Rico, Department of Computer Science, San Juan, Puerto Rico
E-mail: eric.gamess@ciens.ucv.ve

Humberto Ortiz-Zuazaga
University of Puerto Rico, Department of Computer Science, San Juan, Puerto Rico
E-mail: humberto@hpcf.upr.edu

Abstract—InfiniBand is widely accepted as a high performance networking technology for datacenters and HPC clusters. It uses the Remote Direct Memory Access (RDMA) where communication tasks that are typically assigned to CPUs are offloaded to the Channel Adapters (CAs), resulting in a significant increase of the throughput and reduction of the latency and CPU load. In this paper, we make an introduction to InfiniBand and IP over InfiniBand (IPoIB), where the latter is a protocol proposed by the IETF to run traditional socket-oriented applications on top of InfiniBand networks. We also evaluate the performance of InfiniBand using different transport protocols with several benchmarking tools in a testbed. For RDMA communications, we consider three transport services: (1) Reliable Connection, (2) Unreliable Connection, and (3) Unreliable Datagram. For UDP and TCP, we use IPoIB. Our results show significant differences between RDMA and IPoIB communications, encouraging the coding of new applications with InfiniBand verbs. Also, it is noticeable that IPoIB in datagram mode and in connected mode have similar performance for small UDP and TCP payload. However, the differences get important as the payload size increases.

Index Terms—Computer Networks, Performance Evaluation, RDMA, InfiniBand, IP over InfiniBand, Benchmarking Tools.

I. Introduction

In a typical IP data transfer, an application produces the data to be sent in user memory. Through a system call, these data are copied into kernel memory where headers will be added, before been passed to the NIC (Network Interface Card) for transmission. On the reception side, the data are passed from the NIC to system buffer, where the headers will be used to determine the destination process before been removed. Then, a system call is required to transfer the data from kernel memory to user memory where the application will be finally able to use them. This process imposes extremely high CPU loads on the system. RDMA (Remote Direct Memory Access) communications differ from normal IP communications because they bypass kernel intervention in the communication process. That is with RDMA, the CA (Channel Adapter) directly places the application's buffer into packets on sending, and the content of the packets into the application's buffer on reception, without any intervention of the CPU. This allows a much better communication system with zero copy. Moreover, the CA also manages the splitting and assembly of messages into packets in RDMA, while IP fragmentation and TCP segmentation are in charge of the CPU in typical IP communications. As a result, RDMA provides high throughput and low latency while incurring a minimal amount of CPU load.

In the last few years, three major RDMA fabric technologies have emerged: InfiniBand [1][2][3], RoCE [4][5] (RDMA over Converged Ethernet), and iWARP [6][7] (internet Wide Area RDMA Protocol). InfiniBand has received special attention from many manufacturers and researchers, especially in the field of HPC (High Performance Computing). In this paper, we make a low level performance evaluation of InfiniBand as a communication system with different services of the transport layer, using benchmarks. We also assess IPoIB as a solution to run legacy socket-oriented applications over InfiniBand.

The rest of this paper is organized as follows: Related work is discussed in Section II. A survey of InfiniBand is made in Section III. The testbed for our experiments is presented in Section IV, while some benchmarking tools for point-to-point network evaluation are presented in Section V. The results of our network performance evaluation is presented and discussed in Section VI. Finally in Section VII, we provide concluding remarks and directions for future work in this area.

II. Related Work

A lot of work has been done in the field of network performance evaluation at the level of Ethernet. For example, Gamess and Ortiz-Zuazaga [8] evaluated the performance of point-to-point connections at the level of UDP, TCP, and MPI [9][10] (Message Passing Interface) on a HPC cluster connected through a Gigabit Ethernet network. In [11], the authors empirically assessed the UDP and TCP throughput, delay, and jitter when using VPNs (Virtual Private Networks), in a simple testbed composed of a router (PC with two NICs) that connected two end-nodes with FastEthernet links (100 Mbps). Narayan and Lutui [12] did an evaluation of UDP and TCP, for IPv4 and IPv6, when using jumbo frames in a controlled environment where two end-nodes were connected through a chain of two routers (PCs with two NICs), with FastEthernet links (100 Mbps). A comparison of the network performance between Windows XP, Windows Vista, and Windows 7 was conducted by Balen, Martinovic, and Hocenski [13], under IPv6 and IPv4. Their testbed consisted of two computers connected through a point-to-point link with Gigabit Ethernet.

However, up until now, just a few works have been done with InfiniBand. Cohen [14] evaluated InfiniBand in a back-to-back connection between two end-nodes, i.e. a fabric without InfiniBand switches. Latency, throughput, and CPU load were reported by the author. In [15], Rashti and Afsahi evaluated three network technologies (10-Gigabit iWARP, 4X SDR InfiniBand, and Myrinet-10G) at the user-level and MPI layer. The authors of [16] evaluated 4X FDR InfiniBand and 40GigE RoCE on HPC and cloud computing systems. They did some basic network level characterizations of performance, but most of the work is done with MPI point-to-point and collective communication benchmarks. In [17], Sur, Koop, Chai, and Panda did a network-level performance evaluation of the Mellanox ConnectX architecture on multi-core platforms. They evaluated low level operations such as RDMA Write and RDMA Read, as well as high level applications as a whole.

III. A Survey of InfiniBand

In this section, we make a survey of InfiniBand and see important concepts that can significantly help for the understanding of this research work.

A. *Basic Components of an InfiniBand Fabric*

An InfiniBand subnet is made of the following entities: (1) end-nodes, (2) links, (3) switches, (4) subnet managers or SMs, and (5) subnet manager agents. Additionally, traffic between subnets is allowed through routers. Fig. 1 shows two InfiniBand subnets (one in yellow and the other one in blue) connected by a router.

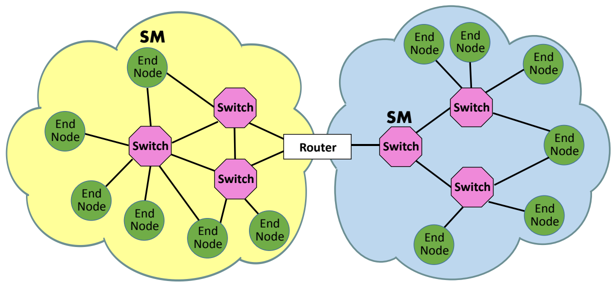

Fig.1. The InfiniBand Architecture

End-nodes are attached to links through one or more CAs (Channel Adaptors) and send messages to other end-nodes. The CA is the InfiniBand version of the NIC (Network Interface Card). A CA can have one or more IBA ports. The IBTA (InfiniBand Trade Association) defines the following MTUs: 256, 512, 1024, 2048, or 4096 bytes. Messages must be segmented into packets for transmission according to the PMTU. Segmentation of messages into packets on transmission and reassembly on receipt are provided by channel adapters at the end-nodes.

The links of an InfiniBand fabric are bidirectional point-to-point communication channels, and may be either copper or optical fiber. To achieve greater bandwidth, several links can be used in parallel or grouped together. This link association is called the "width" of the link, and common widths include: 1X, 4X, 8X, and 12X. A basic 1X copper link has four wires, consisting in two differential signaling pairs (one for sending and the other for receiving). A 4X coper link has eight pairs of wires, four in each direction. This concept can be easily generalized to 8X and 12X. The IBTA has proposed several speed grades known as: SDR (Simple Data Rate), DDR (Double Data Rate), QDR (Quadruple Data Rate), FDR (Fourteen Data Rate), and EDR (Enhanced Data Rate). SDR, DDR, and QDR use 8B/10B encoding, i.e., 10 bits carry 8 bits of data. In other words, the data rate in 80% of the signal rate. FDR and EDR use the more efficient 64B/66B encoding. Table 1 shows the signal rate and data rate achieved by InfiniBand, depending on the width of the link. The non-shaded rows represent the signal rate, while the shaded rows correspond to the data rate.

Table 1. Signal and Data Rates Achieved by InfiniBand in Gbps

	SDR	**DDR**	**QDR**	**FDR**	**EDR**
1X	2.50	5.00	10.00	14.0625	25.78125
	2.00	4.00	8.00	13.64	25.00
4X	10.00	20.00	40.00	56.25	103.125
	8.00	16.00	32.00	54.54	100.00
8X	20.00	40.00	80.00	112.50	206.25
	16.00	32.00	64.00	109.09	200.00
12X	30.00	60.00	120.00	168.75	309.375
	24.00	48.00	96.00	163.63	300.00

Switches relay packets received from a port attached to one link in the subnet through a port attached to another link in the same subnet (see Fig. 1). That is, the packets stay within the subnet. To do so, the switch selects an entry in its forwarding table that corresponds to the DLID (Destination Local IDentifier) of the packet. This entry will indicate the switch port through which the packet is to be output. The switches get their forwarding table from the Subnet Manager (SM), during the initialization of the subnet, or when a modification occurs in the subnet.

The Subnet Manager (SM) is the most critical piece of software in an InfiniBand subnet, and can run on a switch or a end-node. In Fig. 1, the SM of the yellow subnet is running on an end-node, while the SM of the blue subnet is hosted on a switch. SMs have several objectives that include: (1) the discovery of the subnet topology, (2) the assignation of LIDs (Local IDentifiers) to all devices in the subnet and the creation of forwarding tables for them, (3) the reception and processing of traps from Subnet Manager Agents (SMAs), and (4) the monitoring of the subnet for the discovering and management of changes such as adding or removing end-nodes. After an initial discovery and activation of the subnet, the subnet manager periodically scans and updates the information when needed. A subnet can have multiple instances of SMs for high availability and redundancy reasons, but only one is master, while all others remain on standby. Standby SMs do not manage the subnet, rather they periodically poll the master SM (via SubnGet) to determine if it is still actively managing the subnet or has failed. When the master SM fails, a standby SM takes over without interrupting network service.

All devices in the subnet must run a dedicated SMA (Subnet Management Agent) not shown in Fig. 1. During the initial startup or the reconfiguration of InfiniBand devices, the SM contacts the SMAs, via SMPs (Subnet Manager Packets), to get and set configuration parameters.

Routers are not currently in wide deployment, and are intended to be used to divide a very large network into smaller subnets connected together by routers. Routers route packets received on a port attached to a link in one subnet through a port attached to a link within another subnet. When doing so, routers have to change the LRH (Local Routing Header) of the packets. Routers have a routing table, and route packets according to the DGID (Destination Global IDentifier) in the GRH (Global Routing Header).

B. Addresses in InfiniBand

In InfiniBand, three types of address are defined: LIDs, GUIDs, and GIDs.

LIDs (Local IDentifiers) are local scope addresses assigned by the SM to devices in the subnet at startup time. LID addresses are 16-bit long and are present in the LRH (Local Routing Header) of packets as DLID (Destination LID) and SLID (Source LID). The DLIDs are used by switches for forwarding in the subnet, i.e., they act as Layer-2 addresses in the OSI model. When a subnet is reconfigured, new LIDs are assigned to the various devices within the subnet. LID addresses must be unique within a subnet and are divided in four groups as follows:

- LID address 0000h is reserved and must never be used.
- LID addresses 0001h-BFFFh are for unicasting. A packet with a unicast DLID (Destination LID) is always delivered to a single target port, the port with that LID address in the subnet.
- LID address C000h-FFFEh are for multicasting. The idea is to deliver a packet with a multicast DLID (Destination LID) to all the ports that are member of that multicast group.
- LID address FFFFh is also known as the PLID (Permissive LID) and must be accepted by any destination. At startup time, before the LID assignment by the SM, it is used as the DLID in LRH for communications.

According to the IBTA specification, a CA can have as few as one or as many as 255 ports. They are numbered starting at 1. Each CA port that is connected to the subnet must be assigned at least one LID address by the SM, or a range of contiguous LID addresses if the LMC (LID Mask Control) is used. To facilitate multipath routing, a base LID and a LMC value shall be assigned by the SM to each end-port. The LMC is a 3-bit field which represents 2^{LMC} paths (a maximum of 128 paths). The base LID must have the LMC least significant bits set to 0. That is, if LMC=0, the base LID may be any unicast LID. If LMC=1, the base LID may be any unicast LID multiple of 2. If LMC=2, the base LID may be any unicast LID multiple of 4, and so on. Switches can have as many as 255 ports, numbered starting at 1. Also, switches must implement port 0, the management port through which the SM can administer the switch. It is typically a virtual port with no physical attachment to a link. The SM must assign a unique LID address to port 0, i.e., a range of LID addresses is not allowed (LMC=0). Also, it is worth mentioning that the SM never assigns LID addresses to physical ports of a switch.

GUIDs (Global Unique IDentifiers) are global scope 64-bit addresses defined in [18] as IEEE EUI-64 (64-bit Extended Unique Identifier). It is the concatenation of a 24-bit OUI (Organizationally Unique Identifier) value assigned by the IEEE Registration Authority, and a 40-bit extension identifier assigned by the organization with that OUI assignment. Manufacturers assign GUIDs to chassis, CAs, CA ports, switches, and router ports. The SM can assign additional local scope GUIDs to ports on a router.

GIDs (Global IDentifiers) are 128-bit addresses used to identify end-node ports, router ports, and multicast groups. They are constructed by concatenating a 64-bit GID prefix with a EUI-64, as shown in Fig. 2. They are similar to IPv6 addresses with some restrictions. GID addresses are present in the GRH (Global Routing Header) of packets as SGID (Source GID) and DGID (Destination GID). The DGID is used by routers for routing purpose.

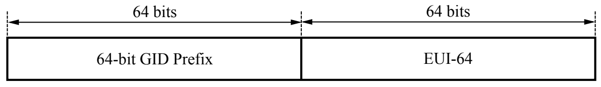

Fig.2. GID Format

C. Notion of Queue Pairs

On every port of a CA, a number of bi-directional message transport engines can be created, each of which is referred to as a QP (Queue Pair). A QP is a pair of queues: the SQ (Send Queue) and the RQ (Receive Queue):

- On the SQ, message transfer requests are posted. For the execution of each one, the QP SQ logic transmits an outbound message to a remote QP RQ.

- On the RQ, WRs (Work Requests) are posted so that the QP RQ logic can handle the inbound messages transmitted by the remote QP SQ logic.

D. Data Integrity in InfiniBand

At the link level, there are two CRCs, the variant CRC (VCRC) and the invariant CRC (ICRC), that ensure data integrity. The 16-bit VCRC includes all fields in the packet and is recalculated at each hop. The 32-bit ICRC covers only the fields that do not change from hop to hop. The VCRC provides link-level data integrity for each hop while the ICRC provides end-to-end data integrity.

E. The IBA Transport Services

IBA offers several transport services that include: RC (Reliable Connection), UC (Unreliable Connection), RD (Reliable Datagram), and UD (Unreliable Datagram).

In the RC transport service, a private connection must be established between the two RC QPs in the two CAs. The communication is private, i.e., the two RC QPs can send messages to each other, but not to any other QP. As a consequence, a RC packet has a “Destination Queue Pair” field but no “Source Queue Pair” field. The message size can be anything between 0 B and 2 GB, and messages larger than the PMTU are segmented into multi-packet transfers. The Ack/Nak mechanism permits the requester logic (QP SQ) to verify that all the packets are delivered to the responder (QP RQ). Each request packet contains a sequential PSN (Packet Sequence Number) that the responder logic (QP RQ) uses to verify that all request packets are received in order, and are only processed once. The operations that must be supported by the RC service include: RDMA Read, RDMA Write, and Send.

In the UC transport service, a private connection must be established between the two UC QPs in the two CAs. The communication is private, i.e., the two UC QPs can send messages to each other, but not to any other QP. As a consequence, a UC packet has a “Destination Queue Pair” field but no “Source Queue Pair” field. The message size can be anything between 0 B and 2 GB, and messages larger than the PMTU are segmented into multi-packet transfers. There is no Ack/Nak mechanism, hence the requester logic (QP SQ) cannot verify that the packets are delivered to the responder (QP RQ). Each request packet contains a sequential PSN that the responder logic (QP RQ) uses to verify that all request packets are received in order, and are only processed once. However, if the responder logic (QP RQ) detects out-of-order request packets, it will ignore the remaining request packets of the current message and await for the beginning of a new request packet with a BTH:Opcode (Base Transport Header) of the “First” or “Only” type. RDMA Read is not supported. However, RDMA Write and Send must be supported by the UC service.

In the UD transport service, there is no initial connection setup with the remote QP prior to sending or receiving messages. Hence, there is no private connection and the “Source Queue Pair” and “Destination Queue Pair” fields must be specified in all the requests posted to the local QP SQ. Similarly to RC and UC, the “Destination Queue Pair” is within the BTH header. However, UD has an additional header called DETH (Datagram Extended Transport Header) where the “Source Queue Pair” will be placed, since it is not a connection oriented transport service. The message size is limited to the PMTU and must be accommodated in a single packet, i.e., in UD transport service, there is no notion of multi-packet transfers. There is no Ack/Nak mechanism, hence the requester logic (QP SQ) cannot verify that the packets are delivered to the responder (QP RQ). Even though each packet contains a sequential PSN, it is not meaningful because the entire message is encapsulated in a single packet. RDMA Read and RDMA Write are not supported. The UD transport service only supports the Send message transfer operation.

F. Internet Protocol over InfiniBand

InfiniBand provides “verbs” to do low level IOs, but till date, very few applications have been developed with them. Hence, a mechanism is required to run TCP/IP on top of InfiniBand. The role of IPoIB [19][20][21] (IP over InfiniBand) is to provide an IP network emulation layer on top of InfiniBand networks, allowing the numerous existing socket-based applications to run over InfiniBand networks unmodified. As a drawback, the performance of those applications will be considerably lower than if they were directly written to use RDMA communications natively, since they do not benefit from typical features of InfiniBand (kernel bypass, reliability, zero copy, splitting and assembly of messages to packets in the CAs, etc). Linux has a module, called “ib_ipoib”, which implements IPoIB. This module creates a virtual NIC (ib0, ib1, ib2, etc) for each InfiniBand port on the system, which makes an HCA (Host Channel Adapter) act like an ordinary NIC.

IPoIB has two modes of operation: datagram mode [20] and connected mode [21]. In datagram mode the UD transport service is used, while the connected mode is based on the RC transport service. By default, IPoIB on Linux is configured in datagram mode. However, it is easy to switch between modes using the simple commands of Fig. 3. Line 01 shall be used to switch to connected mode, while Line 02 can be entered to switch to datagram mode.

```
01:  echo  connected  >  /sys/class/net/ib0/mode
02:  echo  datagram   >  /sys/class/net/ib0/mode
```

Fig.3. Switching Between Datagram and Connected Modes in IPoIB

G. OpenFabrics Enterprise Distribution

The OpenFabrics Alliance aims to develop open-source software that supports the three major RDMA fabric technologies: InfiniBand, RoCE (RDMA over Converged Ethernet), and iWARP (internet Wide Area RDMA Protocol). The OFED stack includes software drivers, core kernel-code, middleware, and low level benchmarking tools. It offers a range of standard protocols (e.g. IPoIB and MPI), and supports many file systems (e.g. Lustre and NFS over RDMA). Its first version was released in 2005.

IV. Testbed for our Experiments

For our experiments, the testbed was based on a four-node cluster with CentOS v6.6. As shown in Fig. 4, the cluster was made of four end-nodes, one InfiniBand switch (SW1), and one Gigabit Ethernet switch (SW2). The InfiniBand switch was a Mellanox Technologies SX6012, with 12 QSFP ports that support full-duplex signal rate of 56 Gbps (FDR). It is a managed switch that can be administered through the CLI (Command Line Interface) and SNMP (Simple Network Management Protocol), and also offers IPMI support. It was running Mellanox MLNX-OS version 3.4.2008 as operating system. The Gigabit Ethernet switch was a Netgear GS724T, with 24 10/100/1000 BASE-T Gigabit Ethernet ports.

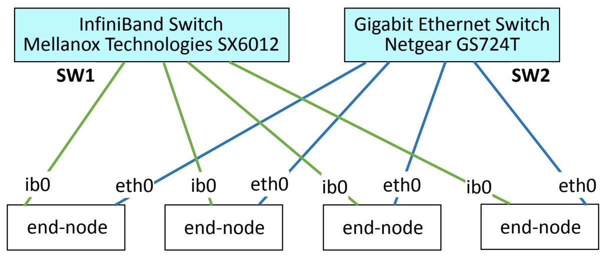

Fig.4. Testbed for our Experiments

Through SW1, we connected the end-nodes together, using InfiniBand 4X FDR links for high performance (green links in Fig. 4). Through SW2, we connected the end-nodes for administration and monitoring purposes using Gigabit Ethernet links (blue links in Fig. 4). The end-nodes had the following characteristics:

- Processors: 2 16-core Intel Xeon E5-2630 v3 at 2.4 GHz
- RAM: 64 GiB – 4 x 16 GiB DIMM (DDR4 2133 MHz)
- HCA: Mellanox Technologies single-port MT27500 ConnectX-3
- NIC: Dual-port Intel I350 Gigabit Ethernet
- Hard Disk: Seagate ST1000NM0033 (1 TB, 7200 RPM, 128 MB cache, SATA 6.0 Gb/s) for a local installation of the operating system (CentOS v6.6).
- Remote Management: IPMI.

V. Benchmarking Tools Used in our Experiments

InfiniBand is relatively new, hence just few applications have been written using the native verbs. For performance evaluation, OFED offers several micro benchmarks based on the client/server model and known as "perftest" (ib_read_bw, ib_read_lat, ib_write_bw, ib_write_lat, ib_send_bw, and ib_send_lat). ib_read_bw and ib_read_lat are oriented to the evaluation of RDMA Read transactions over InfiniBand, and report the bandwidth and latency, respectively. ib_write_bw and ib_write_lat are similar to the previous tools, but for RDMA Write transactions. ip_send_bw calculates the bandwidth of Send transactions between a pair of end-nodes. The client floods the server with Send messages and they both calculate the throughput of the operation. The test supports several transport services (RC, UC, and UD), bidirectional flows on which they both send and receive at the same time, change of the MTU, the number of iterations, the message size, and much more. ip_send_lat measures the latency of sending a packet with a specified quantity of payload bytes between a pair of end-nodes. They perform a ping-pong benchmark on which a node resends a packet only when it receives it. Each of the end-nodes samples the CPU each time they receive a packet in order to calculate the latency.

Qperf is another benchmarking tool based on the client/server model distributed as part of OFED. Qperf measures bandwidth and latency between two end-nodes. It can work over TCP/IP (socket-oriented) as well as the RDMA transports (verbs-oriented). On TCP/IP, tests include bandwidth and latency for UDP, TCP, and SCTP. On RDMA transports, available tests include bandwidth and latency for RDMA Read and RDMA Write. Moreover, tests are also available for bandwidth (unidirectional and bidirectional) and latency of Send transactions, using RC, UC, or UD as the transport services.

Many socket-based benchmarking tools have been proposed for network performance evaluation at the level of UDP and TCP. Some of the popular open source projects include Netperf and Hpcbench [22][23]. Netperf is a benchmarking tool that can be used to measure various aspects of networking performance. Its primary focus is on bulk data transfer (TCP_STREAM, UDP_STREAM, etc) and request/response performance (TCP_RR and UDP_RR) using either TCP or UDP. It is designed around the basic client/server model. In the TCP_STREAM test, a constant bitrate of data is transferred from the client (netperf) to the server (netserver), and the actual throughput is reported as the result. It is worth mentioning that the reported throughput is equal to the maximum throughput, since Netperf saturates the communication link. The UDP_STREAM test is similar to the TCP_STREAM test, except that UDP is used as the transport protocol rather than TCP. In the TCP_RR test, a fixed quantity of data is exchanged between the client (netperf) and the server (netserver) a number of times, and the benchmark reports the transaction rate which is the number of complete round-trip transactions per second. The UDP_RR is very much the same as the TCP_RR test, except that UDP is used rather than TCP.

Hpcbench [22][23] is a network benchmarking tool focused on HPC environments. It is comprised of three independent sets of benchmarks measuring UDP, TCP, and MPI communications. Similar to many other tools, the UDP and TCP tests are based on the client/server paradigm. Basically, Hpcbench allows users to evaluate the maximum throughput and the RTT, in any of the three communication standards, for a message with a length that users must specify. For throughput measurements, Hpcbench floods the network with messages of the accorded size, and reports the observed throughput, which also corresponds to the maximum throughput for this message size. For the RTT measurements, the two processes of Hpcbench bounce a message of the specified size, and report the average RTT by dividing the total time for the experiment, by the number of transactions. For throughput tests, unidirectional and bidirectional flows are allowed, however the RTT evaluation is limited to unidirectional flows.

VI. Results and Analytical Comparision

A. *Evaluation of the Latency of Send Transactions*

In this section, we run the ib_send_lat benchmark between two end-nodes using different transport layers (RC, UC, and UD), while varying the payload sizes and signal rates (SDR, DDR, QDR, and FDR). We choose an MTU of 4,096 bytes. Table 2 shows the results for the RC transport service. For small payload sizes, the latency is almost the same for the four signal rates and around 0.85 us. As the payload size is incremented, we can see that the smallest latency is achieved by FDR, while SDR has the biggest one. The differences are significant with big payload sizes, since the transmission time is getting more important that the propagation time in these cases.

Table 2. Latency in Microseconds for the RC Transport Service

Size (Bytes)	SDR	DDR	QDR	FDR
4	0.92	0.86	0.82	0.82
8	0.93	0.86	0.83	0.83
16	0.95	0.87	0.83	0.83
32	0.97	0.89	0.88	0.87
64	1.04	0.95	0.93	0.89
128	1.16	1.05	1.03	0.97
256	1.84	1.69	1.64	1.58
512	2.20	1.92	1.78	1.74
1,024	2.89	2.34	2.04	2.03
2,048	4.22	3.16	2.61	2.50
4,096	6.34	4.22	3.15	2.84
8,192	10.53	6.31	4.19	3.48
16,384	18.85	10.49	6.29	4.78
32,768	35.48	18.81	10.69	7.73
65,536	68.72	35.43	19.05	12.88
131,072	135.21	68.68	35.67	23.22
262,144	268.17	135.16	68.91	43.87
524,288	534.09	268.12	135.39	85.16
1,048,576	1,065.95	534.08	268.36	167.75
2,097,152	2,129.71	1,065.97	534.30	333.08
4,194,304	4,257.21	2,129.73	1,066.27	664.02
8,388,608	8,512.54	4,257.59	2,130.49	1,428.55

The results for the latency in the case of the UC transport service are shown in Table 3. There are very similar to the results of Table 2.

In the case of the UD transport service, the results are presented in Table 4. It is worth remembering that the payload size of an UD message is limited to the PMTU, in our case 4,096 bytes. Hence, in this experiment, we vary the payload sizes up to 4,096 bytes. It is also noticeable that the latency for UD is a little over the latency for RC and UC. This is due to the additional header of UD packets, i.e. DETH is 8-byte long, and it is an additional header presents in all the UD packets.

Table 3. Latency in Microseconds for the UC Transport Service

Size (Bytes)	SDR	DDR	QDR	FDR
4	0.91	0.86	0.82	0.82
8	0.92	0.86	0.84	0.83
16	0.93	0.88	0.87	0.83
32	0.96	0.88	0.87	0.86
64	1.03	0.94	0.91	0.89
128	1.15	1.04	1.02	0.96
256	1.84	1.66	1.61	1.50
512	2.23	1.92	1.78	1.75
1,024	2.92	2.33	2.09	2.04
2,048	4.24	3.15	2.61	2.50
4,096	6.36	4.21	3.16	2.84
8,192	10.56	6.31	4.20	3.49
16,384	18.87	10.50	6.28	4.77
32,768	35.49	18.82	10.47	7.54
65,536	68.75	35.44	18.79	12.83
131,072	135.25	68.69	35.41	23.17
262,144	268.21	135.16	68.66	43.83
524,288	534.13	268.13	135.15	85.16
1,048,576	1,065.96	534.08	268.87	167.78
2,097,152	2,129.67	1,065.94	534.85	333.15
4,194,304	4,257.02	2,129.60	1,066.69	664.35
8,388,608	8,512.00	4,257.34	2,130.79	1,429.77

Table 4. Latency in Microseconds for the UD Transport Service

Size (Bytes)	SDR	DDR	QDR	FDR
4	0.96	0.89	0.85	0.84
8	0.97	0.89	0.85	0.84
16	0.98	0.90	0.86	0.85
32	1.00	0.92	0.87	0.85
64	1.06	0.96	0.94	0.92
128	1.19	1.05	1.03	0.99
256	1.88	1.67	1.64	1.59
512	2.23	1.89	1.77	1.75
1,024	2.90	2.31	2.04	2.00
2,048	4.23	3.13	2.61	2.47
4,096	6.92	4.78	3.74	3.39

B. *Evaluation of the Throughput of Send Transactions*

In this section, we run the ib_send_bw benchmark between two end-nodes using different RDMA transport layers (RC, UC, and UD), while varying the payload sizes. We choose a signal rate of FDR and an MTU of 4,096 bytes. Fig. 5 shows the results for unidirectional communications. Since UD is limited to a payload size equals to the PMTU, it has values up to a payload size of 4,096 bytes. The best performance is achieved by UC, while RC has the poorest one due to the overhead caused by the Ack/Nak mechanism.

Fig. 6 depicts the results for bidirectional communications. They are very similar to the results obtained for the unidirectional case (Fig. 5). However, we can observe a degradation of the service in the case of bidirectional communications for messages with a payload size greater than or equal to 8,388,608 bytes. Also it is worth pointing out that the results reported by Fig. 6 is the throughput in one direction, i.e. the total throughput is twice the one reported.

In Fig. 7 (unidirectional) and Fig. 8 (bidirectional), we study the effect of the signal rate over the throughput of the RC transport service, when varying the payload size of the messages. To do so, we run the ib_send_bw benchmark between two end-nodes of our testbed. For any value of the payload size, we have four bars that correspond to SDR, DDR, QDR, and FDR, respectively. For each signal rate of the unidirectional case (Fig. 7), the throughput increases and reaches a limit close to the corresponding data rate. The bidirectional case (Fig. 8) is similar to the unidirectional one, however the performance shows a degradation for messages with a payload size greater than or equal to 8,388,608 bytes, in the case of FDR.

We study the effect of the signal rate over the throughput of the UC transport service in Fig. 9 (unidirectional) and Fig. 10 (bidirectional), when varying the payload size of the messages. To do so, we run the ib_send_bw benchmark between two end-nodes of our testbed. For any value of the payload size, we have four bars that correspond to SDR, DDR, QDR, and FDR, respectively. For each signal rate of the unidirectional case (Fig. 9), the throughput increases and reaches a limit close to the corresponding data rate. The bidirectional case (Fig. 10) is similar to the unidirectional one, however the performance shows a degradation for messages with a payload size greater than or equal to 8,388,608 bytes, in the case of FDR.

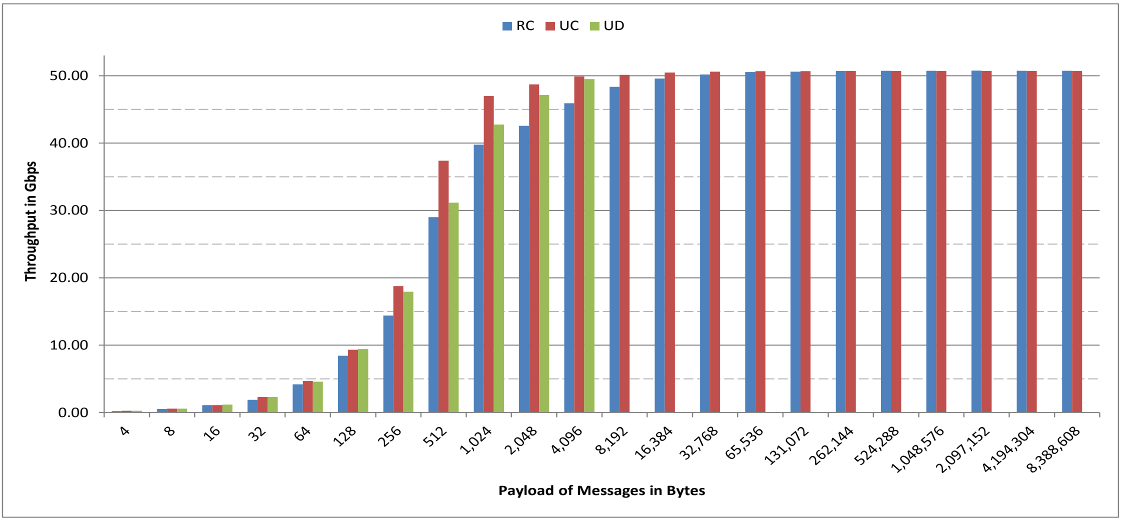

Fig.5. Throughput for Unidirectional Communication for Send Transactions with RC, UC, and UD

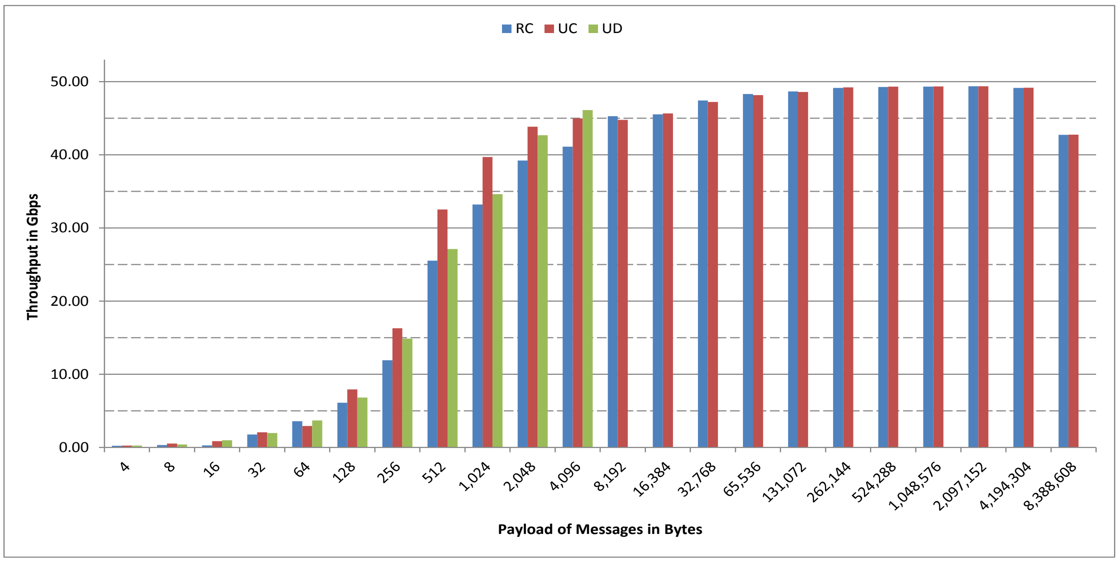

Fig.6. Throughput for Bidirectional Communication for Send Transactions with RC, UC, and UD

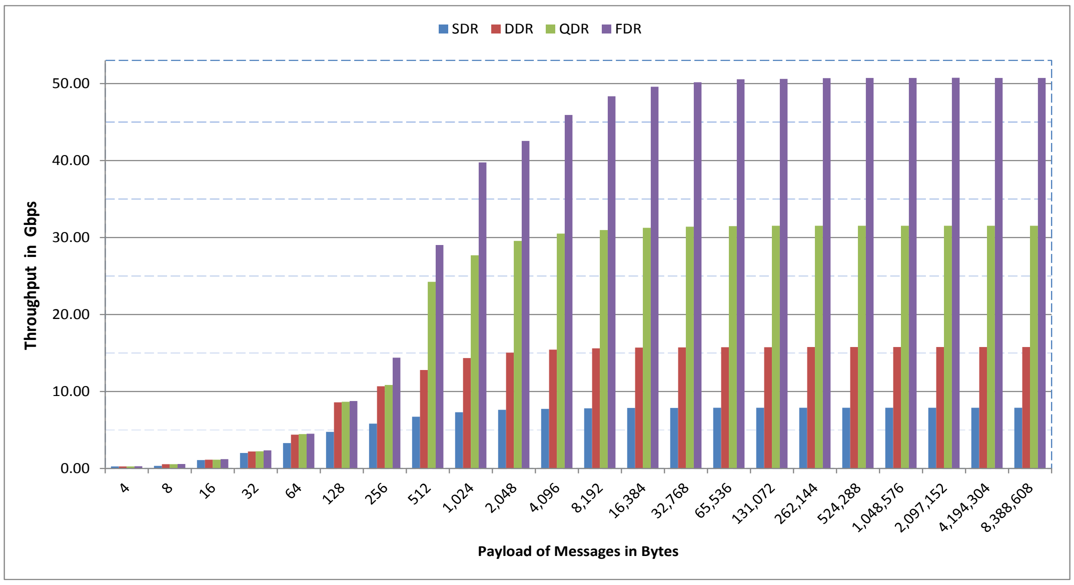

Fig.7. Throughput for Unidirectional Communication for Send Transactions with RC

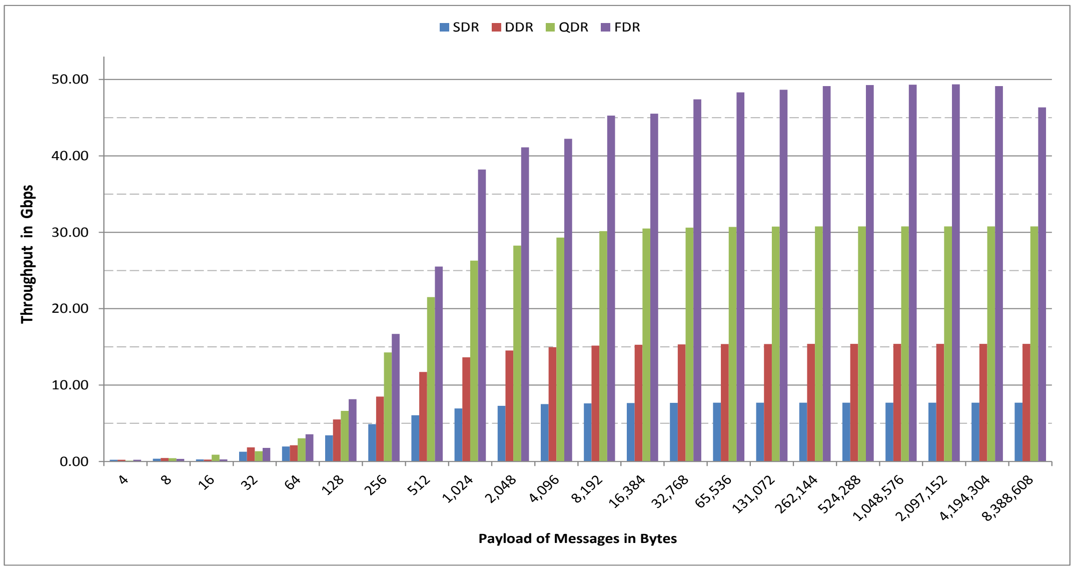

Fig.8. Throughput for Bidirectional Communication for Send Transactions with RC

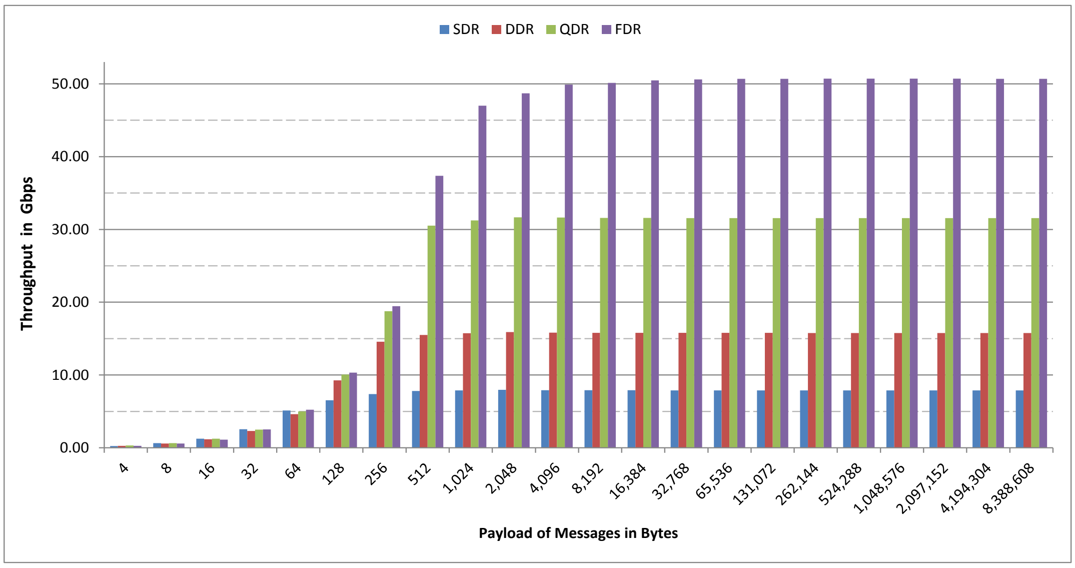

Fig.9. Throughput for Unidirectional Communication for Send Transactions with UC

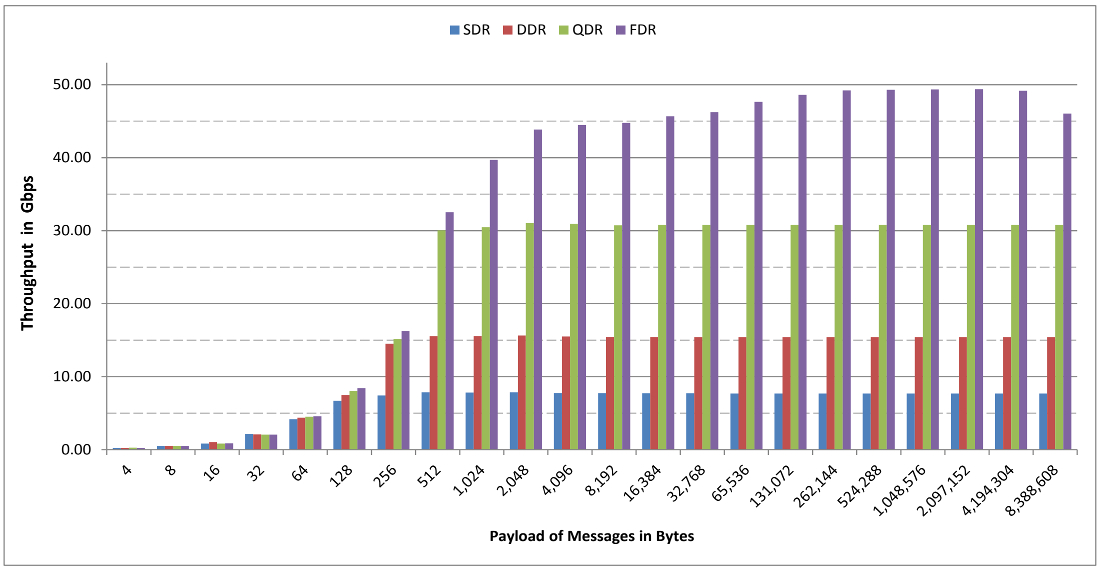

Fig.10. Throughput for Bidirectional Communication for Send Transactions with UC

C. Evaluation of IPoIB in Datagram Mode

Since there are many network applications that have been developed with sockets, it is important to evaluate the performance of UDP and TCP when using IPoIB on InfiniBand. Table 5 shows the results of the UDP throughput with IPoIB in datagram mode, when using 4X FDR links, with IPv4. We repeated the tests with different benchmarking tools (Netperf, Hpcbench, and Qperf) to investigate the consistency of the results. We did not test with a bigger payload size since it is limited by the size of an IPv4 packet. As we can see, the results for the three benchmarking tools are quite consistent. Also it is worth to mention that the throughput observed for a payload size of 32,768 bytes for UDP is around 10 Gbps, with is by far below the one obtained by Send transactions (50.18 Gbps for RC and 50.61 Gbps for UC).

Table 5. Throughput in Mbps for UDP with IPoIB in Datagram Mode

Size (Bytes)	Netperf	Hpcbench	Qperf
4	17.06	NA	16.60
8	34.09	NA	33.62
16	66.70	NA	65.17
32	135.99	131.44	131.45
64	270.03	265.28	265.18
128	550.17	526.72	514.73
256	1,081.73	1,067.11	1,054.22
512	2,177.04	2,107.92	2,053.11
1,024	3,192.45	3,109.18	3,107.29
2,048	3,902.52	3,894.22	3,851.52
4,096	4,613.65	4,592.14	4,583.88
8,192	6,257.15	6,238.63	6,352.86
16,384	7,880.56	7,839.85	7,925.45
32,768	9,247.56	10,446.27	9,721.09

Table 6 shows the results of the UDP and TCP latency with IPoIB in datagram mode, when using 4X FDR links, with IPv4. We repeated the tests with different benchmarking tools (Hpcbench and Qperf) to investigate the consistency of the results. We did not test with a bigger payload size since it is limited by the size of an IPv4 packet. As we can see, the results for the two benchmarking tools are quite consistent. We did not use Netperf since it does not report latency. Also it is worth to notice that the latency observed for a payload size of 32,768 bytes with UDP and TCP is around 35.0 us, with is by far above the one obtained by Send transactions (7.73 us for RC and 7.54 us for UC).

Table 6. Latency in Microseconds for UDP and TCP with IPoIB in Datagram Mode

Size (Bytes)	UDP		TCP	
	Hpcbench	Qperf	Hpcbench	Qperf
4	7.37	7.37	8.42	8.40
8	7.41	7.42	8.51	8.49
16	7.44	7.47	8.72	8.71
32	7.46	7.53	8.83	8.82
64	7.48	7.55	9.00	8.97
128	7.50	7.59	9.32	9.15
256	7.61	7.62	9.71	9.60
512	7.74	7.73	10.20	10.12
1,024	7.81	7.82	10.35	10.52
2,048	11.26	11.12	11.55	11.63
4,096	14.86	14.32	14.52	14.92
8,192	18.44	18.28	18.93	18.58
16,384	23.34	22.51	23.51	23.41
32,768	35.71	33.17	35.02	34.51

D. Evaluation of IPoIB in Connected Mode

Table 7 shows the results of the UDP throughput with IPoIB in connected mode, when using 4X FDR links, with IPv4. We repeated the tests with different benchmarking tools (Netperf, Hpcbench, and Qperf) to investigate the consistency of the results. We did not test with a bigger payload size since it is limited by the size of an IPv4 packet. As we can see, the results for the three benchmarking tools are quite consistent. We can see that the throughput is higher with IPoIB in datagram mode (Table 5) than with IPoIB in connected mode (Table 7). Also it is worth to mention that the throughput observed for a payload size of 32,768 bytes for UDP is around 7.7 Gbps, with is by far below the one obtained by Send transactions (50.18 Gbps for RC and 50.61 Gbps for UC).

Table 7. Throughput in Mbps for UDP with IPoIB in Connected Mode

Size (Bytes)	Netperf	Hpcbench	Qperf
4	11.76	NA	11.09
8	21.22	NA	20.40
16	44.47	NA	43.62
32	87.76	89.25	88.81
64	175.95	186.53	172.40
128	355.47	359.34	337.43
256	430.78	437.05	429.57
512	728.87	715.14	723.40
1,024	1,147.29	1,153.02	1,149.81
2,048	1,383.77	1,440.21	1,392.26
4,096	1,509.61	1,512.89	1,479.82
8,192	3,341.35	3,393.58	3,422.57
16,384	7,649.47	7,614.32	7,438.21
32,768	7,782.21	7,785.15	7,653.52

Table 8. Latency in Microseconds for UDP and TCP with IPoIB in Connected Mode

Size (Bytes)	UDP		TCP	
	Hpcbench	Qperf	Hpcbench	Qperf
4	7.39	7.38	8.43	8.42
8	7.43	7.41	8.52	8.50
16	7.48	7.48	8.72	8.72
32	7.52	7.56	8.85	8.84
64	7.55	7.59	9.05	8.99
128	7.61	7.62	9.40	9.19
256	7.64	7.64	9.82	9.74
512	7.76	7.75	10.38	10.42
1,024	7.87	7.88	10.59	10.68
2,048	11.28	11.30	11.92	11.72
4,096	15.97	15.01	16.45	15.31
8,192	18.52	18.35	22.63	21.58
16,384	31.57	31.41	34.58	31.65
32,768	45.83	45.17	55.25	53.13

Table 8 shows the results of the UDP and TCP latency with IPoIB in connected mode, when using 4X FDR links, with IPv4. We repeated the tests with different benchmarking tools (Hpcbench and Qperf) to investigate the consistency of the results. We did not test with a bigger payload size since it is limited by the size of an IPv4 packet. As we can see, the results for the two benchmarking tools are quite consistent. We did not use Netperf since it does not report latency. We can see that the latency is lower with IPoIB in datagram mode (Table 6) than with IPoIB in connected mode (Table 8). Also it is worth to notice that the latency observed for a payload size of 32,768 bytes with UDP and TCP is around 45.0 us and 55.0 us, respectively, with is by far above the one obtained by Send transactions (7.73 us for RC and 7.54 us for UC).

VII. Conclusions and Future Work

In this paper, we did an introduction to the InfiniBand architecture, an emerging network technology that has been widely accepted in datacenters and HPC clusters. We made a performance evaluation of InfiniBand with low level benchmarking tools in a controlled environment, for different transport services (RC, UC, and UD), when varying the payload length and the signal rate (SDR, DDR, QDR, and FDR). The results obtained in our tests showed very high throughput and low latency and encourage the usage of this technology. However, InfiniBand is based on “verbs” and applications must be developed with RDMA support in order to take advantage of this fast network. In the meanwhile, the IETF has proposed a solution, called IPoIB, in order to run socket-based applications over InfiniBand. It is a tradeoff between running socket-based applications over InfiniBand without any changes and loosing part of the high performance of InfiniBand. Hence, we also assessed the performance of IPoIB in datagram and connected modes. Our experiments showed significant differences between native verbs and IPoIB. Also, it is worth pointing out that the datagram mode of IPoIB outperforms the connected mode.

As future work, we plan to make a performance evaluation of IPv4 and IPv6 [24][25] when using IPoIB. Another direction that we plan to explore is the development of mathematical models to represent the maximum throughput and the minimum latency that can be achieved by different transport services of InfiniBand.

Acknowledgment

We want to thank the NSF (National Science Foundation) which partially supported this research under grant number 1010094 through the EPSCoR Track 2 program. We also express our gratitude for all the valuable assistance and comments that we received throughout this project from José Bonilla and Ramón Sierra of the High Performance Computer Facility of the University of Puerto Rico, and José Muñoz and Osvaldo Casiano of the Resource Center for Science and Engineering of the University of Puerto Rico.

References

[1] InfiniBand Trade Association, InfiniBand Architecture Specification Volume 1, Release 1.3, March 2015.
[2] InfiniBand Trade Association, InfiniBand Architecture Specification Volume 2, Release 1.3, March 2015.
[3] MindShare and T. Shanley, InfiniBand Network

Architecture, 1st edition, Addison Wesley, November 2002.
[4] InfiniBand Trade Association, Supplement to InfiniBand Architecture Specification Volume 1, Release 1.2.1, RDMA over Converged Ethernet (RoCE), Annex A16, April 2010.
[5] InfiniBand Trade Association, Supplement to InfiniBand Architecture Specification Volume 1, Release 1.2.1, RoCEv2, Annex A17, September 2014.
[6] Mellanox, RoCE vs. iWARP Competitive Analysis, White Paper, August 2015.
[7] R. Recio, B. Metzler, P. Culley, J. Hilland, and D. Garcia, A Remote Direct Memory Access Protocol Specification, RFC 5040, October 2007.
[8] E. Gamess and Humberto Ortiz-Zuazaga, Evaluation of Point-to-Point Network Performance of HPC Clusters at the Level of UDP, TCP, and MPI, IV Simposio Cient ífico y Tecnol ógico en Computaci ón, Caracas, Venezuela, May 2016.
[9] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, Version 3.1, High Performance Computing Center Stuttgart, June 2015.
[10] W. Gropp, E. Lusk, and A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, 3rd edition, MIT Press, November 2014.
[11] S. Narayan and M. Fitzgerald, Empirical Network Performance Evaluation of Security Protocols on Operating Systems, International Journal of Wireless and Microwave Technologies, Vol. 2, No. 5, pp. 19-27, October 2012.
[12] S. Narayan and P. Lutui, TCP/IP Jumbo Frames Network Performance Evaluation on a Test-bed Infrastructure, International Journal of Wireless and Microwave Technologies, Vol. 2, No. 6, pp. 29-36, December 2012.
[13] J. Balen, G. Martinovic, and Z. Hocenski, Network Performance Evaluation of Latest Windows Operating Systems, in proceedings of the 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Vol. 7, Split, Croatia, September 2012.
[14] A. Cohen, A Performance Analysis of 4X InfiniBand Data Transfer Operations, 2003 International Parallel and Distributed Processing Symposium (IPDPS 2003), Nice, France, April 2003.
[15] M. Rashti and A. Afsahi, 10-Gigabit iWARP Ethernet: Comparative Performance Analysis with InfiniBand and Myrinet-10G, 2007 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007), Long Beach, CA, USA, March 2007.
[16] J. Vienne, J. Chen, Md. Wasi-ur-Rahman, N. Islam, H. Subramoni, and D. Panda, Performance Analysis and Evaluation of InfiniBand FDR and 40GigE RoCE on HPC and Cloud Computing Systems, Symposium on High-Performance Interconnects, Santa Clara, CA, USA, August 2012.
[17] S. Sur, M. Koop, L. Chai, and D. Panda, Performance Analysis and Evaluation of Mellanox ConnectX InfiniBand Architecture with Multi-Core Platforms, 15th Annual IEEE Symposium on High-Performance Interconnects (HOTI 2007), Stanford, CA, USA, August 2007.
[18] Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority, http://www.standards.ieee.org/regauth/oui/tutorials/EUI64.html.
[19] V. Kashyap, IP over InfiniBand (IPoIB) Architecture, RFC 4392, April 2006.
[20] J. Chu and V. Kashyap, Transmission of IP over InfiniBand (IPoIB), RFC 4391, April 2006.
[21] V. Kashyap, IP over InfiniBand: Connected Mode, RFC 4755, December 2006.
[22] B. Huang, M. Bauer, and M. Katchabaw, Network Performance in High Performance Linux Clusters, in proceedings of the 2005 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2005), Las Vegas, Nevada, USA, June 2005.
[23] B. Huang, M. Bauer, and M. Katchabaw, Hpcbench – A Linux-Based Network Benchmark for High Performance Networks, in proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS 2005), Guelph, Ontario, Canada, May 2005.
[24] J. Davies, Understanding IPv6, 3rd ed., Microsoft Press, June 2012.
[25] J. Pyles, J. Carrell, and E. Tittel, Guide to TCP/IP: IPv6 and IPv4, 5th edition, Cengage Learning, June 2016.

Authors' Profiles

Eric Gamess received a MS in Industrial Computation from the National Institute of Applied Sciences of Toulouse (INSA de Toulouse), France, in 1989, and a PhD in Computer Science from the Central University of Venezuela, Venezuela, in 2000. He is currently working as a professor at Central University of Venezuela, Venezuela. Previously, he worked as a professor at University of Puerto Rico, Puerto Rico, and "Universidad del Valle", Colombia. His research interests include Vehicular Adhoc Networks, Network Performance Evaluation, IPv6, and Network Protocol Specifications. He is a member of the Venezuelan Society of Computing and has been in the organization committee and program committee of several national and international conferences.

Humberto Ortiz-Zuazaga received a BS in Computer Science from the University of Cincinnati, USA in 1992, and a PhD in Computing Sciences and Engineering from the University of Puerto Rico at Mayaguez in 2008. He is an associate professor in the Department of Computer Science, University of Puerto Rico at Rio Piedras, and Director of the University of Puerto Rico High Performance Computing Facility. His research is primarily focused in bioinformatics of gene expression and cyberinfrastructure.

How to cite this paper: Eric Gamess, Humberto Ortiz-Zuazaga,"Low Level Performance Evaluation of InfiniBand with Benchmarking Tools", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.10, pp.12-22, 2016.DOI: 10.5815/ijcnis.2016.10.02