InfiniBand Survey

Jeremy Langston
School of Electrical and Computer Engineering
Tennessee Technological University
Cookeville, Tennessee 38505
Email: jwlangston21@tntech.edu

***Abstract*—InfiniBand is a high-speed I/O interconnect technology devised by many leading industry companies. The goal is to ultimately replace onboard local interconnects with an interface that connects internal and, primarily, external devices for shared I/O at rates above 60 Gbps. Detailed here is the framework for the InfiniBand design, following the TCP/IP communications stack scheme.**

I. Introduction

High-end computing systems are currently full of different high-speed I/O technologies that are all pulling for their part of the market share. Each of these is backed by individual or small groups of companies and provide proprietary communications and connections. This poses a big problem for the future of high-speed I/O. The InfiniBand architecture (IBA) was formed to rid the market of these proprietary concerns. In 1999, a group of over 220 companies conceded to establishing a new standard architecture [7]. IBA was led by a committee formed by Dell, Compaq, HP, IBM, Intel, Microsoft, and Sun. Due to the vast amount of companies involved, the designers wished to have plenty of variability for multiple vendors to have their niche. However, many details could not be changed and must be standardized. This standardization will ensure that the different implementations of vendors will all work together; otherwise the IBA would fall apart and revert to a proprietary nature. The specifications given to IBA are vast [4][5] with areas for vendor variance.

Current interconnect buses are quite limited. The three main buses in wide use are PCI (peripheral components interface), PCI-X, and PCI Express. PCI was introduced in the 90s as a replacement for the ISA interface. It has long been the interface of choice, but very limited on data rates. Every PC card plugged into the PCI bus shared the same connection meaning that only one device could use the bus at once. Also, the data rate was limited to 32 bit at 33 MHz, but increased to 64 bit at 66 MHz [1]. The successor to PCI was PCI-X, merely an increase in clock frequency. PCI-X is a 64 bit interface running at 133 MHz. Although the PCI-X interface gave a 1 GB/s throughput, cost was a limiting factor due to the high number of pins required [6]. Both PCI and PCI-X were pushed as far as they could. A true upgrade was in order. PCI Express, formerly known as the 3GIO bus [2], is the current local interconnect standard. The objective here was to use a small amount of pins at a very high bandwidth. While PCI Express does offer high bandwidths and scalability, it shares two major limitations with the other versions of the PCI bus architecture.

The PCI bus is intended purely for a single system, local interconnect. As such, PCI does not support external connections. In the case of a server environment where there are many I/O devices on many different servers with a high degree of interaction, PCI falls short and InfiniBand steps in. The other limitation is the way PCI devices interact with a system via memory mapped I/O and tree-structured [6]. PCI uses bridges to expand the number of devices to be connected, inducing a tree-like hierarchy. Furthermore, the PCI devices share a block of memory in which each device is mapped a single address. This interfacing strategy is particularly hard to share between systems. InfiniBand employs a logical component called a queue pair for mapping directly to memory. The memory mapping differs for each system and does not have a hierarchical structure.

This paper describes the InfiniBand architecture detailed by the different components included in a typical IBA network (Section 2) and the particular

communication layer it uses as described in Section 3. Section 4 discusses how memory is managed and the different levels it is managed at. Path migration, techniques for following different physical or logical routes, is covered in Section 5, and Section 6 concludes the paper.

II. Components

A subnet is the smallest constituent of an IBA network. It is made up of endnodes, channel adapters, cabling, switches, and, optionally for inter-subnet communication, routers and repeaters, as seen in Figure 1. Endnodes are the ultimate sources or sinks of the communication on the IBA network [3]. An endnode may be a server or workstation, or may be an I/O device such as a disk storage array. The physical interface for connecting endnodes is facilitated by channel adapters (CA). CAs are essentially direct memory access (DMA) engines for enhanced memory transfer [4]. As explained later in the transport layer section, memory accesses can be local or remote. CAs are directly and indirectly addressed using IDs. There are two types: host channel adapters (HCA) and target channel adapters (TCA). HCAs, used for servers and workstations, differ from TCAs, used for devices, by the features they have and operations they can perform. These operations are referred to as verbs, and are explained later. Due to the nature of I/O devices, they need no such operational interface and thus TCAs are less complex. Ultimately, HCAs will be implemented directly into the onboard communications bus of the system to drive down the communication delay. This is an end goal. In order to reach this goal, HCAs were designed to interface with existing buses (e.g. PCI). An increase in IBA popularity will lead to manufacturers designing system boards with onboard IBA interfaces.

Connecting endnodes, and thus CAs, are the media, switches, routers, and repeaters. Communication is bi-directional. The media used may be either copper or optical fiber and is discussed in further detail under Physical Layer, or may be printed traces on a system board [4]. Switches provide a means of patching communication links together. They are synonymous with ethernet switches and operate on the link layer. The application for a switch is limited to inter-subnet communication. Due to the packet formatting and routing limitations, switches cannot interconnect other subnets. Routers cover this limitation by working on the network layer, thus having all of the required sophistication to route data using extra packet header information. Repeaters have only a single task: to replicate the incoming/outgoing data for increased connection distances. As the distance between components grows, the signal degrades due to noise, attenuation, and other factors. Repeaters operate on the physical layer and perform no routing at all; therefore, they are transparent and non-addressable. Also, although atypical, direct connection of CAs is supported.

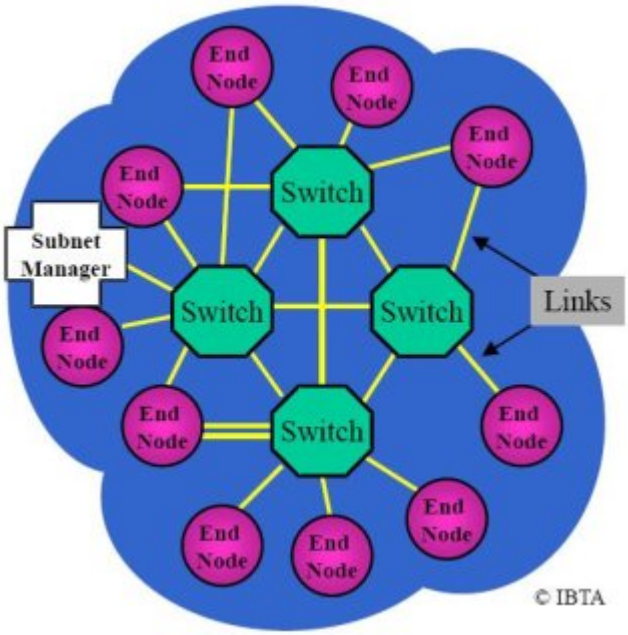

Fig. 1. A sample IBA subnet.

The subnet is formed and maintained by a subnet manager. Explained later, the subnet manager actively identifies new or removed endnodes and informs existing nodes of things such as MTU sizes, etc.

III. Communications Stack

The approach to IBA communication models that of the OSI ISO model, and more specifically the TCP/IP stack. There are five different layers that provide abstractions of data that add meaning. Such an approach allows for distinct views of the implementation, instead of a conglomerate of the entire design. More importantly, some layers may

be considered, or may be ignored depending on necessity. Switches are an example of this as they do not need to know the use of the data it transports, only that it should be sent to a specific destination. Each layer, excepting the physical, adds header information to the data to clarify the meaning of the data. Following is a description of each layer and the importance of the facilities they may apply.

IBA Link	Signal Pairs	Signaling Rate	Data Rate	Full-Duplex Data Rate
1X	2	2.5 Gbps	2.0 Gbps	4.0 Gbps
4X	8	10 Gbps	8 Gbps	16 Gbps
12X	24	30 Gpbs	24 Gpbs	48 Gbps

TABLE I
IBA LINKS [2]

A. Physical Layer

The physical layer is the lowest level in the stack. Specified here are connectors, media, form factors, bit rates, signaling techniques, and simple packet validity checks, among other functions [4]. As discussed above, both copper and optical fiber are plausible for mediums. Both require a physically separate send and receive channel. In the case of copper, a differential pair is required to reduce the suceptibility to noise. Regardless of the media used, the signaling rate, is 2.5 Gbaud. More specifically, the data rate is 2 Gbps as 8/10B encoding is used[1]. Increasing the data rate becomes a matter of increasing the amount of physical links [8]. The standard single bi-directional link is referred to as 1X, and may be increased to 4X and 12X where 4 and 12 are the amount of physical links, or lanes. As shown in Table 1, a proportionate increase in rate is realized for an increase in lanes. IBA allows for significant expandability by clocking the signaling rate higher than the single data rate (SDR). The supported clocking factors are double (DDR) and quad data rate (QDR). This increases the signaling rates to 5 Gbps and 10 Gbps respectively. Currently, only SDR and DDR connections are available. The length of the medium is limited to within 20m for copper and 200m for fiber, and the distance shortens as the data rate increases.

An advantage of the IBA design is that cut-through switching can be used [8]. When multiple links are used, they may be used in parallel, sending an entire portion of the transmission at once, and buffering is not needed for the bulk of the transmission. Once the destination address is received, no further buffering is required and is sent directly to that address. This type of forwarding has a downfall in that error checking cannot be performed. Also, cut-through switching cannot be used when transmitting between different clocking speeds; buffering must be used.

The only error checking that may be used at this layer is checking for a correctly formed packet. Bits are transmitted on the media forming symbols that frames a packet. Starting and ending symbols delimit the packet, informing the interface of an incoming or completed packet. Checking for these symbols allows for errors to be seen and handled.

B. Link Layer

There are three main aspects of the link layer: packet formatting, flow control, and intra-subnet routing [4], although a lot of things happen at this layer.

1) Packets: InfiniBand has data packets for normal I/O communication and management packets for establishing and maintaining links such as incurring flow control. A closer look at the data packet (Figure 2) shows each of the five stack layers and their corresponding headers. The link layer protocol has three separate headers it adds to the packet. The local routing header, LRH, is used for routing packets within a subnet. It denotes the source and destination local IDs (LIDs), the length of the packet payload, and identifiers for virtual lanes and service levels (explained below). LIDs are assigned to each port in a CA and each switch/router, but not each port on a switch/router. During intra-subnet communication the destination LID in the LRH is unchanged. However, the destination LID is replaced with a router's LID if the packet is to be sent to another subnet. The other two additions the

[1]The 8/10B encoding scheme provides a rudimentary synchronization technique. Using a standard, agreed-upon encoding table, each byte is converted to a 10 bit sequence that necessarily induces changes in logic levels. Such a conversion may take the binary sequence from '11000001' to '10110011', reducing the string of 0's. The drawback of this mechanism is an increased bit rate of 25%.

link layer makes are for cyclic redundancy checks, or CRCs. An invariant CRC covers the entire packet at transmission and never changes. This allows for end-to-end error checking verification. The second CRC is variant and takes into account a changing destination LID during inter-subnet communication. IBA has support for unicast (normal operation) and multicast addresses.

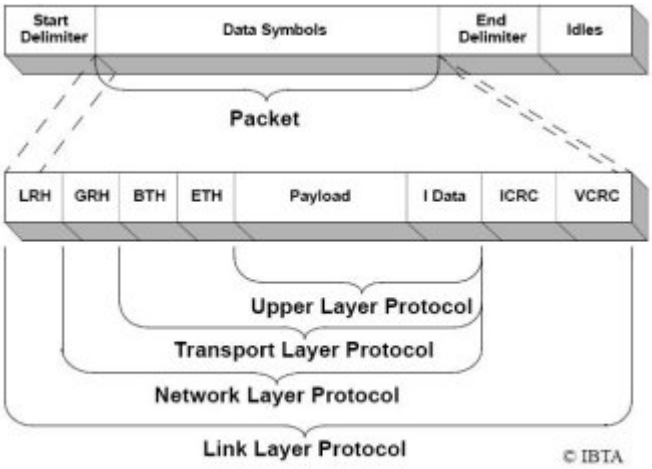

Fig. 2. InfiniBand packet structure.

2) Virtual Lanes and Service Levels: Virtual lanes, VL, are logical links for independent communications. They are used to separate traffic, avoid deadlock, and give priorities to certain links [3]. Every physical link has at least two VLs: one for data and another for management. Up to 16 VLs can be used per link, giving a maximum of 15 data lanes. As seen in Figure 3, each VL is time multiplexed over the physical link and arbitrated by a table of priorities. Flow control is implemented here on a credit basis. Each lane is given a number of credits, the number of packets that may be sent without loss, that are determined by the user. A high priority limitation is included to avoid one VL from hogging the majority of the available bandwidth (the management lane, VL15, is not subject to flow control). Also, each VL has a queue of buffers. If the queue is full, the transmitter must wait until space has been made. For unreliable communications, this technique is of no consequence. A problem arises when the packet is transmitted through a switch with a different number of VLs. In order to correctly transpose communications on different VLs, an abstraction called service levels (SLs) is used to map them properly. The SL-to-VL mappings are in all related endnodes and switches/routers.

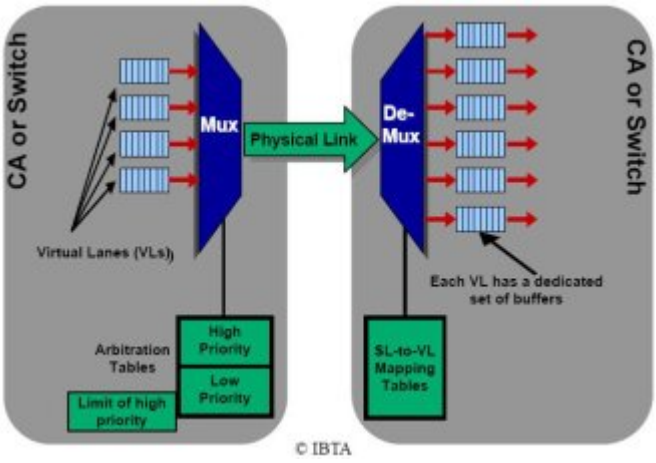

Fig. 3. Virtual Lanes.

C. Network Layer

All routing between subnets is performed at the network layer. The need for this layer is analogous to the same layer of the TCP/IP stack. Internal to the subnet, computers and devices are typically given a 192.168.x.x address adding expandibility through more addresses and security. In order for an external computer or device to communicate with one of these addresses, extra header information is required to find the particular destination node. For IBA, a global routing header, GRH, is used that is similar to IPv6 [7]. The header has many of the same fields as the LRH, but on a global level. To allow for high scalability, the IDs cover a much larger range. The global ID, GID, is formed by using the endnode's port global unique ID, GUID, and the destination subnet's ID; the GID is a valid IPv6 address. A GUID is given by IEEE and hardcoded by the manufacturer. This gives routers enough information to properly process and route the packets to the destination node. Multicasting is facilitated by changing the initial bits of the GID [3]. As mentioned previously, when inter-subnet communication is made, the destination LID changes as the subnet changes and thus the variant CRC changes.

D. Transport Layer

The next layer up is the transport layer which has many functions such as service types, packet segmentation, remote DMA (RDMA), and partitioning. Included are base transport headers (BTH) that specify the target queue pair, an operation to perform, a sequence number, and the partition. Extended transport headers (ETH) are added when using RDMA, atomic operations, and reliable datagrams [4].

1) Queue Pairs: Another abstraction is made defining the notion of a queue pair, QP. This is synonymous with TCP/IP sockets and logical ports. A QP consists of a send and receive buffer that hold the operation to be performed and any required data. As seen in Figure 4, a QP is instantiated for each transmission sent within a CA, and thus only one QP per port. The size of a QP is determined at creation and is the amount of work requests that can be handled at once. Work requests are discussed in the upper layer protocols section. Also when a QP is created, a block of memory is assigned to it. This is further discussed in Section 4.

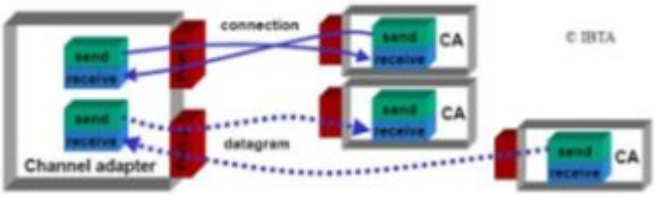

Fig. 4. Queue Pairs using connection and connectionless service types.

2) Service Types: There are four main types of communication services provided by IBA, as well as support for raw IPv6 and ethertype datagrams. They are classified by being connection or connectionless and reliable or unreliable. Reliable services basically guarantee that the packets will arrive in order, undamaged, and acknowledged. The variant and invariant CRCs are used to check for damaged packets. Unreliably sent packets are the exact opposite; no special action is taken to ensure they arrive entact. Connection-oriented service types establish a logical connection before transmitting data, whereas connectionless types send data as soon as they are able without forming a logical connection for the entirety of the transmission. That being said, the four main types of services are reliable connection (RC), unreliable datagram (UD), unreliable connection (UC), and reliable datagram (RD) [3]. Again pulling ideas from TCP/IP, a datagram is a connectionless transmitted packet. Of these, RC, UC, and RD support RDMA (covered below). Reliable datagrams are counter-intutitive by definition and may seem as though they have limited usefulness. However, they have great use in programs that run in parallel with multiple processors [3] where scalability is a problem. Consider an application in which communication is required between ten processors (N) in a system with each processor running 100 processes (M), and there are three systems (P). From the equation of $QPs = (P-1) * (N-1) * M^2$, the required number of QPs without reliable datagrams comes to 180,000, per node. Obviously this does not scale and the reliability and performance drops dramatically. RD counteracts this by using a new mechanism called an end-to-end context. This EE context holds state information and counters that pertain to reliability. For each connected node an EE context is created, and not for each process. This significantly reduces the impact on reliability and overhead.

3) Segmentation: A simple quality of service feature of IBA is the use of a maximum transfer unit (MTU) to limit the length of a message's payload. If the MTU is too large, buffer overlows and slowdowns or deadlock may occur. Here, the message payload may be up to 2 GB in size, which is much higher than the typical MTU size of a few kilobytes [9]. The size of the MTU is set by the subnet manager (see Management section). The transport layer splits messages up depending on if they are longer than a single MTU. When split, each portion is assigned a sequence number. The receiving QP uses these sequence numbers to reassemble the data into the QPs memory block.

4) RDMA: Remote DMA addresses the overhead on the CPU caused by I/O [8]. Normally, an I/O interface must interrupt the CPU and wait for the CPU to acknowledge the interrupt and switch contexts. The CPU then initializes the DMA to allow direct access to memory for the I/O interface. However, the DMA engine must periodically interrupt the CPU for packet processing, causing more overhead. Once all packets are received and processed, the data

is copied to user memory. IBA's implementation puts all of the workload on the channel adapter (CA) and no CPU interrupting is required. The data is processed by the CA and stored directly into user memory. This eliminates overhead caused by context switching and copies of data in memory. The remote descriptor indicates that a separate CA can use the DMA function for increased transfer speeds - memory can be moved from one location to another regardless of source or destination locations. To utilize RDMA, the requesting endnode must retrieve an R_key from the destination endnode, including a starting address and memory size. This R_key allows the requesting endnode to perform reads and writes to this remote memory location and allows for multiple RDMA connections. If the endnode initializing the RDMA request uses an invalid R_key, the destination endnode refuses the request and sends a negative acknowledgement. Related to RDMA are atomic operations. These are sequences of instructions that are performed remotely on memory and a value returned as required. A lock is placed on the memory locations in question that prevents access until the instruction sequence is complete.

5) Partitioning: Since the end goal of IBA is to become the I/O fabric for all systems, a resource sharing problem occurs when endnodes are connected. Adding a physical link between the fabrics in each system will cause all of the associated fabrics to become one. However an endnode may not wish to share all of its resources, but, left alone, no privacy nor security is given. An example of a resource not to be shared would be an optical disk or a bootable main hard disk. Partitioning gives a logical break from the shared fabric by requiring endnodes to have the appropriate P_key. Each endnode holds a table of P_keys that QPs can map to. When a QP wishes to send a packet, the associated P_key is attached and delivered to the receiving QP. As illustrated in Figure 6, if the P_key matches, the request is confirmed and progresses as normal. If the P_key does not match, the packet is dropped. In order to further extend security, P_keys cannot be directly defined by a CA. They are created automatically by subnet managers and cannot be changed.

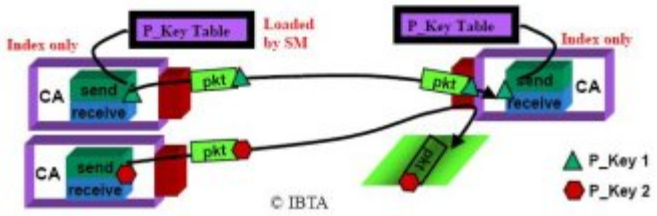

Fig. 5. Partitioning keys being accepted and rejected.

E. Upper Layer Protocols

The top layer of the stack, ULP, deals closest with the user and application. This is where the different aspects of the InfiniBand architecture can be directly used. There are two primary sections: management and consumer protocols.

1) Verbs: These protocols generate work requests that are formed by verbs. Work requests may be to copy a value in memory, store memory remotely using RDMA, etc. Verbs work with the operating system to describe IBA operations [4] - they are not intended to be API, but describe how an operation occurs, thus allowing vendor variance. Verbs are used for managing the CA, creation/destruction of QPs, etc. After processing the work request through the appropriate verb, the request is placed on a queue to be sent or received, as seen in Figure 7. The lower layers of the stack pick up from there and proceed with the request.

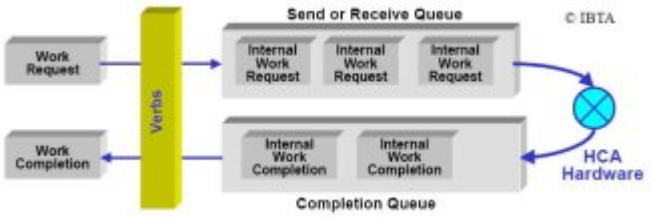

Fig. 6. Work request, verb, and CA interaction.

2) Management: IBA subnet management is handled at this layer. For each subnet, there is a subnet manager that initializes the network, assigns LIDs, sets MTUs, and loads routing tables that depict communication paths from endnode to endnode [3]. All CAs have a subnet management agent that interacts with the subnet manager, accepting LIDs, MTUs, etc. However, a management agent can also be a manager. This ability allows the network to be resilient to the possibility of a manager going

offline. Depending on implementation, each agent will poll the manager periodically; if the manager does indeed go offline, management will fail-over to one of the agents. Initialization of the network begins with each CA, router, and switch polling the other devices on the network. Each device has a management priority that is determined by MTU, data rate, or some other means, or a combination thereof. When a polling device senses another device with a higher priority, the polling device stop polling as it cannot be the subnet manager. Eventually a manager is chosen and all other devices periodically poll in the case failure has occurred. All of the management communication is made using the first two QPs, QP0 and QP1, and in VL15. As aforementioned, management packets are not considered in flow control or prioritization - they are sent as soon as they can feasibly be sent.

3) Consumer: This section defines a few protocols currently adopted for user access. Without these protocols, the applications using IBA cannot fully take advantage of the underlying hardware. The most well-known protocols are MPI, uDAPL, and IPoIB, although there are others such as SDP, SRP, and iSER [9]. The majority of research done on InfiniBand is on this topic. MPI (message passing interface) is widely used for parallel programming when processes need to interact as it is specifically made for performance in high-bandwidth connections. This protocol has the advantage that it has been previously adopted so developers will be more familiar with it. Due to this, MPI has become the standard for the InfiniBand architecture and several implementations are available (e.g. HP-MPI, Intel MPI, MVAPICH2). However, a problem with this protocol, and all other consumer protocols, is that the application must be rewritten to properly use it. The next protocol is uDAPL (User Direct Access Programming Language), which is another rather generic protocol. The goal here is to almost completely ignore the transport layer and go directly to the next layer down, giving RDMA facilities to the user. uDAPL is generic in that it can be used for any architecture that includes RDMA capacities. IPoIB (internet protocol over InfiniBand) allows TCP/IP applications to access the IBA devices. This has the great advantage that IPv4 and IPv6 addresses may be used to communicate with the different IBA devices, but IPoIB is limited and cannot use the RDMA feature.

IV. Memory Model

In a computer using a host channel adapter, a block of memory referred to as a memory region is allocated using verbs [3]. This gives the HCA a location to directly manipulate. During this registration, a local L_key is created and stored by the HCA. Any attempt at access to this memory region must be accompanied by this L_key or the attempt will be rejected in the same way that RDMA R_keys are rejected. The task of creating a memory region is time-intensive and as such should be done infrequently. During a work request, a portion of the memory region is allocated known as a memory window. This allows for segmenting the memory for multiple work requests and is done at a byte granularity. The last level of the memory model are protection domains which are directly used by QPs. They are designed to correspond to individual processes and applications.

V. Path Migration

In order to provide higher availability, multiple paths and routes can be used for fail-over pending a redundant path being cut or timing/traffic issues. There are two types of path migration mechanisms implemented by IBA: automatic path migration and send queue drain. Automatic path migration is a physical mechanism that supports fail-over to another existing path. After a set amount of failed message transfers, the hardware dynamically switches to the redundant path without needing to report a failure. The second mechanism, send queue drain, is software implemented and is used for maintenance events or suspending transmissions in general. When such an event is to occur, a send queue drain work request is stored in a QP. This disallows any future work requests to be queued. Once the send drain work request is relieved, QPs may continue to work as before.

VI. Conclusion

This document has outlined the InfiniBand architecture - the need for it and how it works. As I/O communications increase, changes must be made to allow for the required higher bandwidths. The

ability of IBA to scale as time and requirements progress allows for this technology to be widely accepted. Much thought and deliberation was put into the design by many companies all wishing to see a non-proprietary interconnect for high speed I/O capacities. Time will only tell if InfiniBand will gain popularity in these high-end computing systems. At the time of this writing, Mellanox, a leader in IBA products, announced their 2 millionth IBA port sold [6].

REFERENCES

[1] JNI Corporation, "An Introduction to InfiniBand - Bringing I/O Up to Speed," white paper, Rev. 1.1, 2002.

[2] B. Curry, T. Kehoe, P. Barbieri, B. Page, M. Ninow, A. Handa, C. Andreoli, "PCI-SIG Introduces PCI Express (Formerly 3GIO) High-Speed Serial Interconnect Specification," News release, Washington, 2002.

[3] G. F. Pfister. (2001, Nov. 26). High Performance Mass Storage and Parallel I/O: Technologies and Applications (1st ed.) [Online]. Available: http://www.gridbus.org/ raj/superstorage/chap42.pdf

[4] InfiniBand Trade Association, "InfiniBand Architecture - General Specifications," Vol. 1, Release 1.0a, June 2001.

[5] InfiniBand Trade Association, "InfiniBand Architecture - Physical Specifications," Vol. 2, Release 1.0a, June 2001.

[6] Mellanox Technologies, "Understanding PCI Buss, 3GIO and InfiniBand Architecture," Mellanox Tech., Santa Clara, CA, Dec. 10, 2001.

[7] G. F. Pfister, "Aspects of the InfiniBand Architecture," *Proc. of the 2001 IEEE Int. Conf. on Cluster Comp.*, New Port Beach, CA, 2001.

[8] Cisco Systems, "Understanding InfiniBand," Whitepaper, Cisco Systems, Inc., 2006.

[9] HP, "Using InfiniBand for high performance computing," technology brief, 2nd edition, Hewlett-Packard Development Company, L.P., January 2007.

[10] S. D. Byrd, "Data and Network Communication Technology," in Systems Architecture, 4th ed. Canada: Thomson Learning, 2003, ch. 8, pp. 317-319.