PCI Express and Advanced Switching: Evolutionary Path to Building Next Generation Interconnects

David Mayhew and Venkata Krishnan
StarGen, Inc.
Marlborough, MA 01752
[Mayhew, Krishnan]@StarGen.com
http://www.stargen.com

Abstract

With processor and memory technologies pushing the performance limit, the bottleneck is clearly shifting towards the system interconnect. Any solution that addresses PCI's bus-based interconnect, which has serious scalability problems, must also protect the huge legacy infrastructure. PCI Express provides such an evolutionary approach and allows a smooth migration towards building a highly scalable next generation interconnect. Advanced Switching further boosts the capabilities of PCI Express by allowing a rich application space to be covered that includes multiprocessing and peer-to-peer communication. Indeed, the synergy between PCI Express and Advanced Switching permits the adoption of an evolutionary yet revolutionary approach for building the interconnect of the future.

1. Introduction

System interconnects are currently in a significant state of flux. The resolution of this uncertainty will have a significant impact on system architecture, and, in the longer term, on the definition of a system itself. The historic distinctions between processor buses, mezzanine interconnects, and LANs will be challenged by the wide scale adoption of low-latency, high-bandwidth switched serial interconnects.

The current uncertainty surrounding system interconnects should resolve itself in a relatively short time with the nearly unanimous backing from the industry for the PCI-SIG controlled - and yet largely Intel created - *PCI Express* (PCIe)[1] specification. We believe that the emergence of PCIe as the heir apparent to the PCI interconnect legacy will result in the widespread adoption of a PCIe follow-on technology called *Advanced Switching* (AS)[2].

It has long been the goal of many in both the computer and communications industries to bridge the gap between their respective systems. In many ways, the adoption of inexpensive computational interfaces and parts by the telecommunications industry has already realized this goal in part; but the emergence of PCIe, to a small degree, and AS, to a much greater degree, will make the integration of computer and communications equipment a reality.

1.1. Terminology

For sake of this discussion, we define three types of interconnects typically used in today's systems. Based on where it resides as well its purpose, interconnects may be classified as a (a) Processor (b) Mezzanine (c) Local Area interconnect.

The Processor interconnect, typically known as the processor or front side bus, is the final (or initial) conduit for data moving into (or from) the CPU's on-chip cache/registers. A processor interconnect is typically used to connect the processor to another component that contains a memory controller and one or more mezzanine level interconnection ports[3]

A mezzanine interconnect has historically been a processor (CPU) independent interconnect with very limited reach (typically measured in centimeters not meters). They generally employ an address/data read/write data model with memory-like semantics. These interfaces have been targeted at allowing simple translation between processor bus memory operations and mezzanine interconnects transactions. PCI is, to

Proceedings of the 11 th Symposium on High Performance Interconnects (HOTI'03)
0-7695-2012-X/03 $17.00 © 2003 **IEEE**

date, by far the most popular and successful of the mezzanine interconnects.

Finally, Local Area interconnects or networks (LANs) grew up between the public access network (WANs) and mezzanine interconnects. LANs have continued to mature into protocols that now completely blur the historic distinction between LAN and WAN, with perhaps the only significant current distinction being that LANs use computer-borne technology (Ethernet) and WANs use telecommunication protocols (SONET and OC-48). For our purposes, the distinction between a LAN and a mezzanine interconnect lies in the LAN's reliance on protocols that do not share the mezzanine's memory read/write semantics, though it may be better to draw a simpler distinction based on the ability, or lack there of, to issue a read at the hardware level. Mezzanine interconnects can read and write; LANs can only write.

1.2. Mezzanine Interconnects

There are three types of mezzanine interconnects: shared parallel, switched parallel, and switched serial. The distinction between shared and switched is based on whether an interconnect's electrical interface has a multi-drop capability (shared) or whether the interface only supports point-to-point connections (switched). The distinction between parallel and serial is based on whether an interface is source clocked, that is, the interface is strobed by a clock signal (parallel) or whether the interface relies on some form of clock recovery, that is, the interface is self clocked (serial).

1.2.1. Shared Parallel Interconnects

The list of significant shared parallel interconnects is long and storied. A few of the major players here include: S-100[4], PC-AT[5], Multi-Bus (II)[6], VME[6], PCI[7], and PCI-X[8]. As previously stated, PCI is, to date, the shining star of this category and, indeed, of all categories. Its ubiquity provides an existence proof for the purported advantages of mezzanine interconnects. Namely that mezzanine interconnects allow a vendor independent mechanism in which a broad range of processor architectures can talk to a broader range of devices types.

Of note is the fact that currently almost the entire software infrastructure of the computer industry is tied to the interconnect model of PCI. At least some of the technologies that have/are attempting to displace PCI as the standard mezzanine interconnect of the future provide a completely transparent PCI bus programming model (namely PCI-X[8], HyperTransport[9], and PCIe[1]). Of equal interest are fortunes of InfiniBand[10] and RapidIO[11], which lack PCI transparency. Conversely, PCI-X's compatibility with both the logical and physical characteristics of PCI resulted in its relatively quick adoption in the computational server space.

Shared parallel interconnects do, in most respects, present the worst set of characteristics of the three mezzanine structures. The primary advantage of sharing is that, for very small systems, it is relatively easy to implement. Unfortunately, shared parallel interconnects have some fundamental electrical limitations (loads on a circuit), speed concerns (capacitance of lots of bus loads), reliability problems (the ability for one ill-behaved resident on a bus to make communication between all other residents impossible), scalability problems (everyone shares bandwidth from a common pool; the more residents on the bus the smaller the effective bandwidth per resident), and physical distance problems (skew in source clocked parallel structures places fundamental limitations on the distance between connectors).

1.2.2. Switched Parallel Interconnects

Many of the fundamental limitations of shared parallel interconnects are addressed by switched parallel interconnects. The electrical loading, reliability, scalability, and, to some degree, speed and physical distance issues of shared structures are solved by the adoption of switched structures. Signal skew remains an issue, as does the effective width of such connections when cabled, but, for some applications, a switched parallel interconnect is the best current technology. In general, a parallel interface can always be made serial. The only real issue is whether an appreciable amount of silicon supports a potentially serial version of a parallel interface.

A number of new standards are emerging that fall into the switched parallel mode of operation, most notably HyperTransport[9] and RapidIO[11] (attempting to migrate to a serial interface), though switch parallel interconnects are hardly new[12][13][14].

1.2.3. Switched Serial Interconnects

Switched serial interconnects solve the last of the issues with interconnect technology, which is the physical reach. A parallel interface, by definition has at least 2 signals (one clock and one data). A serial interface can be reduced to a single signal. Serial signals can also be ganged into groups of physically independent streams that can be striped on arbitrary bit intervals (bytes seem to be very popular). In many instances, the physical reach of an interconnect dictates the scale of system solution that can be addressed by the technology.

Many of the interconnect solutions that were initially parallel solutions have migrated to serial solutions. The shift from parallel to serial is naturally made when transitioning a technology from a copper-cabled solution to an optical solution. However, high solution cost remains a significant barrier to the adoption of optical solutions, and serial copper cabled solutions can be very cost effective for applications that can be solved on a backplane or that can make use of short haul (less than 5 meters) cabling.

One may vehemently claim that PCIe along with AS is just another solution in the vast compendium of interconnects that includes Infiniband[10]. There is, however, a major distinction between PCIe and other interconnect solutions, such as Infiniband. The primary strength behind PCIe is in its support for legacy PCI while addressing the inadequacies of PCI. Indeed, PCIe is fundamentally nothing more than a serialization and packetization of PCI. It is completely compatible with PCI. The additional features that PCIe offers over PCI are in addition to and not a replacement of the mechanisms supported by PCI. Hence, the reams of existing software that expect PCI to underpin them will continue to function untouched in a PCIe world. Advanced Switching further enhances the capabilities of PCI express to include support for multiprocessing and peer-to-peer computing. We believe that the evolutionary path afforded by PCIe from PCI, along with AS capabilities, would enable this interconnect to become a dominant mezzanine technology in the near future. In the following sections, we highlight the important features of both PCI Express and Advanced Switching.

The rest of the paper is organized as follows: Section 2 describes PCI Express and highlights its important features. This is followed by discussion of Advanced Switching in Section 3. Finally, Section 4 concludes the paper.

2. PCI Express

One of the reasons for PCI's remarkable success was that a few of its visionary pioneers recognized the importance of a standard model for bridging between PCI bus segments[15]. This capability was intended to solve the electrical load restrictions of PCI by allowing a single (pair if on an add-in card) load on one bus to spawn an entirely new subordinate bus segment that could have a full compliment of additional electrical loads. This standardization allowed for the creation of a robust, software-independent, PCI-to-PCI bridge market with a standard software model for recognizing devices behind bridges. More importantly for PCIe, it provided a legacy software transparent model for switching within the PCI model: a PCI switch could logically be thought of as a collection of PCI-to-PCI bridges in which one bridge is the upstream bridge, which is then connected to a private local bus via its downstream side to the upstream sides of a group of additional PCI-to-PCI bridges. This is exactly the model adopted by PCIe for its switching. PCIe switching represents a strict hierarchy of logical PCI-to-PCI bridges. A logical view of the PCIe switch that provides such an hierarchical model is shown in Figure 1.

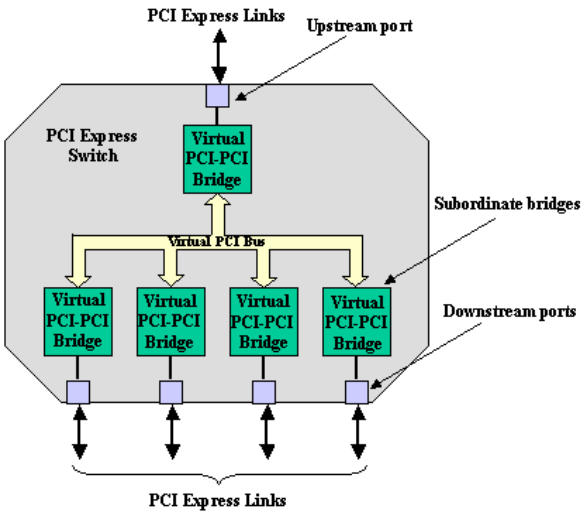


Figure 1. A PCI Express Switch provides a traditional PCI-bridging hierarchical structure.

Indeed, PCIe strictly preserves the PCI logical model, while simultaneously transforming PCI from a *shared parallel mezzanine* structure to a

switched serial one. In the following section, we describe the different layers of PCI Express in brief. In the subsequent section, we highlight the three important features of PCI Express.

2.1. PCI Express Layers

2.1.1. Physical Layer

A PCIe link is composed of one or more transmit/receive pairs. The initial specification is based on 2.5 gigabit raw data rate SERDES (referred to as a lane). The PCIe SERDES specification is very similar to the InfiniBand specification[10] with a lowered drive capability. PCIe lanes are scrambled to reduce Electro Magnetic Interference (EMI). A PCIe link is composed of 1, 2, 4, 8, 12, 16, or 32 lanes. A lane is also bit-order reversible, which is termed as *lane reversal*. For example, a 4 lane port with lane numbers A0, A1, A2, A3 could be connected to another 4 lane port with lane numbers B3, B2, B1, B0 respectively, and, during link negotiation, one of the components would agree to logically reverse its lane ordering such that A0 would be logically connected to B0, in spite of the fact that physically A0 is connected to B3. Lane reversal allows ease of routing on printed circuit boards.

PCIe links are 8b/10b encoded[16]. This has the net effect of reducing usable PCIe lane bandwidth to 2 gigabits. PCIe uses the commonly used 10-bit control codes[16](known as comma or K codes) to communicate at this layer. Such communication includes link width negotiation, lane reversal, and packet delimiting. All PCIe packets are 32-bit aligned and the K code delimiters are included in that alignment restriction.

2.1.2. Data Link Layer

PCIe links are managed by Data Link Layer Packets referred to as DLLPs. These DLLP packets, sent between link partners, are assumed to be lost (a bit corruption of a packet results in the packet being discarded and thus lost). DLLPs are used to synchronize links, exchange and update link credit information, acknowledge the receipt of packets, and to exchange link state information.

DLLPs are fixed 64-bit packets that are covered by a 16-bit CRC. After accounting for the K-code delimiters, up to 32 bits are available for DLLP type identification and data payload.

2.1.3. Transaction Layer

User data is exchanged via Transaction Layer Packets (TLPs). TLPs are reliable; hence, if TLP loss or corruption is detected, packet retransmission is required. Continued inability to successfully transmit a TLP will result in link retraining and ultimately in a link being disabled.

A TLP consists of a pair of K code delimiters, a 16-bit packet preamble, packet header, potentially some packet payload, and a 32-bit CRC. The packet preamble identifies a packet's sequence number for packet acknowledgement and loss and recovery purposes. The packet header options are a straightforward set of reads, writes, and read completions for configuration, I/O, 32-bit memory, and 64-bit memory spaces. Maximum packet size is 4k bytes, though implementations may further restrict this limit. PCIe employs a segmentation and reassembly mechanism for transcending packet size restrictions. A per packet CRC-32 supports the detection of transmission errors.

2.2. Features

2.2.1. Credit-Based Flow Control

PCI manages congestion via retry. Bi-directional packetized protocols cannot efficiently rely on a retry mechanism. Instead, PCIe employs a credit-based flow-control model[17]. This model requires that a transmitter have credit for a TLP packet before forwarding the packet (DLLPs do not require credit and can be sent at any time). This mechanism guarantees that a receiver has space to hold the packet before it is sent and allows the transmitter to adjust its data flow to match the congestive characteristics of its link partner.

The link credit update model employs a monotonic increasing counter model that is immune to DLLP loss. The significant characteristic of this model is that the transmitter uses one counter and one register and the receiver employs one counter. The receiver returns the cumulative number of credits that it has returned (with wrap at the limit of the counter size). The transmitter has a cumulative counter for the number of credits it has consumed (again with wrap) and a register in which the last update value sent by the receiver is stored. The initial credit value allocated by the receiver is used to initialize this register. The transmitter subtracts its counter from its register

to determine whether it can send any given packet. The major advantage of this model is that while the loss of a credit update may result in temporary credit starvation, there is no possibility of credit leakage. A successfully received credit DLLP is fully state restorative.

2.2.2. Virtual Channels

PCI transactions do not contain any priority information. Also, the transactions are such that writes are strongly ordered while reads cannot bypass writes. PCIe provides QoS support by employing virtual channels (VCs) and traffic classes (TCs). This allows future systems to be enhanced to take advantage of the ability to prioritize data within a switched fabric. PCIe TLPs have a 3-bit traffic class field that identifies the relative priority of the associated packet. PCIe components can implement 1, 2, 4, or 8 virtual channels. A virtual channel (VC) represents an entire set of switching resources dedicated to a specific channel. Traffic classes are mapped to virtual channels. For example, a legacy PCI infrastructure could be supported by mapping all 8 TCs to a single VC. A more sophisticated next generation system could employ 2 VCs to create two independent data flows with one flow being given priority over the other. The mapping between VCs and TCs is software controlled with a modicum of flexibility in some cases. Of course, implementation of a single VC or all 8 VCs fixes the VC to TC mapping.

2.3. PCI Express Limitations

PCIe's greatest advantage, its strict compatibility with PCI, is also one of its greatest long-term liabilities. A PCIe fabric spans a single global address space. There is no notion of a system boundary within PCIe. A PCIe fabric is, by definition, a *single* system. Multiple hosts (root complexes) cannot share a fabric. All routing within a fabric is strictly hierarchical, that is, if any significant amount of peer-to-peer communication existed in a fabric, then for statistically random traffic pattern, half of all such traffic would traverse the switch attached to the root complex. No options exist for shortening the paths between leaf nodes by directly connection switches between hierarchies. Advanced Switching (AS), the topic of the next section, addresses all these limitations and expands upon other detail.

3. Advanced Switching

Its support for PCI legacy notwithstanding, PCIe is suited only for providing scalability in systems with a single host processor (master) with a number of devices (slaves). Since all communication is under the control of a single host, PCIe is not well suited for a large application space that includes multiprocessing and peer-to-peer communication.

Advanced Switching (AS) not only enables direct unicast communication between AS end devices but also supports multicast. Its support for multiple hosts permits a high-degree of redundancy to be achieved. The additional capabilities are primarily the result of defining a new transaction layer for AS. Indeed, AS uses the same physical layer as PCIe. It also shares much of the link layer; additional DLLPs have been incorporated for the purpose of exchanging VC credit-flow control as well as congestion management messages between AS link partners. It also inherits the differentiated traffic classes (TCs) and the Virtual Channel (VC) concepts and provides rich QoS capabilities.

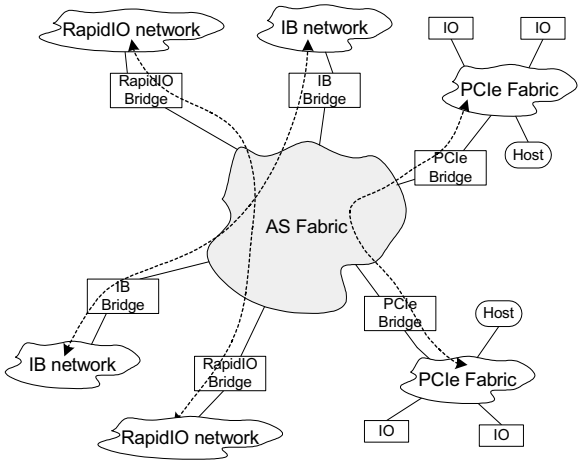


Figure 2. Protocol Tunneling through AS.

The transaction layer for AS enables a protocol-agnostic model for routing packets within AS fabrics. AS essentially provides a medium through which not only PCIe, but also other protocols such as HyperTransport[9], RapidIO[1], Infiniband[10] and StarFabric[18] may tunnel through. See Figure 2.

In the rest of this section, we restrict ourselves to discussing three important aspects of AS namely

(a) protocol encapsulation (b) routing and (c) congestion management.

3.1. Protocol Encapsulation

AS employs a split header approach. The first half of the header is purely concerned with packet routing within an AS fabric and with the identification of the type of the second half of the packet. The second half of the header is route independent and purely concerned with control and/or data delivery. In this approach, almost any protocol can fill the role of the second half of a packet. The AS fabric proper does not care what type of data it is routing, and, in general, the routed data has relatively little interest in exactly how it was routed.

AS supports a simple protocol encapsulation model in which an AS route header can be attached to a data protocol and that data protocol can be switched within AS. For example, an AS fabric can carry PCIe packets, without a PCIe fabric even being aware of the existence of AS. AS can look like a big (or small) PCIe switch. A PCIe-to-AS bridge could encapsulate a PCIe protocol packet within an AS packet on one side of a fabric such that the PCIe packet de-encapsulates on the other side of the fabric.

3.2. Routing

The design goal of AS was focused on fabrics with fewer than a thousand endpoints (though tens of thousand of endpoints are supported). This is in stark contrast to a typical IP-based LAN/WAN environment where the number of endpoints are orders of magnitudes higher. Also, the frequency with which AS endpoints are expected to enter or exit a fabric is relatively low, especially when compared to Ethernet/IP environments. Consequently, the routing adopted for AS differs from that of a traditional IP approach. In the context of describing AS routing, we describe briefly the two approaches for routing a packet through a network namely (a) *destination-based routing* and (b) *source-based routing*.

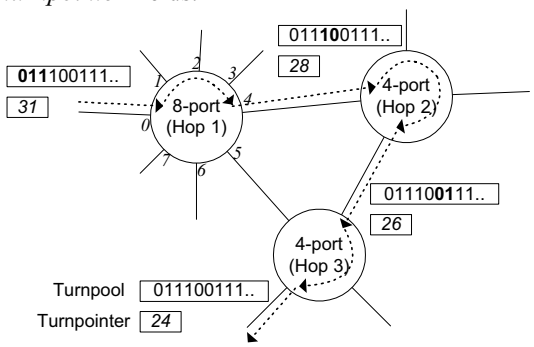
A *destination-based* routing approach requires the source node to specify only the source and destination address. A destination address is required to make a forwarding decision at each hop as a packet traverses a network. A source address is not strictly required, except when a response is required. This approach is taken in an Ethernet/IP environment in which a switch as well as a router is used for forwarding packets. The switch, which operates at the packet's link layer destination address (MAC), serves to eliminate the Ethernet collisions at the end nodes while the router is responsible for packet forwarding decisions between such switches. The forwarding decision – which requires a longest prefix match of the network layer destination address[19] – is a computationally arduous task that must be performed at wire-speed and requires additional hardware support[20]. To alleviate this problem, label based techniques such as MPLS have been proposed[21].

It may, however, be contended that if the total number of nodes that can be supported in a fabric is restricted, the forwarding table can accommodate an entry for any potential end node and thereby allow a straightforward lookup. Indeed, Infiniband[10] uses such an approach where the number of local identifiers (LID) within a *subnet* for a unicast packet is limited to a 14-bit range. Nevertheless, such a scheme still requires a routing table of 16K entries to be supported in every switch node in the fabric[22].

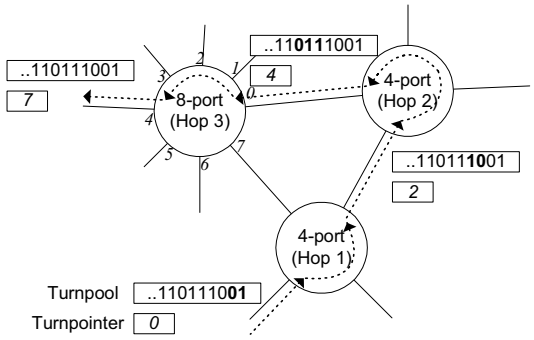
A *source-based* routing approach, on the other hand, requires the exact path to the destination to be specified by the source within the packet header. This obviates the need for a destination identifier. Since the actual forwarding decision need not be made within the switching hardware, complexity is reduced. Endpoints or internal switch nodes can determine the reverse path quite easily, so no source identifier needs to be included in the header. The ability to exactly identify the source of any given packet also solves the source identification problem (problems that arise from the misidentification of a packet's origin).

AS uses a source-based path routing approach for unicast packets and a destination-based table-lookup routing approach for multicast packets. For unicast packets, a path consists of a 32-bit *turnpool* (that includes a direction bit) and a 5-bit *turnpointer*. The turnpool contains the sequence of turns a packet must take as it traverses the fabric while the turnpointer allows each switch node to use a particular turn from the turnpool. The turnpointer is updated when a packet moves from one switch node to the next. The update is based on the number of bits each switch node consumes in the turnpool. Assuming support for loopback, a N-port switch consumes $\lceil log_2 N \rceil$

bits. Figure 3 illustrates how a packet is forward- and backward-routed through a set of AS switch nodes using the *turnpool* and *turnpointer* fields.



(a) Forward Routing



(b) Backward Routing

Figure 3. AS Source-based Path Routing.

Since packets for a particular source-destination pair use the same path for routing, there is no scope for packet re-ordering[23] – one that occurs in adaptive destination-based routing approaches – allowing end node protocols to be greatly simplified. Such a deterministic approach to routing however has the potential to deadlock. Though conventional deadlock avoidance techniques that allow packets to move from one VC to another during transit[24] are not applicable in AS, deadlock can indeed be avoided in AS. One solution would be to partition the fabric into multiple sub-fabrics where each sub-fabric uses a particular VC[25]. Alternatively, at configuration time, a central fabric manager can set up paths in each source node such that deadlock-free routing is guaranteed.

Unlike unicast, the source-based path routing approach becomes prohibitively expensive for a multicast packet that targets several destination nodes. Hence the destination-based table-lookup approach for multicast. The lookup is based on a multicast group identifier which may be as large as 64K. However, a switch is required to support a minimum of only one multicast group. Each valid entry in the multicast table contains the set of egress ports in the switch onto which the incoming packet must be replicated. Finally, even though multicast packets do not use a path routing approach, such packets do build a path en-route to their destination. This permits any response packet from a destination node to make use of path routing. As a result, there is also no need for the source node to provide its identifier for multicast packets.

The destination-based approach is more flexible and can better accommodate changes to the fabric. However, the relatively small number of end-devices in AS (10-100s of nodes) and the infrequent changes to the fabric itself permits each source in the fabric to pre-determine its path to every other destination. Overall, the initial set-up cost for enabling path routing may be preferable to the overhead incurred in the destination-based approach.

3.3. Congestion Management

AS, like PCIe, uses credit-based flow control[17]. This requires the sender to have the requisite VC credits for a packet before it can begin transmitting on its output port. Unlike in a TCP/IP environment, the receiver does not drop packets due to lack of buffer space. AS only uses retransmission for packets that experience data corruption during transmission, and then only between link partners. As a result, the infamous congestion collapse condition[26] is avoided. The credit-based flow control notwithstanding, an AS switch is still prone to congestion when the aggregate bandwidth of incoming flows targeting a common output port exceeds the port's bandwidth.

Congestion in AS occurs on a VC basis since the output port bandwidth is typically partitioned across multiple VCs based on the VC priority. Thus, a switch may be severely congested for a particular VC and yet may allow other VC flows to go unimpeded. Congestion as described in the rest of this section is therefore in the context of a specific VC.

In keeping in line with other approaches, congestion detection is done in AS switch nodes.

The method adopted to detect congestion is however implementation specific. For instance, switches with output buffers may use the average queue size along with a threshold[27]; alternatively, one may also detect congestion based on input buffer status[28]. Packets, however, may not be dropped[27] when congestion is detected unless they are explicitly marked as *perishable* by the source node.

The congestion notification uses the Explicit Congestion Notification (ECN)[29] approach. Here, the switch plays a passive role by marking downstream packets when congestion is detected. It is the responsibility of the upper-level protocols on the destination node to take appropriate action on receipt of such marked packets. The action may include notifying the source node to back off. AS not only supports the typical ECN model but also takes a step further by allowing the switch node to play an active role in the notification process. Indeed, the switch may notify the source directly and with path routing in place, the backward path to the source is easily generated in the switch. The source can respond to the switch-initiated notification without involving upper-level protocols. An AS switch generates an independent *BECN* (Backward ECN) packet to the source of a congested packet and marks a congested packet's *FECN* (Forward ECN) flag to signal the destination that the packet experienced congestion.

Finally, the source node responds to a BECN by adjusting its packet injection rates using the additive increase-multiplicative decrease (AIMD) algorithm or one of its variants[30]. Overall, the approach taken by AS is similar to the BECN/FECN settings in frame relay switches[31]. However, unlike AS switches, frame relay switches do not create a new control packet but set the BECN bit on an existing packet heading to the source.

In addition to supporting conventional endpoint based congestion management approach, AS also supports a switch-centric scheme. Since network traffic is extremely bursty and does not necessarily follow a Poisson model[32], transient congestion will be a common occurrence in some systems.

Consider the case when multiple flows share the same output port on a switch and diverge in the adjacent downstream node. Since AS uses flow credits, any transient congestion on one of the output ports in the downstream node affects all flows. This is illustrated in Figure 4.

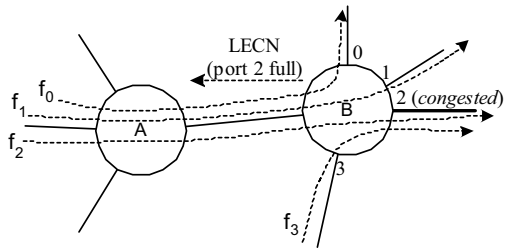


Figure 4. Transient Congestion on Port 2 in downstream switch B affects flows f_0 and f_1 as they share a common output port in switch A with flow f_2. A LECN message from switch A (denoting port 2 being full) allows switch A to give higher priority to f_0 and f_1 packets.

AS addresses this problem by allowing switch nodes to periodically send notification messages to adjacent fabric nodes about the buffer usage of its output ports. These LECN (Local ECN) messages permit the upstream node to refine its scheduling such that higher priority may be accorded to packets targeting non-congested output ports in the downstream switch over those targeting potentially congested output ports.

4. Conclusions

The approach afforded by the PCI Express architecture allows complete support for PCI thereby protecting the existing investment in legacy PCI infrastructure. Yet, it addresses the bus-based limitations of PCI. With such ease of migration afforded by PCI Express, its wide spread adoption is indeed beyond any doubt.
Advanced Switching, while sharing the link and physical layers of PCI Express, further boosts PCIe's capabilities. This includes support for multiprocessing and peer-to-peer communication. AS also goes far and beyond enhancing PCI Express. Its protocol-agnostic nature allows it to serve as a universal platform for supporting a much wider range of protocols.

Together, PCI Express and Advanced Switching have the potential to provide an evolutionary yet revolutionary approach for building the next generation interconnects.

5. References

1. PCI Special Interest Group, "PCI Express Base Specification Revision 1.0a," Apr 2003.

2. Intel White Paper, "Advanced Switching for the PCI Express Architecture," May 2003. http://www.intel.com/design/network/25173701.pdf
3. AMD White Paper, "AMD Eighth-Generation Processor Architecture," Oct 2001. http://www.amd.com
4. "IEEE Standard 696 Interface Devices," IEEE Computer Society, June 1982.
5. "PC/104 Specification," PC/104 Embedded Consortium, Aug. 2001.
6. J. Zalewski, "*Advanced Multimicroprocessor Bus Architectures*," IEEE Computer Society Press, 1995.
7. PCI Special Interest Group, "PCI Local Bus Specification, Revision 2.2," Dec. 1998.
8. PCI Special Interest Group, "PCI-X 2.0 Protocol Specification Revision 2.0," July 2003.
9. HyperTransport Consortium, "HyperTransport Technology: Simplifying System Design," Oct. 2002. http://www.hypertransport.org
10. Infiniband Trade Association, "Infiniband Architecture Specification, Release 1.0," Oct. 2000. http://www.infinibandta.org
11. RapidIO Trade Association, "RapidIO Technical Whitepaper Rev 3." http://www.rapidio.org
12. N. J. Boden, et al , "Myrinet: A Gigabit Per Second Local Area Network," *IEEE Micro*, Vol. 15, No. 1, Feb. 1995, pp. 29-36.
13. F. Petrini et al, "The Quadrics Network: High Performance Clustering Technology," *IEEE Micro*, Vol. 22, No. 1, Jan/Feb 2002, pp. 46-57.
14. R.W. Horst and D. Garcia, "ServerNet SAN I/O Architecture," *Hot Interconnects V*, Aug 1997.
15. PCI Special Interest Group, "PCI-to-PCI Bridge Architecture Specification Revision 1.1," Dec. 1998.
16. A.X. Widmer and P.A. Franaszek, "A DC-Balanced Partitioned-Block 8B/10B Transmission Code," *IBM Journal R&D*, Vol. 27, Sep 1983, pp. 440-451.
17. H.T. Kung and R. Morris, "Credit-Based Flow Control for ATM Networks," *IEEE Network*, Vol. 9, No. 2, Mar/Apr. 1995, pp. 40-48.
18. StarFabric Trade Association, "StarFabric: Universal Switched Interconnect Technology." http://www.starfabric.org
19. V. Fuller, T. Li, J. Yu, and K. Varadhan**. "**Classless inter-domain routing," *IETF RFC 1519***,** Jun. 1993.
20. P. Gupta, S. Lin, and N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds," *Proc. IEEE Infocom'98*, Mar. 1998.
21. E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *IETF RFC 3031*, Jan. 2001.
22. Agilent Technologies HDMP-2840 Product Brief, Aug 2002. V. Jacobson, "Congestion Avoidance and Control," *Proc. SIGCOMM'88*, Aug. 1988, pp. 314-329.
23. J. Bennett, C. Partridge, and N. Sheetman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Transactions on Networking*, Vol. 7, No. 6, pp.789-798, Dec. 1999.
24. W.J. Dally and C.L. Seitz, "Deadlock-free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, Vol. 36, pp. 547-553, May 1987.
25. T. Skeie, O. Lysne and I. Theiss, "Layered Shortest Path (LASH) Routing in Irregular System Area Networks," *Workshop on Communication Architecture for Clusters (CAC'02)*, Apr 2002.
26. J. Nagle, "Congestion Control in IP/TCP Internetworks," *IETF RFC896*, Jan. 1994.
27. S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, pp. 397-413, Aug. 1993.
28. J.R. Santos, Y. Turner, and G. Janakiraman, "End-to-end congestion control for Infiniband," *Proc. IEEE Infocom'03*, Apr. 2003.
29. K.K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *IETF RFC3168*, Sep. 2001.
30. S. Floyd, "Congestion Control Principles," *IETF RFC2914*, Sep. 2000.
31. W. Goralski, "*Frame Relay for High-Speed Networks*," John Wiley & Sons, Inc., New York, Jan. 1999.
32. V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transaction on Networking*, Vol. 3, No. 3, pp. 226-244, June 1995.