Undestanding RapidIO, PCIe and Ethernet

By Barry Wood
Tundra

While there are many ways to connect components in embedded systems, the most prominent are the high speed serial standards of Ethernet, PCI Express, and RapidIO. All of these standards leverage similar Serialiser/Deserialiser (SerDes) technology to deliver throughput and latency performance greater than what is possible with wide parallel bus technology. For example, RapidIO and PCI Express leveraged the XAUI SerDes technology developed for Ethernet.

The trend towards leveraging a common SerDes technology will continue with future versions of these specifications. The implication is that raw bandwidth is not a significant differentiator for these protocols. Instead, the usefulness of each protocol is determined by how the bandwidth is used.

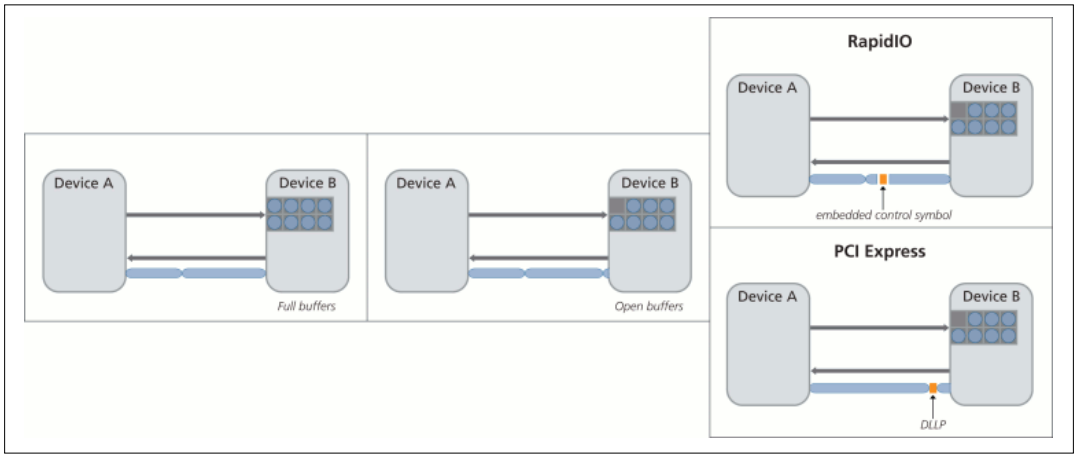

Figure 1: RapidIO embedded control symbols and PCIe DLLPs.

Protocol summaries

Most designers are familiar with basic Ethernet protocol characteristics. Ethernet is a 'best effort' means of delivering packets. The software protocols built on top of the Ethernet physical layer, such as TCP/IP, are necessary to provide reliable delivery of information, as Ethernet-based systems generally perform flow control at the network layer, not the physical layer. Typically, the bandwidth of Ethernet-based systems is over-provisioned between 20 and 70 per cent. Ethernet is best suited for high latency inter-chassis applications or on-board/inter-board applications where bandwidth requirements are low.

PCI Express (PCIe), in contrast, is optimised for reliable delivery of packets within an on-board interconnect where latencies are typically in the microsecond range. The PCIe protocol exchanges Transaction Layer Packets (TLPs) such as reads and writes, and smaller quantities of link-specific information called Data Link Layer Packets (DLLPs). DLLPs are used for link management functions, including physical layer flow control. PCIe was designed to be backwards compatible with the legacy of PCI and PCI-X devices, which assumed that the processor(s) sat at the top of a hierarchy of buses. This had the advantage of leveraging PCI-related software and hardware intellectual property. As discussed later in this article, the PCI bus legacy places significant constraints on the switched PCIe protocol.

RapidIO technology has been optimised for embedded systems, particularly those which require multiple processing elements to cooperate. Like PCIe, the RapidIO protocol exchanges packets and smaller quantities of link-specific information called control symbols. RapidIO has characteristics of both PCIe and Ethernet. For example, RapidIO provides both reliable and unreliable packet delivery mechanisms. RapidIO also has many unique capabilities which make it the optimal interconnect for on-board, inter-board, and short distance (<100 m) inter-chassis applications.

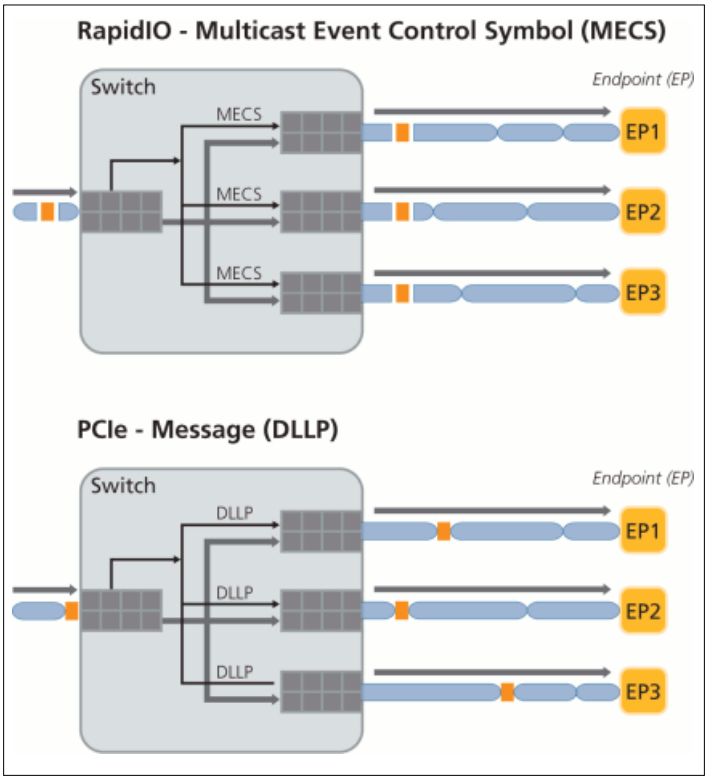

Figure 2: RapidIO Multicast Event Control Symbol and PCIe DLLP.

Physical layer

At the physical/link layer, the protocols have different capabilities when it comes to flow control and error recovery. Ethernet flow control is primarily implemented in software at the network layer, as this is the most effective for large networks. Ethernet's only physical layer flow control mechanism is PAUSE, which halts transmission for a specified period of time. The limited physical layer flow control means that Ethernet networks discard packets to deal with congestion.

In contrast, PCIe and RapidIO physical-layer flow control mechanisms ensure reliable delivery of packets. Each packet is retained by the transmitter until it is acknowledged. If a transmission error is detected, a link maintenance protocol ensures that corrupted

packets are retransmitted.

PCIe ensures delivery using DLLPs, while RapidIO uses control symbols. Unlike DLLPs, RapidIO control symbols can be embedded within packets. This allows RapidIO flow control information, such as buffer occupancy levels, to be exchanged with low latency, allowing more packets to be sent sooner. Figure 1 illustrates this concept. In the leftmost panel, Device A cannot send any packets to Device B because the buffers in Device B are full. Device B is sending packets to Device A continually. In the middle panel, one buffer in Device B becomes free. At this point, Device B must inform Device A that it can send a packet. In the PCIe panel on the bottom right, the DLLP cannot be transmitted until transmission of the current packet is complete. In the RapidIO panel on the top right, a control symbol is embedded in a packet that is being transmitted. This allows the RapidIO protocol to deliver packets with lower latency and higher throughput than the other protocols. The ability to embed control symbols within packets allows the rest of the RapidIO flow control story to be much richer than PCIe or Ethernet, as discussed later in this article.

Beyond more efficient flow control, the ability to embed control symbols within packets gives RapidIO an ability that currently neither PCIe nor Ethernet can offer. Control symbols can be used to distribute events throughout a RapidIO system with low latency and low jitter (Figure 2). This enables applications such as distribution of a common real time clock signal to multiple end-points, or a frame signal for antenna systems. It also can be used for signalling other system events, and for debugging within a multi-processor system. As shown in Figure 2, PCIe DLLPs introduce a significant amount of latency and jitter every time the DLLP is transferred through a switch. In contrast, the RapidIO protocol allows a signal to be distributed throughout a RapidIO fabric with less than 10 Unit Interval of jitter and 50 nanoseconds of latency per switch, regardless of packet traffic.

PCIe and Ethernet may choose to extend their respective specifications in future to allow events to be distributed with low latency. Introduction of a control-symbol like concept would be a large step for Ethernet. Several initiatives are underway within the Ethernet ecosystem to improve Ethernet's abilities within storage applications that may need a control-symbol like concept. Ethernet is also being enhanced to incorporate simple XON/XOFF flow control.

PCIe currently does not allow DLLPs to be embedded within TLPs as this concept is incompatible with the legacy of the PCI/X bus operation. DLLPs embedded within TLPs create periods where no data is available to be placed on the legacy bus. PCIe end-points could operate in store-and-forward mode to ensure that packets are completely received before being forwarded to the bus, at the cost of a drastic increase in latency and lowered throughput. Given the PCIe focus of on-board interconnect for a uniprocessor system, and the continued need to maintain compatibility with legacy bus standards, it is unlikely that the PCIe community will be motivated to allow DLLPs to be embedded within TLPs.

Bandwidth options

Beyond flow control and link maintenance, the most obvious difference between Ethernet, PCIe and RapidIO at the physical/link layer are the bandwidth options supported. Ethernet has a long history of evolving bandwidth by ten times with each step. Ethernet currently operates at 10Mbps, 100Mbps, 1Gbps, and 10Gbps. Some proprietary parts also support a 2Gbps (2.5 Gbaud) option. Next generation Ethernet will be capable of operating at 40 and/or 100Gbps.

PCIe and RapidIO take a different approach, as on-board, inter-board and inter-chassis interconnects require power to be matched with the data flows. As a result, PCIe and RapidIO support more lane rate and lane width combinations than Ethernet. PCIe 2.0 allows lanes to operate at either 2 or 4Gbps (2.5 and 5 Gbaud), while RapidIO supports lane rates of 1, 2, 2.5, 4 and 5Gbps (1.25, 2.5, 3.125, 5, and 6.25 Gbaud). Both PCIe and RapidIO support lane width combinations from a single lane up to 16 lanes. PCIe also supports a 32 lane port in its specification.

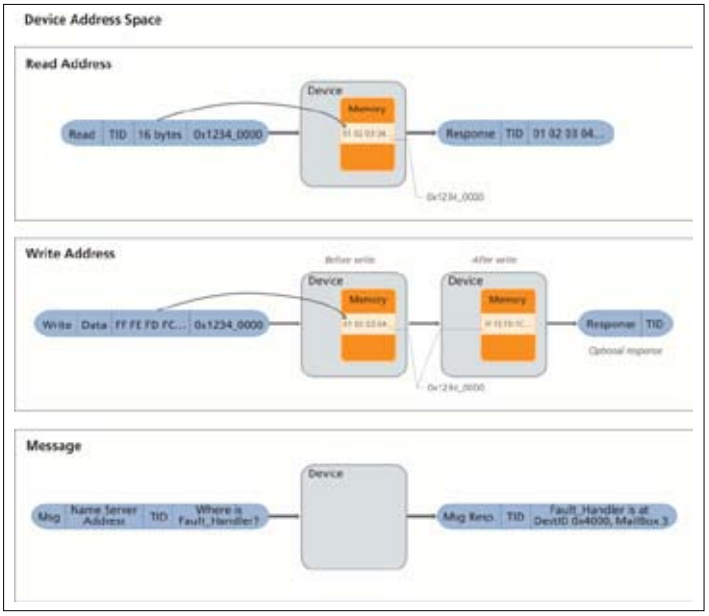

Figure 3: Read, write, and message semantics. The read semantic retrieves data 01 02 03 04 from address 0x1234_0000. The write semantic writes data FF FE FD FC to address 0x1234_0000. The message semantic determines the location of Fault_Handler without knowing referencing the memory map.

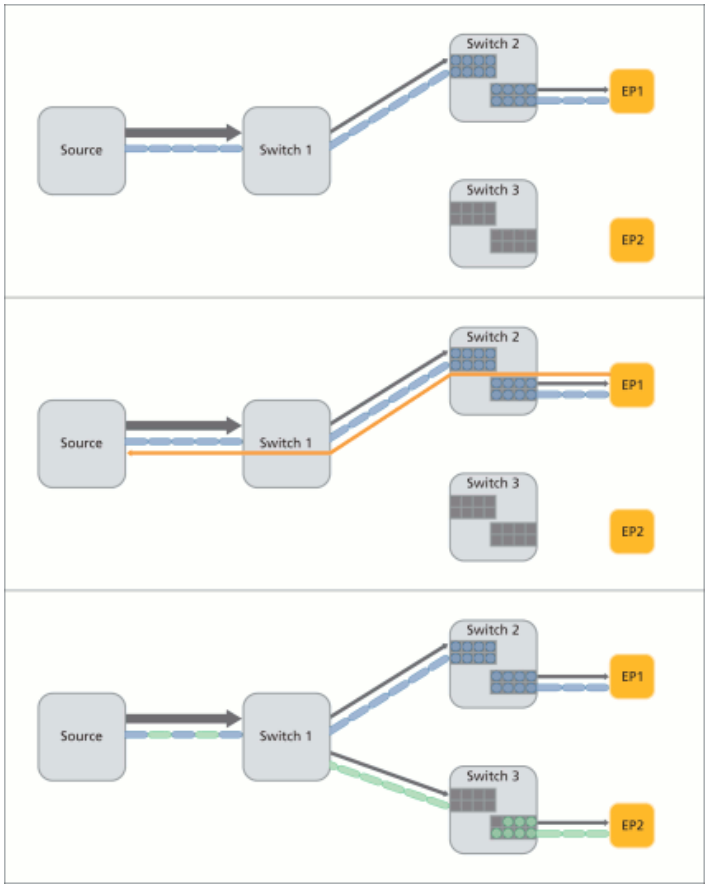

Figure 4: RapidIO's virtual output queue backpressure mechanism.

For a given lane width, a RapidIO port can supply both more and less bandwidth than PCIe, allowing system designers to tune the amount of power used to the data flows in a system.

Transport layer

The RapidIO and Ethernet specifications are topology agnostic. Any set of end-points can be connected using any fabric topology, including rings, trees, meshes, hyper-cubes, and more esoteric geometries such as entangled networks. Packets are routed through these topologies based on their network address. An example of an Ethernet network address is an Internet Protocol (IP) address. RapidIO network addresses are called destination identifiers, abbreviated as destIDs.

Topology-agnostic protocols enable a wide variety of resource sparing and redundancy options. For example, some systems use an N+1 sparing strategy to ensure high availability and reliability without significantly increasing the amount of hardware. (Sparing provides unused equipment as a backup for in-use equipment. In N+1 sparing, N components have one common spare, so that the system can continue to operate if one of the N components fails. This strategy is also known as 1: N sparing.) Support for different topology options also allows system designers to eliminate performance bottlenecks in systems by matching data paths to data flows. System expansion and evolution is unconstrained within RapidIO and Ethernet networks.

In contrast, PCIe supports a tree structure with a single Root Complex at the top of the hierarchy. Several routing algorithms are used, depending on the type of TLP. The single topology supported by PCIe is part of the legacy of the PCI/X bus support. The PCIe standard has been extended with Multi-Root I/O Virtualisation (MRIOV), which increases the number of physical topologies that PCIe can support. However, as MRIOV is expensive to implement, it is not certain if it will be adopted within the PCIe ecosystem. PCIe components also support NTB functionality, whereby one PCIe hierarchy is granted access to a defined memory range in another PCIe hierarchy. System reliability and availability of this approach are discussed further on in this article.

Many Ethernet-based systems are also functionally limited to tree structures. For example, many implementations of Ethernet make use of variations on the Spanning Tree Protocol to pare down the set of available links to a tree of used links. As part of an initiative to extend Ethernet into the storage network, protocols are being specified that will enable the creation of a spanning tree for each of the 4096 virtual channels supported by Ethernet. This will allow more links to be used simultaneously. However, support for 4096 virtual channels increases the complexity of switching, and may require more buffering and increased latency through the switch.

Logical layer

There are several large differences between RapidIO, PCIe and Ethernet at the logical layer. The most obvious differences are the semantics supported, as shown in Figure 3. PCIe packets support address-based read and write semantics. In PCIe, the entity which originates a read or write must know the target address in the system's global memory map. This is a natural approach for a control plane application. However, this reliance on a global address map can lead to tightly coupled software systems which are difficult to evolve.

The PCIe protocol also supports messaging through Message TLPs. However, Message TLPs support a limited number of functions such as interrupt and reset signals. This is much different than Ethernet and RapidIO message packets, which are used for inter-process communication.

Unlike PCIe, software protocols built on the Ethernet physical layer only support messaging semantics. To send a message, the originator only needs to know the address of the recipient. Addressing schemes are generally hierarchical, so that no node must know all addresses. Addresses may change as a system evolves, enabling software components to be loosely coupled to each other. These attributes are necessary for data plane applications.

RapidIO supports both read/write and messaging semantics. Beyond the obvious architectural advantages and system flexibility, the support of read/write and messaging allows the use of a single interconnect for both control and data planes. RapidIO systems are thus simpler than those which must incorporate both PCIe and Ethernet, which in turn provides the benefit of reducing both power consumption and cost.

PCIe is unlikely to incorporate inter-process communication messaging semantics within its packet definitions, as these conflict with the operation of the legacy PCI and PCI-X busses. PCIe is designed around the concept of a Root Complex which contains the only processors in the system. In this paradigm, no messaging is done over PCIe to other software entities, so messaging semantics have no value.

It could be argued that Ethernet supports read-and-write semantics through the Remote Direct Memory Access (RDMA) protocol, which supports direct memory copies between devices. However, RapidIO (and PCIe) read-and-write semantics are far more efficient than RDMA. The RDMA protocol is built upon other Ethernet-based protocols, such as TCP, and requires a significant amount of header overhead for each packet. RDMA implementations are much more expensive in terms of latency and bandwidth when compared with RapidIO read/write transactions. While some RDMA offload engines exist, it is difficult to envision using RDMA for control plane functions such as programming registers.

One possible application of messaging semantics is the encapsulation of other protocols. RapidIO has standards for encapsulating a variety of protocols. The ability to encapsulate gives system designers many advantages, including future-proofing RapidIO backplanes. Any future, legacy or proprietary protocol can be encapsulated and transported using standard RapidIO components. For example, the existing RapidIO specification for Ethernet encapsulation allows designers to leverage Ethernet-based software within a RapidIO-based system.

Ethernet also supports encapsulation, and has several encapsulation standards to choose from. RapidIO can encapsulate more efficiently than Ethernet, as the layers of Ethernet-based protocols require more header overhead.

At one time, there was an initiative which attempted to standardise encapsulation of a variety of protocols over a PCIe-like fabric. This initiative failed years ago due to the complexities involved. It was too difficult to extend PCIe, with its legacy bus related requirements, to be a fabric for embedded systems.

Reliability, availability

Most systems have requirements for reliability and/or availability. Systems with reliability and/or availability requirements need mechanisms for detection of errors, notification of errors, analysis and isolation of the faulty component, and recovery. At a high level, PCIe, RapidIO, and Ethernet have similar capabilities in all of these areas. There are significant differences in sparing strategies, and in the ability to rapidly isolate the system from the effects of faulty components.

In the beginning, the Internet used software protocols at the network level to deliver reliable communications over a network that could experience severe damage. Consequently, Ethernet's original error management capabilities were aimed at detection, isolation and avoidance of new holes in the network, rather than robustness of individual systems. System reli-

ability was achieved through duplication of network components. Later enhancements have added standardised error capture and error notification mechanisms to simplify network management.

RapidIO has sparing, error capture and error notification capabilities similar to that of Ethernet.

PCIe supports limited sparing strategies, as its transport layer is limited to tree structures. PCIe's Non-Transparent Bridging, previously mentioned, allows two or more tree structures to communicate, and is sufficient for 1+1 sparing (also known as 1:1 sparing). NTB is difficult to scale to systems employing N+M sparing. In theory, Multi-Root I/O Virtualisation (MRIOV) can be used to support N+M sparing in PCIe systems where N+M totals eight or less. However, because the sub-trees within a MRIOV system cannot communicate with each other, recovery from failures may require a system outage in order to reconfigure the system to isolate failed components and to incorporate new ones.

Ethernet's error detection mechanisms are generally slower when compared to PCIe and RapidIO, as Ethernet was designed for a network that is distributed across the planet. Both PCIe and RapidIO have error detection and notification mechanisms with much lower latency compared to Ethernet.

While both PCIe and RapidIO guarantee delivery of packets, under error conditions they can discard packets to prevent faulty components from causing fatal congestion. However, the PCIe error condition mechanism is not configurable. Packets are always discarded when a link must retrain. Additionally, the PCIe isolation mechanism is activated only after several milliseconds. These are not desirable behaviours in all systems.

In contrast, the RapidIO standard allows a system-specific response to errors such as link retraining. When errors occur, the system can immediately start discarding packets, or it can retain packets and allow congestion to occur. RapidIO makes use of a 'leaky bucket' method of counting errors, with two configurable thresholds. The DEGRADED threshold gives system management software an early warning that a link is experiencing errors. The FAILED threshold can be set to trigger packet discard at a user defined error rate. The flexibility of RapidIO error management reflects the varying needs of embedded system designers.

Flow control

Flow control cuts across the physical, transport and logical layers of interconnect specifications. Flow control capabilities are critical to ensuring the correct and robust operation of a system under a range of conditions, including partial failure and overload. Flow control mechanisms allow the bandwidth available to be used as efficiently and completely as possible. Flow control strategies are becoming more and more important in order to minimise the amount of bandwidth and power wasted by over-provisioning high frequency serial links.

It is not possible to talk about a unified Ethernet flow control strategy, as many unrelated Ethernet-based messaging standards have protocol-specific flow control strategies to avoid packet discard. Generally, the flow control strategies of these standards are based on reducing the rate of transmission when packet loss is detected. The flow control strategies are generally implemented in software, and require significant buffering capabilities to allow for retransmission.

PCIe flow control is limited to the physical layer. The PCIe flow control mechanism is based on tracking credits for packet headers and data chunks, tracked separately for Posted, Non-Posted, and Completion transactions.

RapidIO specifies flow control mechanisms at the physical and logical layers. Physical layer flow control mechanisms are designed to deal with multiple-microsecond periods of congestion. At the physical layer, RapidIO offers PCIe-style flow control supplemented with a simple retry mechanism. The simple retry mechanism is very efficient to implement, with minimal performance penalty compared to PCIe-style flow control. The RapidIO physical-layer flow control also includes a virtual output queue-based backpressure mechanism. This mechanism, introduced in RapidIO 2.0, allows switches and end-points to learn which destinations are congested, and to send traffic to uncongested destinations. This feature enables distributed decision making, ensuring that available fabric bandwidth is maximally utilised. The latency of decision-making is low, as congestion information is exchanged using control symbols which, as already discussed, can be embedded within RapidIO packets.

The virtual output queue backpressure mechanism is illustrated in Figure 4. In the top panel, the source is able to send much faster than EndPoint (EP) 1 can accept packets. This results in a congestion status control symbol being sent by EP 1 to Switch 2, which cascades the message back to Source. The congestion status control symbol could also have been originated by Switch 2 when it detected congestion on the port connected to EP 1. Once Source receives the congestion status control symbol, it starts to send packets to EP 2, reducing the rate of packet transmission to EP 1.

RapidIO's logical layer flow control mechanisms are designed to avoid congestion within the fabric by metering the admission of packets to the fabric, thus managing congestion at the network level. This approach is similar to Ethernet-based software protocols. Admission of packets for specific flows can be administered through an XON/XOFF type protocol, as well as by rate-based and credit-based flow control. Perhaps most importantly, these flow control mechanisms can also be used at the application layer to engineer software application performance. Best of all, these mechanisms were designed to be implemented in hardware, freeing precious CPU cycles to deliver value to the customer. RapidIO's flow control mechanisms ensure that RapidIO-based systems use available bandwidth in an efficient, predictable manner.

Summary

Ethernet, PCIe, and RapidIO are all based on similar SerDes technology. SerDes technology is therefore not a differentiator for these technologies, but the way they make use of the available bandwidth is. Each technology is optimised for a particular application space.

Ethernet has been optimised for networks which are geographically distributed, have long latencies, and dynamic network configurations. PCIe has been optimised to support a hierarchical bus structure on a single board. Both have been used for on-board, inter-board, and inter-chassis communications, and in many cases both are used in the same system. RapidIO has the potential to combine the benefits of these two interconnects into a single interconnect, with associated power and cost savings.

RapidIO is the best interconnect choice for embedded systems. RapidIO has capabilities similar to PCIe and Ethernet, as well as capabilities that other interconnects will not duplicate, such as:

- Low latency, low jitter distribution of system events
- Combined link level and network level flow control mechanisms
- Configurable error detection and topology agnostic routing enable efficient sparing, high reliability and availability
- Hardware implementation of both read/write and inter-process communication messaging semantics

These capabilities allow system architects to create better performing systems which consume less power and are easier to scale.