Technology White Paper

System Interconnect Fabrics: Ethernet versus RapidIO® Technology

By Greg Shippen, System Architect

Freescale Semiconductor's Digital Systems Division, NCSG

Member, RapidIO Trade Association

Technical Working Group and Steering Committee

RapidIO® Trade Association - 3925 W. Braker Lane, Ste. 325 - Austin, Texas 78759 - 512.305.0070
Used by permission.

Foreword

An important trend in the embedded industry is the move from proprietary solutions toward standard interconnects and mechanicals (e.g. ATCA, Ethernet, RapidIO®). Traditionally, Ethernet is the incumbent interconnect technology for embedded systems, yet the existence of alternatives has been driven by the realization that some high-performance applications exceed the limits of this traditional protocol. The RapidIO standard, with growing switch, DSP and processor endpoint support, has been deployed in many applications and continues to gain widespread support. RapidIO is particularly popular in wireless infrastructure equipment, where DSP connectivity is critical. As Ethernet and RapidIO continue to evolve, a comprehensive review and comparison is needed to help designers evaluate and select the protocol that best suits their next-generation designs.

This white paper, *System Interconnect Fabrics: Ethernet Versus RapidIO Technology,* is a comprehensive compilation of embedded interconnect information. It provides a high-level market perspective and delivers a detailed technical examination of these two technologies. The paper begins with a top-level description of market requirements for the logical, transport, and physical layers, then moves into a thorough discussion of each protocol.

The author builds a foundation for comparison and discussion through an historical overview of both RapidIO and Ethernet. Understanding the origin of these two standards helps to clarify their current use in embedded applications. A hierarchical side-by-side comparison of the two technologies provides detailed technical data, starting with the logical layer, then moving to the transport and physical layers. Within each comparison is a discussion of the various distinctions of the interconnects, highlighting significant differences.

In addition to the protocol discussions, this paper examines practical considerations such as power usage, throughput and latency, as well as economic factors related to silicon die size/cost, the volume of silicon shipped, and the size of the silicon and support ecosystems.

The author makes the case for Ethernet to continue gaining strength in the WAN and Internet markets while RapidIO becomes the preferred choice for applications that demand higher levels of performance and quality of service (QoS). In closing, the author provides a list of value propositions that should be analyzed by system designers as they evaluate how each interconnect best meets specific application requirements.

System Interconnect Fabrics: Ethernet versus RapidIO Technology delivers a clear and complete analysis that engineers will find useful well into the future of these two successful technologies.

--Linley Gwennap, principal analyst of The Linley Group

February, 2007

Freescale Semiconductor
Application Note

Document Number: AN3088
Rev. 0, 02/2007

System Interconnect Fabrics: Ethernet Versus RapidIO® Technology

Greg Shippen
System Architect
Digital Systems Division, NCSG
Freescale Semiconductor, Inc., Technical Working Group and Member of the RapidIO Trade Association Steering Committee

Market pressures demand that each new generation of networking and communications system deliver both higher performance and new functionality at a lower price. Increasingly, system providers seek to meet these requirements with industry-standard solutions ranging from protocols and interconnects to mechanical designs.

In many networking and wireless applications, Ethernet is the first interconnect technology considered. However, recent alternatives such as RapidIO® have now become viable. RapidIO technology offers several advantages over 1G and 10G Ethernet:

- Low CPU overhead
- Higher effective bandwidth
- Superior Quality-of-Service
- Superior latency and latency-jitter characteristics
- Competitive or lower cost
- Growing ecosystem

This document reviews the use of Ethernet and RapidIO technology as a system interconnect fabric, comparing them against the requirements for such fabrics. Quantitative analysis is presented where possible. A LAN or WAN setting is not the focus of this document.

Contents

freescale™ semiconductor

1 Interconnects Everywhere

Many networking and communications systems share similar architectural features. Figure 1 illustrates a generic system with a pipeline data flow. External protocols can be terminated at the entry or encapsulated at the boundary of the system and the data temporarily placed into memory arrays. One or more functional units read the data from memory using internal protocols and then transform it in application-specific ways before again placing it into temporary storage. From here the system moves this data to the exit point of the system using an externally visible protocol.

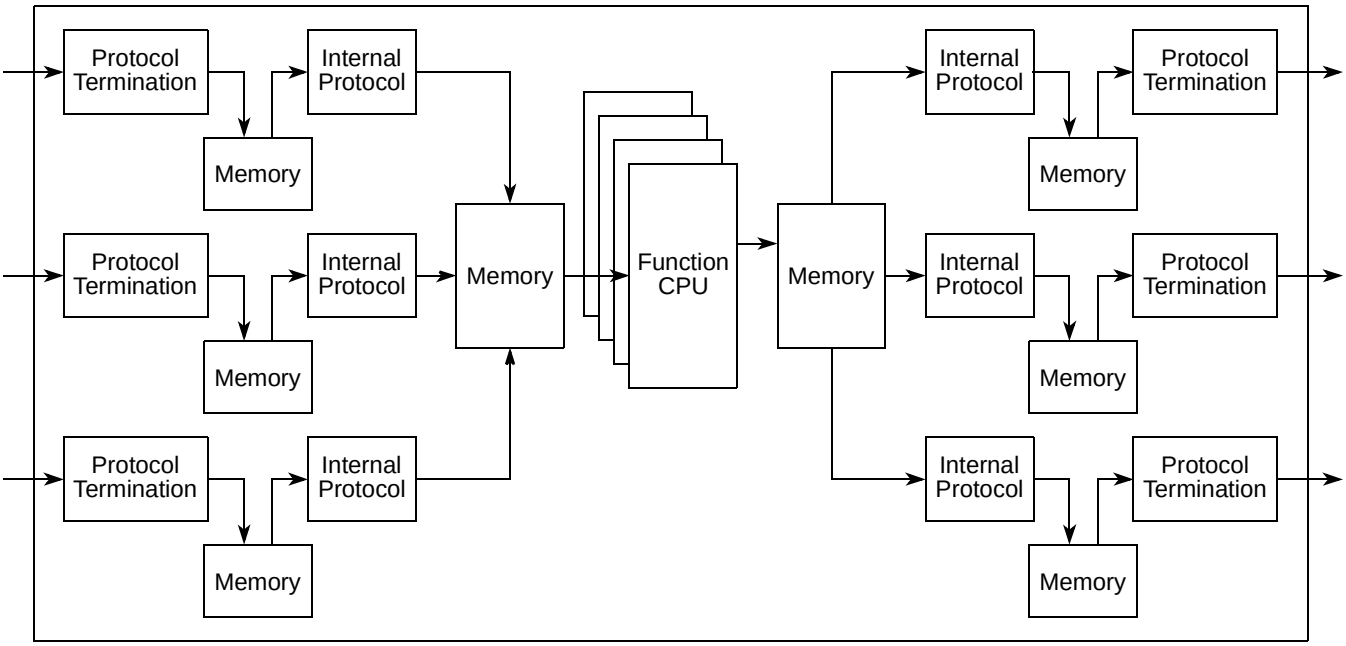

Figure 1. Generic Embedded Communications System

At each step in the pipeline from entry to exit an interconnect protocol is required. As systems grow more complex, the variety of interconnects grows as well, requiring increased engineering costs to design and support them. As Moore's Law increases the speed and complexity of functional blocks, interconnects can become a bottleneck, especially interconnects between ICs and boards. Clearly, increasing interconnect performance is critical to increasing system performance.

While interconnects have a direct impact on achievable system performance, cost often determines the performance and feature set delivered in a product. Reducing the cost of interconnects at a given performance point brings an opportunity either to reduce product cost or invest the savings by increasing performance or adding new features. Industry standards open the door to lower cost in several ways. Standards enable multi-vendor interoperability and thus increase competition. They also allow amortization of engineering costs such as intellectual property (IP) development across a larger ecosystem. When a standardized technology becomes widely used, there is greater potential for lowering costs. An interconnect that can replace IC connections with a system-level backplane logically has the potential to lower costs.

As Figure 1 suggests, a variety of interconnects and protocols often exist in embedded systems. Reducing the number of different interconnect types reduces development and recurring product costs as well. This reduction requires an interconnect flexible enough for wide use in a system.

As Figure 2 shows, interconnects can be classified into four applications depending on their position in the architectural hierarchy:

- IC-to-IC
- Board-to-board
- Chassis-to-chassis
- Local area network

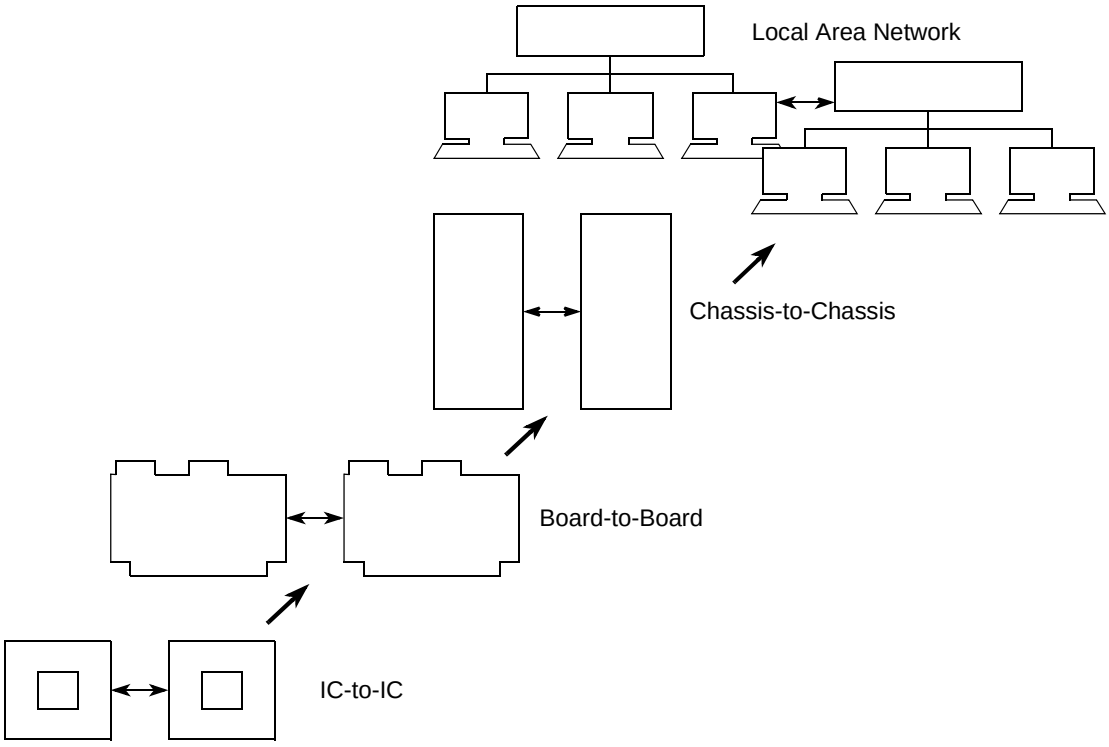

Figure 2. Interconnect Applications

Many proprietary interconnects are optimized for just one of these applications. Using them elsewhere can often incur significant performance and product costs. For example, using a processor front-side bus originally intended to interconnect a CPU, main memory, and I/O system as a system-wide interconnect can be challenging at best and impossible at worst.

A number of standard interconnects have been developed. Early arrivals included 10/100 Mbps Ethernet, TDM, and VME. More recently, standards such as CSIX, HyperTransport, Infiniband, PCI Express, RapidIO, and SPI4-2 have garnered industry support. Two considerations for gauging the ability of any interconnect to meet cost and performance goals as a system fabric are silicon-support ecosystem and technical viability. At first, the interconnect with the largest ecosystem may seem most likely to meet these goals. After all, a large ecosystem should ensure that the technology road map is broad and well invested. Using this measure, the best choice often appears to be Ethernet. For many system applications, Ethernet is indeed a suitable choice. However, the limitations of Ethernet make it challenging to meet certain system-level requirements. For some applications, RapidIO technology is a better solution. A variety of technical advantages contribute to the growing support of RapidIO technology in system applications. This document reviews system fabric requirements and then compares how well Ethernet and RapidIO technology can meet these requirements.

2 System-Level Fabric Requirements

A system interconnect necessarily engages a broader set of requirements than smaller subsystems. Reviewing common system fabric requirements can help us to sort out the suitability of Ethernet or RapidIO as a system fabric.

2.1 Data Plane Versus Control Plane

Most networking and communications systems comprise two often separate system architectures and fabrics: the data plane and control plane. The data plane moves traffic through high bandwidth paths from a remote source to a remote destination. Generally, data plane traffic is not consumed by the system. Often, the data plane must carry this traffic with some degree of Quality of Service (QoS). In the most demanding setting, a previously agreed minimum bandwidth must be guaranteed for some or all traffic.

The obvious measure and requirement on a data plane interconnect is sustained bandwidth. Also of importance is the ability to minimize packet loss, latency jitter, and in some cases end-to-end latency. Achieving high bandwidth in a carrier-grade data plane application often implies the ability to control congestion and partition available bandwidth on a fine-grained basis.

The control plane carries traffic that is generated and consumed locally in the system. Often, this traffic is used to initialize the system and manage run-time errors or to supervise system operation. In some applications, it also directs data plane traffic flow or carries unusual data traffic. Control plane traffic is processor-oriented. Dedicated compute resources communicate with each other while performing complicated system-level control and management algorithms. Not surprisingly, many control plane interconnects derive from processor front-side or I/O buses designed to carrying a combination of processor, memory, and I/O traffic.

The most obvious control plane interconnect requirement is low latency. Completing control algorithms promptly requires system reliability, performance, and predictability. Minimizing latency requires low communication overhead and careful protocol design. Because the control plane is processor-centric, usage paradigms such as address-based read/write and messaging are vital to minimize processor overhead.

Another common requirement is guaranteed delivery. Unlike data plane traffic that can tolerate occasional packet loss, control plane traffic must have robust error detection and hardware correction to reduce the impact of soft errors.

Many systems use dedicated fabrics to move control and data plane traffic. Ideally, to simplify the system and reduce costs, traffic for both would traverse the same interconnect. However, such use can affect system reliability because a failure of both control and data paths affects the ability of the system to diagnose and work around the failure. When reliability and cost requires a combined fabric, a combination of protocol features for both control and data plane traffic is a must.

2.2 Layered Specification Architecture

Some types of embedded equipment have extended 5–10 year product lifetimes. To allow subsystem upgrades and derivative products, a system interconnect should meet evolving requirements over multiple generations of products.

Over time, a hierarchical layered protocol became generally accepted for the sake of long-term flexibility. In this approach, the protocol is partitioned hierarchically as shown in Figure 3. Each layer presents an abstracted set of services upward to higher layers. Therefore, for example, the physical layer can be updated without entailing higher-level hardware and software changes.

The most well-known example of layered hierarchical partitioning is the OSI reference model. Many protocols use variations of this model. In the case of Ethernet and RapidIO technology, each partitions the layers somewhat differently. To discuss them using common terms, this application note defines three general layers that map to one or more layers in the OSI stack as shown in Figure 3.

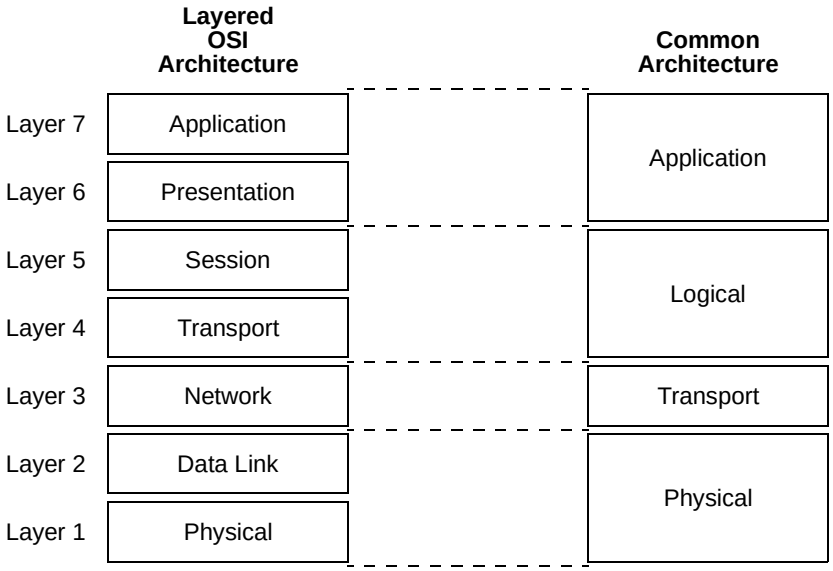

Figure 3. Architectural Layers

Upper-layer hardware or software applications invoke logical-layer operations. The transport layer then sets header fields necessary to route resulting packets to the intended destination. Finally, the physical layer adds fields required by the link protocol to move the packet reliably from one side of the link to the other.

Protocol extensibility is not only layered and hierarchical; it requires “future proofing” so that new features can be added without breaking backward compatibility. Defining spare bits has long been a common solution. However, more than one protocol has been prevented from using spare bits when a specification does not require all devices to set reserved bits in the header to a known state and then to ignore them when interpreting the header. This simple requirement allows new devices to anticipate the state of reserved bits generated by early silicon and act in a backward-compatible way.

2.3 Logical Layer Requirements

The logical layer defines services that move data from one device or endpoint to another. Control and data plane applications use the services appropriate to their tasks. For example, control plane applications often depend on guaranteed delivery.

Data plane applications do not always require guaranteed delivery and may tolerate loss of some packets. Common data plane services include:

- *Data streaming*. Sometimes called datagram services. Conceptually, datagrams convey an ordered stream of data moving from some source to a destination endpoint FIFO. Similar to messaging, datagrams carry a limited amount of data and do not have an associated address.
- *Remote DMA*. Moves data between two endpoints using read or write semantics. It is optimized to move large blocks of data by moving data directly from the source to the destination without the need to copy data from an inbound buffer to the final location in memory
- *Protocol and payload encapsulation*. Allows external protocols and payloads to be carried across an interconnect, often transparently so that the external protocol is unaware of the existence of the intervening interconnect.
- *Traffic management*. Allows entry and exit points in the network to communicate in order to allow negotiation of fabric services such as bandwidth and QoS.

Common control plane services include:

- *Read and write*. Allows an exchange of data between endpoints. The address of the data to exchange is carried from source to destination.
- *Messaging*. Similar to a write, allows a data payload to be sent from source to destination. Unlike writes, messages do not have an associated address but rely on the destination endpoint to define where in memory to place the associated data.
- *Atomics and other distributed computing functions*. Support semaphore services that coordinate distributed processing. Additional services keep caches and other processor-related state coherent across a fabric.
- *Multicasting*. Can be very useful for system management functions or distributed computing settings. The duplication of data can occur in endpoints or switches

2.3.1 Transaction Ordering Policy

System-wide ordering policy is determined by application-level requirements and affects both performance and compatibility with existing software. Strict ordering is compatible with most software mechanisms. However, as ordering rules become weaker, more opportunities exist to reorder traffic and increase performance.

Most software executes sequentially line-by-line in a single control thread in which the effects of one line are assumed to be complete before the next line of code executes. A read operation on one line is expected to reflect updates from writes in previous lines of code. From a system fabric perspective, a read request transaction cannot pass earlier writes but instead must push them ahead.

Other software paradigms involving multiple entities, such as the producer-consumer mode, assume certain system-level ordering conventions. To coordinate the movement of data between two masters, the producer-consumer algorithm expects the producer to update a buffer before it notifies the consumer. Remember, from a system fabric perspective, a read response (producer notification) cannot pass producer writes (buffer data) but instead must push write transactions ahead.

When specific ordering rules are imposed on a system and fabric, how and where they are enforced is an important consideration when dealing with protocol complexity and performance.

2.4 Transport Layer Requirements

The transport layer defines how destination endpoints are addressed and packets are routed from source to destination across the fabric. Requirements for a system fabric using a point-to-point interconnect include:

- Addresses a large number of nodes
- Destination-based routing
- Simplified switches

The transport layer for a chassis-sized system fabric must support up to thousands of nodes. Given the trade-off between header overhead and addressable space, support for more nodes is not a requirement.

Two common routing mechanisms are destination routing and path-based routing. Destination routing implies lookup tables at switching elements. Path-based routing carries the desired path for the packet in the header. As a result, all endpoints must know the system topology. Therefore, path-based routing cannot contain runtime topology changes to limited regions of the network. For example, a hot-insertion or extraction in a destination routed system would require changes to state in adjacent switches rather than every endpoint in the system.

To connect more than two devices, shared bus interconnects require little more than additional copper on the board. Duplicating this connectivity in a point-to-point interconnect requires a low-cost switch device. Reducing switch complexity and therefore cost is a priority.

The interconnect protocol can have a strong impact on switch complexity. For example, complexity increases with a need for complex header parsing or the need to modify each packet as it passes through a switch.

Some types of traffic can tolerate occasional packet loss. For example, real-time data such as voice and video can suffer some data loss without becoming bothersome at the application level. Other types of traffic have great difficulty tolerating loss. For example, loss of a control plane message often means long upper-layer timeouts that can introduce unacceptable control-loop latencies. Unlike predictable and structured user data streams, it can be difficult to determine how to recover promptly after an error when the control algorithm is distributed across multiple processors. A system interconnect that carries both data and control plane traffic must accommodate both lossy and lossless transport. Ideally, correcting for packet loss can be done in hardware rather than in software stacks, which may inject considerable latency when correcting errors.

2.5 Physical Layer Requirements

The physical layer of an interconnect defines how data payloads are transported from one side of a link to another. This definition includes the link protocol as well as the signaling used on the electrical channel.

The length of the physical channel has a strong influence on complexity and cost. Sending a gigabit per second bit stream over hundreds of meters of unshielded cable requires a very different solution than sending that same bit stream across a one meter backplane. For most networking and communications equipment, the physical channel length ranges from tens of centimeters between chips to between 80 and 100 centimeters across a standard rack-mounted backplane. To minimize costs, it is desirable to achieve these distances using standard FR-4 circuit board material. For some applications, the need to span chassis assemblies within a rack makes a cable channel of up to 10 meters desirable.

To minimize the number of signals, there is a serialized embedded clock signaling scheme that uses a single differential pair in each direction. To scale bandwidth without increasing the data rate per pair, multiple single pairs or “lanes” can be used together as a single logical link. The routability of a cost-sensitive backplane imposes a practical limit of about four lanes between cards. Therefore, single and four lane support is a requirement. However, some board-level chip-to-chip applications can use sixteen lanes or more per link for maximum bandwidth. Support for two, eight, and sixteen lanes offers added flexibility. The ability to operate when only some of the lanes are connected is also highly desirable.

A range of supported lane data rates allows the required throughput to be closely matched to minimize power and cost. In practice, the ability to make a relatively small down shift in data rate late in development can sometimes allow a system to ship even though the physical channel does not have the anticipated design margin.

As increasing data rates stretch the limits of electrical signaling technology, maintaining current bit error rates becomes increasingly difficult. Hardware support for link-level error correction becomes a requirement when guaranteed delivery service with QoS is needed.

Some physical installations must connect equipment across distances longer than copper-based interconnects can accommodate at a given data rate. Because optical fiber is a common physical medium for traversing hundreds or even thousands of meters or more, the ability of an interconnect to accommodate fiber-based physical layers is desirable.

2.6 Quality of Service

Quality-of-Service (QoS) requirements are becoming increasingly important in networking and computing equipment. As multimedia services proliferate, new service quality requirements are imposed on system fabrics. For example, real-time video and audio data streams require a minimum guaranteed bandwidth with low end-to-end latency and latency jitter. Real-time interactive streams may also have specific worst-case end-to-end latency requirements. In general, QoS can be measured in terms of throughput, end-to-end latency, and latency jitter. To meet specific requirements based on these metrics, a fabric protocol must be able to:

- Identify individual traffic streams between two endpoints.
- Provide differentiated levels of service to individual streams.
- Control congestion caused by streams as they cross the fabric.

The ability to identify and differentiate traffic in a network is basic to QoS. It is always possible to identify traffic streams based on source and destination endpoint identifiers to allow differentiated service between multiple endpoint pairs. However, most system fabrics must accommodate endpoints that aggregate multiple flows of traffic and therefore need to identify specific flows between any endpoint pair.

After classification of individual streams is accomplished, fabric components must be able to deliver varying levels of service quality based on packet classification.

Delivering minimum guaranteed bandwidth means switches must be able to reserve bandwidth over links for specific classes or streams. This implies that the fabric must have effective flow control mechanisms to combat congestion. The existence of a bandwidth guarantee for one stream implies that other streams have the potential to back up. Congestion control then becomes crucial to delivering guaranteed bandwidth. Controlling latency also requires close control over congestion events because they cause

packets both individually and collectively to experience increased latency as they pass through the network.

Ideally, congestion could be eliminated by injecting much less data than the fabric can carry. For this reason, some systems rely on over-provisioning the system fabric to meet required QoS. Unfortunately, congestion can still develop in over-provisioned fabrics when multiple traffic flows converge on a single link or if head-of-line blocking occurs in a fabric implementation.

As link speeds increase, the reliability of individual links is not expected to increase. Theoretical bit error rates of 10^{-12} to 10^{-15} are common today with system-level error rates measured in seconds to hours depending on the number of links in the system. How quickly the fabric detects and recovers from errors can contribute to latency jitter and affect QoS.

Effective system fabrics need robust flow control to minimize latency and achieve high fabric usage. Because congestion is a dynamic phenomenon lasting from nanoseconds to milliseconds, a well-behaved system interconnect must have a hierarchy of flow control methods. The flow control loop latency between the detection of congestion and the traffic source must be short enough relative to the event itself to allow adequate control. Short-term congestion requires very tight flow control loops. The longer the duration of the congestion event, the broader in scope the flow control mechanism must be.

Avoiding short and mid-term congestion events requires moderately broad flow control methods ranging from link level to end-to-end across the fabric. Prevention of longer-term congestion requires that ingress and egress negotiate input and output bandwidth to avoid swamping the fabric. As congestion and head-of-line blocking occur, individual link usage is negatively affected. Unless limited by flow control mechanisms, system throughput can collapse as congestion works its way backward and encompasses large portions of the system fabric.

The highest end-to-end latencies occur when links saturate, leading to full hardware queues. The most effective flow control mechanisms anticipate congestion by watching queue depth and link usage and then proactively controlling traffic flow at entry points. The least effective mechanisms are invoked only after congestion negatively affects network performance.

2.7 High Availability

In many embedded applications, the ability to continue operation even in the presence of failures (also known as high availability) is a key customer requirement. Following are the fundamental attributes of any highly available system fabric:

- No single-point failure mechanism with multiple management hosts and redundant endpoints and links
- Robust fault detection at the link level and end-to-end
- Fault recovery with error logging and notification; robust fault isolation and containment
- Hot swap of field replaceable units (FRU)

Highly available system fabrics have no single point of failure and must therefore include redundancy, with management and transport mechanisms. It must be possible to create redundant endpoints and links in either hot or cold standby modes of operation.

Highly available fabrics must detect and recover from faults ranging from soft errors that occur during normal operation to more catastrophic and rare hard failures. Because soft errors should be the most common error event, highly available fabrics must be able to correct and recover from soft errors.

All layers of the protocol must contribute to fault detection. Link-level error checking and correction should exist. Errors in data payloads and control protocol must be detectable. To recover from lost response transactions, end-to-end transaction timeouts must exist at the logical layer.

To aid in fault recovery, there must be a method to identify failures, log them, and notify a management entity (often a host endpoint somewhere in the fabric). Minimally disruptive debug capabilities should also exist to aid in error diagnosis and recovery. Recovery from hard failures in a system is complicated because simultaneous activities are often affected and must be notified to take recovery action. Minimizing the number of activities affected is crucial to prompt recovery. System fabrics should be able to isolate faults and prevent errant hardware and software from “babbling” into the system by sending transactions indiscriminately through the network and disrupting functional system activities.

When a hard component failure occurs, the fabric must allow hot extraction of the FRU containing the failed component and hot insertion of a new one. Hot swapping at the system level requires the following capabilities:

- Detection and notification of the host manager when a link failure occurs
- Containment of the error from the rest of the system
  - Prevention of a failed component from babbling into the system fabric
  - Prevention of fabric congestion when data cannot be delivered to the failed component
- Removal of pending transactions to the failed links or component
- Hot removal and reinsertion of an endpoint or switch
- Reinitialization of the link between the system and the newly inserted device
- Reconfiguration of the newly inserted device

Support for hot swapping assumes the existence of system managers that can notify devices communicating with or through a faulty device to stop communication and clean up pending transactions. Managers can use either in or out-of-band mechanisms to carry these commands. Ideally, a system fabric allows in-band fault management even in the presence of system faults because it can reduce system maintenance cost. In all cases, a system manager is involved to coordinate recovery and must itself be redundant in the most aggressive high-availability systems.

3 Ethernet Technical Overview

It is doubtful that the originators of Ethernet could have envisaged how widespread their technology would become. As services layered on top of Ethernet become the common language for the Internet, Ethernet is now ubiquitous as both a wide and local area computer interconnect. Also, it has evolved into multiple technical meanings. Ethernet was originally developed by Xerox. Later, Digital Equipment Corporation, Intel, and Xerox released the “DIX” or Ethernet I standard followed later by Ethernet II. Independently, the IEEE® organization created the 802.3™ specification that also came to be called Ethernet. The IEEE slightly modified the DIX header format in the 802.3 standard to allow coexistence of both standards in a network.

The original deployment model connected a set of closely located computers by a single shared coaxial cable. Taps were attached to the coaxial cable to form drops to each networked computer. As the physical interface (PHY) technology evolved, the physical layer became a point-to-point connection that could be up to 100 meters long. As a result, the deployment model transitioned to point-to-point connections between desktop computers and nearby closets of Ethernet switches or routers configured in a star configuration as shown in Figure 4.

Since its introduction in 1976, the Ethernet PHY layer has been enhanced to support higher and higher data transfer rates. Today, the industry is in a general transition from 10/100 Mbps to 1000 Mbps (1 Gbps) and even 10 Gbps systems beginning to appear in the market. Data is exchanged through packets produced and consumed by endpoints (called “stations” by the IEEE specification). Individual data packets consist of a header and a user payload of between 46 and 1500 bytes. Today, both Ethernet II and IEEE 802.3 header formats are used. The Ethernet II header is the most efficient for carrying commonly used upper-layer protocols such as TCP/IP. As a result, most embedded applications use the Ethernet II format. In the discussion that follows, the Ethernet II format is assumed.

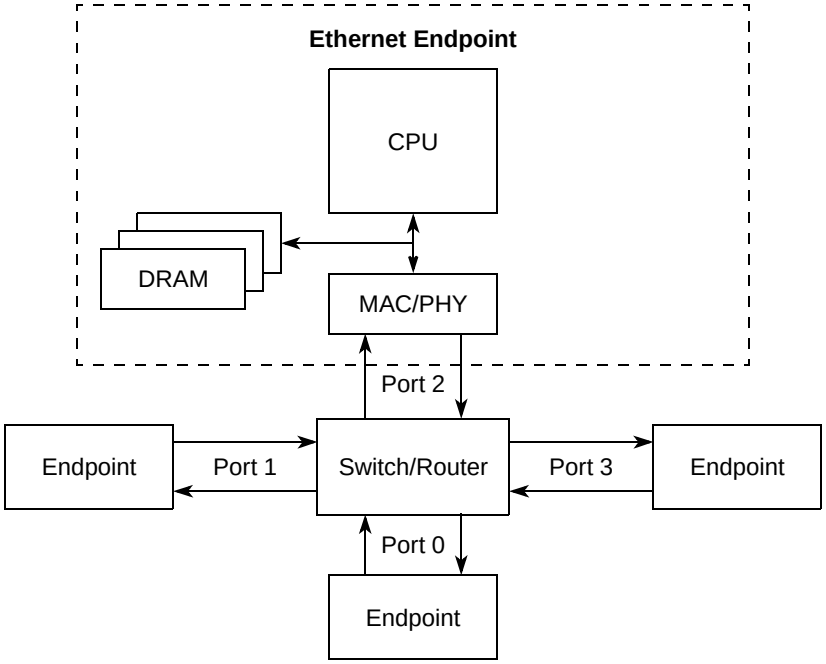

Figure 4. Ethernet Network

3.1 Original Ethernet Design Goals

The key to Ethernet longevity lies in the set of initial design goals that, in retrospect, seem calculated to guarantee a long-term future:

- Ability to connect many computers
- Simple and inexpensive hardware and software at the endpoints
- Flexible and extensible architecture

Because Ethernet was originally intended to connect computer workstations, the assumption was that a processor at each endpoint could participate in running the protocol. This bias was reinforced by the

limited hardware capabilities of the 1970s, which motivated the trade-off between hardware and software strongly in favor of software. With a limited link-level protocol, the required hardware was relatively simple. This arrangement left a relatively large software stack, especially for upper-layer services. This trade-off imposes a growing performance bottleneck as PHY technology increases in performance.

Perhaps the single most important factor in its longevity is the layered hierarchical architecture of Ethernet, which allows new PHY technology to be introduced without substantial changes to the services on top. Backward compatibility of basic protocol formats was a constant requirement throughout the various PHY generations from the initial common coax cable running at 10 Mbps to today's 1 Gbps and 10 Gbps data rates.

3.2 Ethernet Protocol Overview

Application data is encapsulated in packets passed between endpoints. Packets are transmitted with intervening InterFrame Gaps (IFG). A packet consists of a preamble and a frame composed of a string of bit fields starting with the destination MAC address and ending with the frame check sequence (FCS). The entire format is shown in Figure 5.

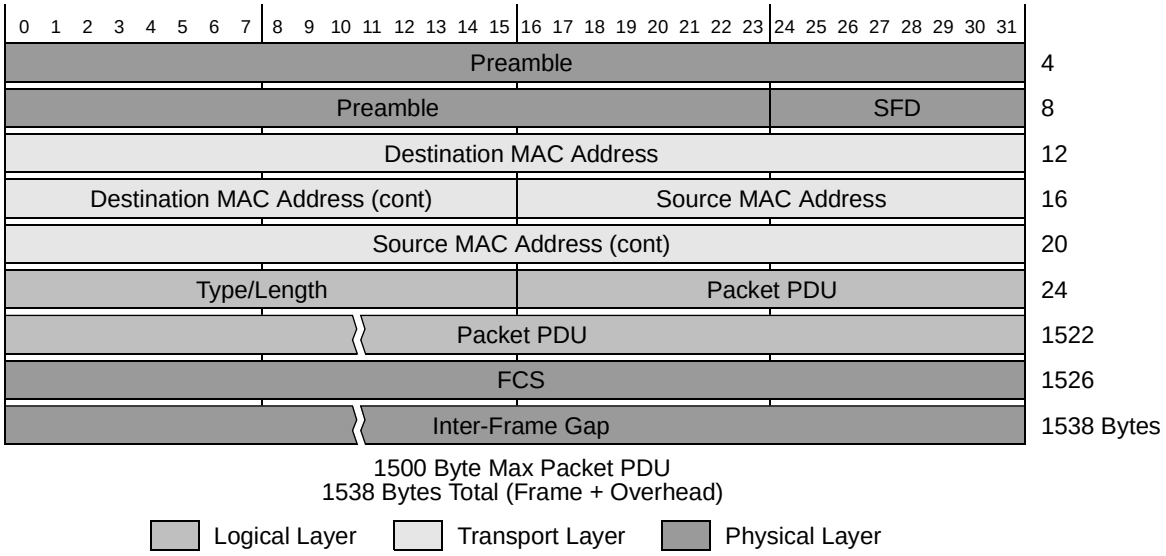

Figure 5. Layer 2 Ethernet Packet Format

The preamble and IFG fields are remnants of the original half-duplex shared coax PHY. The 8-byte preamble carries a specific bit pattern originally used to allow receivers to synchronize their clock recovery circuits. The IFG accommodated the turn-around time necessary for a coax transceiver to shift from transmit to receive and to complete necessary book keeping. The size of this gap is 96 bits or 12 bytes. While more recently-defined PHYs have less need for these fields, they remain for backward compatibility.

The frame header shown in Figure 5 begins with a 48-bit destination and source MAC address. These fields are used to direct packets to their destination and to direct responses back to the source. After the address fields is a Type/Length field. The definition of this field depends on whether the packet uses the IEEE 802.3 or Ethernet II format. The IEEE 802.3 field defines the length of the packet between 64 and 1500 bytes. The Ethernet II field defines the type of payload such as Internet protocol (IP) and use field

values larger than 1535. The two definitions do not overlap and can coexist in a network. The payload is between 46 and 1500 bytes long. A packet has a minimum size of 64 bytes and must be padded when the actual payload is less than 46 bytes. For error protection, a frame check sequence is appended to the user data. This cyclic redundancy checksum (CRC) value is calculated over the frame, starting with the destination MAC address. There is no required order of delivery or completion in Ethernet. Packets can arrive at the destination out-of-order with respect to the transmission order at the source. If ordering is required, layer 3+ must provide this service.

3.2.1 Logical Layer

The Ethernet specification defines up to layer 2 in the OSI stack. The only Ethernet semantic is the unreliable datagram by which user-defined data payloads are streamed across the interface. Packets can be dropped in transit for a variety of reasons. Payloads have no associated memory address.

Because Ethernet implements much of its protocol in software, data movement is orchestrated by a processor that interacts directly with the Ethernet interface hardware. Typically, the processor manages a descriptor-based DMA engine that moves packets in and out of memory. The processor queues up packets or frames that the DMA engine later transfers through the Ethernet interface to a destination.

Applications that need more sophisticated services must layer protocols on top of layer 2. For example, a protocol that performs commanded reads must be implemented before data can be read from a remote endpoint. Many protocols have been created on top of the base layer 2 Ethernet MAC. Some application-level protocols of interest here are listed in Table 1.

Table 1. Ethernet Protocols

Protocol	OSI Layer	Service	Added Services
Sockets	5+	Connection-oriented Reliable	Software access to TCP/IP services
RDMA (iWarp, iSER)	5+	Connection-oriented Reliable	Minimized interrupt and copy overhead at the receiver
TCP	4	Connection-oriented Reliable	Reliable transport Application ports Ordering
UDP	4	Connectionless Unreliable Best Effort	Payload check summing Application ports
IP	3	Connectionless Unreliable Best Effort	SAR Type of service
ARP	2	Routing support for IP	Associates IP with MAC address
MAC	2	Connectionless Unreliable Best Effort	Base specification

The IP allows inter-networking across heterogeneous network protocols by defining a new universal endpoint addressing scheme. In addition, IP defines segmentation and reassembly (SAR) support for user payloads up to 64 Kbytes. Use of IP with user datagram protocol (UDP) or transmission control protocol

(TCP) has become nearly synonymous with Ethernet. Many embedded Ethernet applications use TCP because software support is widely available and understood.

TCP over IP (TCP/IP) presents an end-to-end connection-oriented reliable service to an application. In delivering reliable service, it must compensate for frame loss, packet duplication, high latency, out-of-order delivery, and loss of connectivity. As illustrated in Figure 6, TCP/IP adds significantly to the overall Ethernet header. UDP can be used when this overhead is too high and reliable delivery is either unnecessary or implemented in a proprietary way. UDP is often used when applications need access to services not accessible through TCP/IP, such as multicast.

Remote DMA (RDMA) is a relatively new protocol to lower the overhead required at the receiver. RDMA enables memory-mapped transactions that can place data directly into the destination buffer and thus minimize unnecessary copies and associated interrupts. This service is becoming increasingly important for applications that stream data across the Ethernet and directly into application buffers. Applications include compute clustering and storage. Many network-independent applications use standard library calls such as sockets to abstract away much of the underlying network complexity.

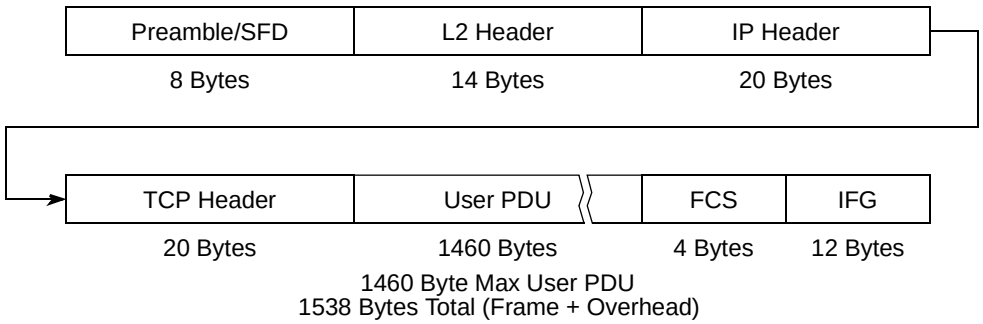

Figure 6. TCP/IP Header Format

3.2.2 Transport Layer

The transport layer defines how packets move from source to destination endpoint in a network. Ethernet uses destination-based routing. At layer 2, addressing is based on MAC addresses. When IP at layer 3 is used, IP addresses also contribute. Layer 2 networks typically consist of endpoints connected by layer 2 switches. Each endpoint is preassigned a unique MAC address. Packets carry a destination MAC address set by the originating endpoint based on its knowledge of the network. Switches use this field to route packets to the proper destination port using internal routing tables. Because switches do not produce or consume packets, switches are transparent to the network.

Layer 3 IP networks introduce a switching device called a router as shown in Figure 7. Routers are similar to switches but they route packets according to the packet layer 3 IP address. Unlike switches, routers are not transparent to the network. Each port of a router appears to the connected layer 2 network as an endpoint and therefore has a unique associated MAC address.

IP networks replace one or more switches with routers that segment the network into multiple layer 2 sub-networks connected by routers. All packets that traverse a router must carry an IP header that includes a source and destination IP address. Every endpoint keeps a table that associates network IP addresses with a MAC address. The IP address of destinations within the endpoint layer 2 sub-network are associated with

their assigned MAC address. All other endpoints, which by definition are outside the local sub-network, are assigned the MAC address of the router port connecting the sub-network to the rest of the network.

In a layer 2 network, the MAC address is end-to-end and remains unchanged as a packet traverses the network. In a layer 3 network, the source and destination IP address fields of a packet are end-to-end and remain unchanged as it traverses the network. The MAC addresses of each packet are modified by routers as it traverses the network. Routers contain lookup tables that associate the destination IP address with an output port and a MAC address of an endpoint within the subnetwork connected to that port. Before the packet leaves the router, the router replaces the source MAC address with that of the output port and the destination MAC address with the value associated with the destination IP address of the packet. When the destination is not located within the immediate layer 2 subnetwork, this MAC address is set to another router leading to the final destination.

Many backplane-oriented embedded systems use Ethernet as a fabric to implement layer 2 switch fabrics. However, these fabrics commonly carry packets with IP or some other optimized proprietary layer 3+ protocol. These protocols are often introduced when the fabric must carry layer 3+ traffic to and from external networks.

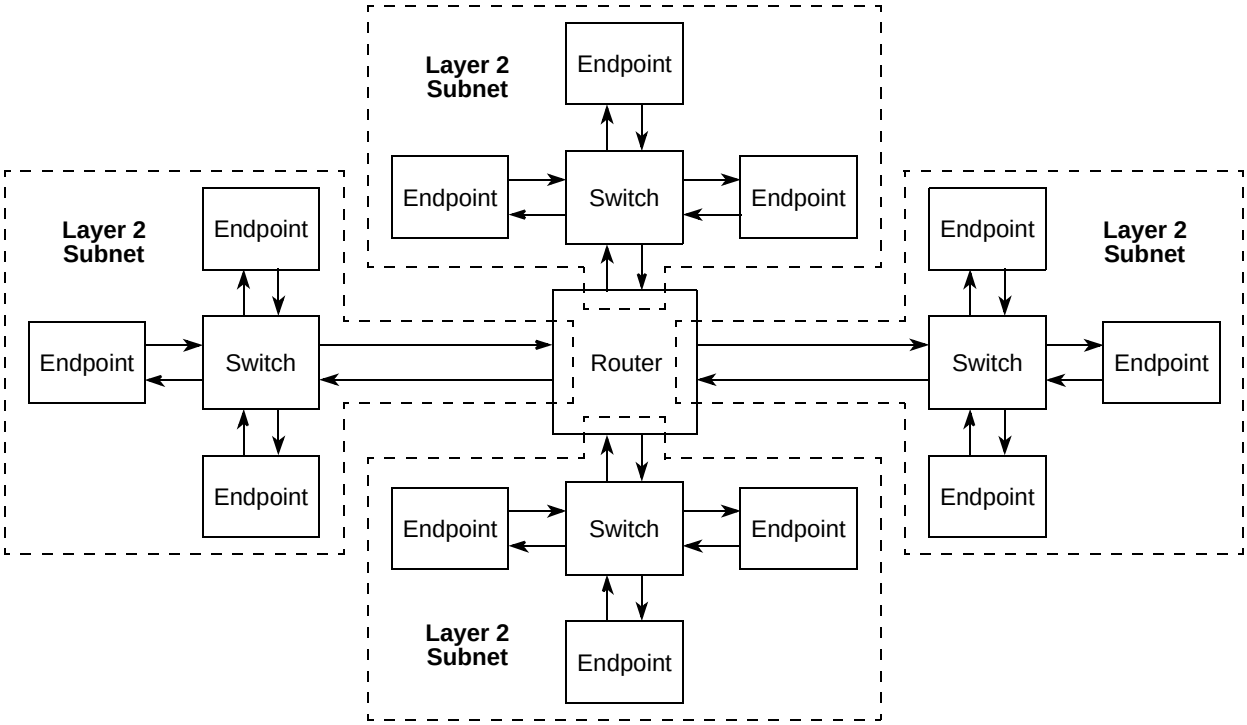

Figure 7. IP Network

The original Ethernet physical layer was a shared cable using the carrier sense multiple access with collision detection (CSMA/CD) protocol to arbitrate use of the cable. Collisions were an expected part of the arbitration process and they corrupted packet data. The packet was retried up to 16 times. However, packets were still allowed to be dropped to simplify the Ethernet protocol and transfer the recovery burden to higher layers. Today, full duplex point-to-point topologies defined in the 802.3x specification have eliminated collisions. However, packets are still dropped because of soft errors or short term congestion

events that exceed available buffering. No link layer protocol exists to recover a dropped packet. End-to-end protocols such as TCP must provide a reliable delivery service to application layers.

3.2.3 Physical Layer

The physical layer consists of the link protocol and the electrical interface between devices. The link protocol moves packets from one side of the link to the other. An Ethernet link itself has a very light protocol. Packets are streamed from transmitter to receiver. The transmitter is free to send packets as fast as possible. With the introduction of the PAUSE frame in the IEEE 802.3x specifications came a flow control method to allow a receiver to slow down the rate of packet transmission (see Section 3.4, “Flow Control”).

At the receiver, the preamble field is used to find the beginning of a frame. The receiver checks the frame using the frame check sequence (FCS), an additional checksum used to verify frame integrity, and then examines the header and data payload as needed for further processing. No acknowledgement exists at this level of the protocol and there is no mechanism for communicating link errors back to the transmitter.

The original Ethernet PHY was a half-duplex protocol over a shared coaxial PHY. Additional PHYs have since been specified, as shown in Table 2. Today, most PHYs define a full-duplex point-to-point topology supported over up to 100 meters of twisted pair cable.

Table 2. Ethernet PHYs

	Aggregate Data Rate	Encoding	Symbol Rate	Number of Differential Channels	Channel
10Base5	10 Mbit/s	Manchester	10 Mbaud	1	Shared Coax
10Base-T	10 Mbit/s	Manchester	10 Mbaud	2	100m Cat3 Cable
100Base-Tx	100 Mbit/s	4B5B + 3-level MLT	125 Mbaud	2	100m Cat5 Cable
1000Base-T	1000 Mbit/s	4D-PAM5	125 Mbaud	4	100m Cat5e Cable
GigE SERDES	1000 Mbit/s	8B10B + 1000Base-CX electricals	1.25 Gbaud	2	Connection to Fiber PHY Backplane FR-4
SGMII	1000 Mbit/s	8B10B + LVDS electricals	1.25 Gbaud	2	Connection between endpoint and PHY or switch

10Base5 is the original 10 Mbps Ethernet I PHY. Fast Ethernet (100 Mbps) introduced three PHYs. The media-independent interface (MII) allows the designer to choose which PHY to use with a given controller. This 4-bit wide parallel interface supports 10 and 100 Mbps speeds. Later, this interface was extended to 8 bits to accommodate Gigabit Ethernet data rates.

Today 10Base-T, 100Base-Tx, and 1000Base-T are the most common PHYs. When separate PHY devices exist, MII or GMII are used to connect them to the Ethernet controller. Because MII and GMII require a relatively large number of pins, various standards have arisen to reduce the pin-count. These reduced interfaces include RMII and RGMII. Another very low pin-count standard now widely used for Gigabit Ethernet is the SERDES-based SGMII. Some Gigabit Ethernet switch vendors support SERDES-based interfaces using either SGMII or IEEE 802.3 1000Base-CX short-haul copper electricals. The main

rationale for these interfaces is to lower pin counts on high-density switch and PHY devices. However, some have leveraged them directly for use in backplane applications.

3.3 Quality of Service

As media services on the Ethernet become more sophisticated, the need for effective Quality of Service (QoS) evolves. Among the standard QoS methods for classifying traffic on a per packet basis are:

- *802.1p+q VLAN tagging*. A layer 2 classification method that adds a 4-byte tag to every frame. As shown in Figure 8, this tag defines a 3-bit priority field for up to eight defined classes of service and a 12-bit VID field to differentiate up to 4096 different streams. Embedded systems frequently use VLAN tagging to differentiate traffic because it is increasingly supported by single-chip layer 2+ switches.
- *Diff-Serv.* A layer 3 classification method that overloads the Type of Service (ToS) field in the IP header and defines up to 64 classes of service.
- *MPLS*. A layer 3 method that involves tagging IP frames at the edge of an MPLS network with a 32-bit value. This tag simplifies routing by representing a specific route through the network. Routers do lookups based on the tag and do not have to parse the header extensively. As the frame leaves an MPLS network, the tag is removed.

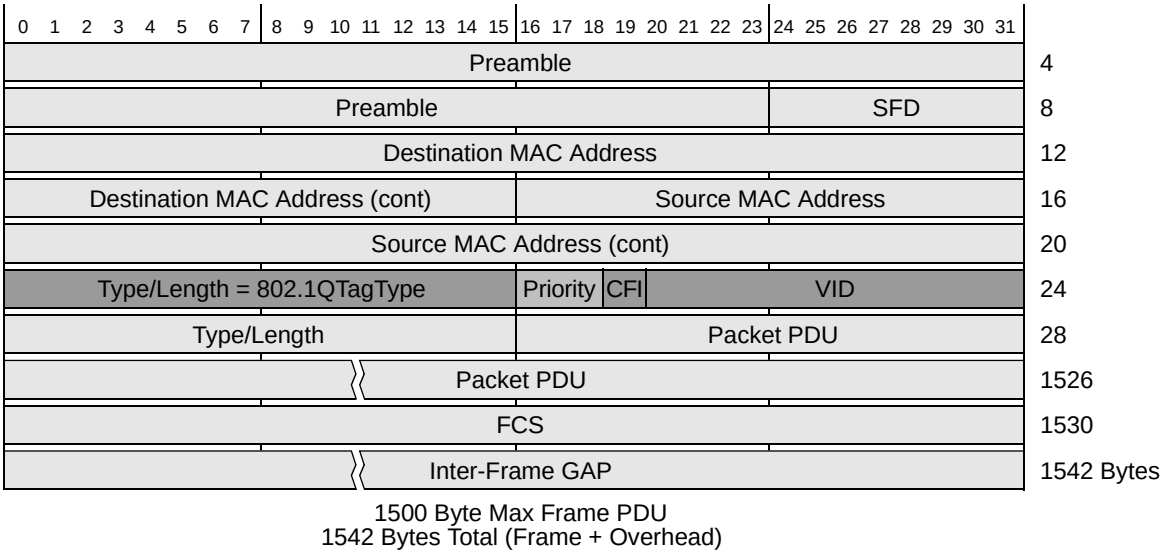

Figure 8. Layer 2 VLAN Tag

3.4 Flow Control

Ethernet is designed for “best effort” service in which packets may be dropped to control congestion or dropped because of errors. As a result, flow control is desirable to control congestion-related packet loss. Initially, flow control in Ethernet existed only as part of layer 3+ services such as TCP. TCP depends on positive acknowledgement messages from the receiver to deliver reliable service. When acknowledgement messages indicate packet loss, it reduces its transmit rate. More recently, Layer 2 MAC control frames were defined in 802.3x to allow the link layers of two connected endpoints to communicate with each other. MAC control frames are normal minimum size 64-byte frames with a unique Type/Length field. The first control frame defines the PAUSE frame that allows an endpoint operating in full-duplex mode to ask

that the opposite endpoint temporarily stop transmitting frames. Any endpoint receiving a PAUSE frame stops sending all frames except a MAC control for up to 64K minimum frame size periods before resuming normal operation. The specification does not require support for PAUSE frames. An endpoint may support reception of PAUSE frames but not be capable of sending them. The Auto Negotiate protocol discussed later allows endpoints to discover whether the opposite endpoint supports PAUSE.

The IEEE 802.1Qau Task Force is defining additional flow control capabilities within a VLAN. This effort targets embedded and backplane applications. It defines a small Ethernet frame that functions as a backward congestion notification (BCN) message for congestion management.

3.5 High Availability

The Ethernet protocol has no inherent single-point failure mechanisms—particularly from the point-of-view of an embedded system. In the wider WAN setting, challenges to high availability include reliability of network management resources. These limited resources must provide naming services and guide routing in a highly dynamic network topology with few network-wide entities overseeing the WAN. Fortunately, embedded systems using Ethernet present a much smaller and more manageable topology and environment.

Ethernet supports redundant paths between any two endpoints either in a hot or cold standby mode. Redundant paths are often used in WAN and LAN networks to increase bandwidth and enhance fault tolerance, so many switches and routers support this capability.

The Ethernet protocol defines the frame check sequence (FCS) at the end of a frame to provide error detection over the entire packet, not including the preamble and IFG. The FCS is checked at each port along the way. If layer 2 switches insert or remove VLAN tags, they must recalculate the FCS in the process. Errors generally cause the packet to be dropped and the error logged using standardized registers. There is no hardware error correction facility. When required, error recovery relies on upper layer end-to-end protocols.

When a fault occurs, in-band notification of management devices is possible if there is sufficient redundancy in the fabric. Some embedded systems use out-of-band mechanisms due to the lossy nature of the transport. For example, the PICMG AdvancedTCA standard defines a management and control interconnect on the backplane in addition to the data plane fabric.

Because MAC addresses are unique to an endpoint, layer 2 Ethernet fabrics require all endpoints in a system to be notified of a change in topology. TCP/IP networks retain a layer of abstraction allowing IP addresses to be retained at endpoints even when the underlying MAC address changes are updated at the routers. Protection from unauthorized and possibly disruptive transactions generated by faulty endpoints is achieved through several mechanisms. Ethernet avoids memory corruption because it has no memory-mapped read and write transactions, and endpoints directly control where inbound datagrams are placed in memory. Because all incoming frames are queued at the receiver, the hardware and network stack must examine each frame and can therefore choose to drop those identified as questionable. In addition, at the transport level VLANs provide some protection by limiting addressability only to a subset of endpoints in the network.

Congestion caused when faulty endpoints inject large amounts of traffic into the network is handled by dropping packets at all points in an Ethernet network. PAUSE frames at the link level and higher level flow control at L3 and above can also be invoked to control this congestion.

Hot extraction and insertion is supported by all Ethernet PHYs using the xBase-T PHYs. The Auto Negotiate protocol first defined with 100Base-T in 802.3u allows identification of endpoints and their capabilities as they enter a network. Some mechanical system standards that support Ethernet fabrics provide an out-of-band management interconnect to inform all endpoints communicating with a faulty device to cease communication and clean up pending transactions.

4 RapidIO Technical Overview

Motorola started work in 1997 on a next-generation front-side bus for its Power- and PowerPC™-based processors. By 1999, Motorola and Mercury Computer had collaborated to complete the initial RapidIO specification. In 2000, these two companies drove the formation of the RapidIO Trade Association, making RapidIO an independent standard. By 2004, the RapidIO Specification had become an international standard as ISO/IEC 18372:2004.

RapidIO technology is a packetized point-to-point interconnect fabric. Packets carry user payloads from 1 to 256 bytes. Both serial and parallel PHYs are defined, allowing effective data rates from 667 Mbps to 30 Gbps. RapidIO networks comprise endpoints and switches as shown in Figure 9. Endpoints source and sink packets. Switches connect more than two devices and move packets from an input to an output port. Many topologies are possible in RapidIO. Large systems often allocate one or more host processors as fabric management devices. At boot up , the host is responsible for initial system discovery and initialization. During runtime, it coordinates and monitors system-level activity and error recovery.

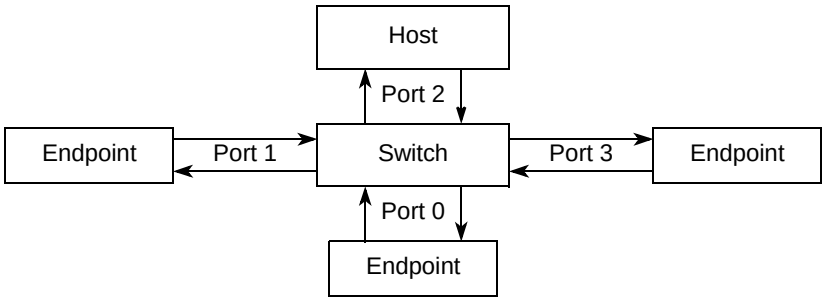

Figure 9. RapidIO Network

4.1 Original Design Goals

RapidIO was originally conceived as a next-generation front-side bus for high-speed embedded processors. The value of a front-side bus that can also function as a system-level interconnect fabric was recognized early in the specification development. The initial design goals for RapidIO technology reflect these early considerations:

- Focus on embedded control plane applications
- Scope limited to in-the-box or chassis applications in the embedded space
- Lower cost and pin count
- Limited software impact
- Simplified switches
- Protocol extensibility

In targeting processor-centric control plane applications first, the RapidIO protocol needed reliable transport with minimal latency. Latency is reduced by minimizing the header field sizes and arranging headers to allow early access to key state. A reliable transport dictates hardware error recovery. By focusing on embedded systems, RapidIO technology avoids backward compatibility requirements that can impose significant limitations and increase complexity. This limit in scope reduces the header overhead by reducing the size of address fields. PHY complexity is reduced by simplifying the required electrical channel. Use of either a source-synchronous parallel or embedded clock serial PHY allows higher bit rates per wire because the data and clock encounter similar propagation delays. These PHYs also reduce the width of the interconnect at a given data rate over earlier parallel shared bus interconnects. To limit the software impact, RapidIO technology retains many usage paradigms supported in bus-based protocols, such as address-based reads and writes, messaging, and doorbell interrupts. By adopting a straightforward destination-based transport model, RapidIO technology reduces switch gate count and die size by minimizing the header fields that must change as packets pass through a switch.

The RapidIO protocol is hierarchical and is designed to be layered as shown on the right in Figure 3. RapidIO headers include spare bits for future protocol extensions. Use of these spare bits is specified to allow backward compatibility in the future.

4.2 Protocol Overview

The layers of the RapidIO protocol work together to allow endpoints in a fabric to communicate. The logical layer is the highest specified layer. It defines operations such as read, write, and messaging, which are implemented using specific transaction types. The transport layer defines how logical layer request transactions are routed to their destination and how responses find their way back to the requestor. The physical layer is the lowest layer of the protocol and specifies the link protocol as well as the electricals used on the physical link. The protocol defines a discrete variable-sized packet carrying header information for each protocol layer and a data payload ranging from 1 to 256 bytes. All packets must be a multiple of 4 bytes long with padding when necessary. Figure 10 shows the packet format for an SWRITE transaction type. The 8-byte packet header is followed by write data of up to 256 bytes. The header is composed of fields defined by the protocol layers.

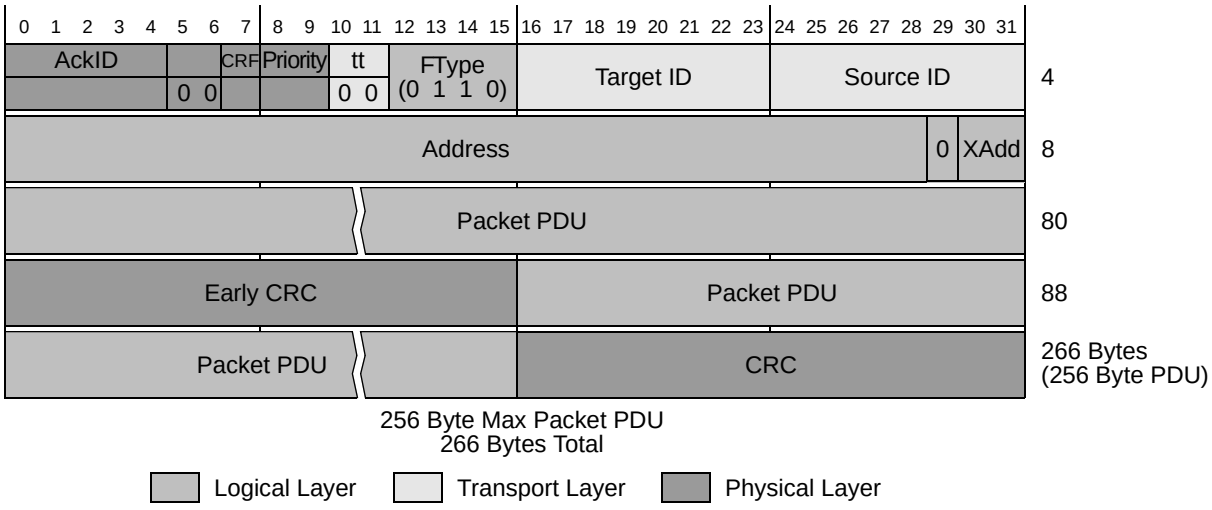

Figure 10. RapidIO Packet Format (SWRITE)

Physical layer fields appear first so that the most immediately relevant fields can be accessed first. The AckID field supports positive acknowledgement of each packet moved across a link. Priority and critical request flow (CRF) bits follow the AckID and are used by the physical layer to enforce ordering, prevent deadlock, and provide QoS features.

Transport layer fields include the tt field, which specifies the size of the TargetID and SourceID fields to define the route request and response transactions across the fabric.

The logical layer fields begin with the FType, which determines the general class of logical layer transaction. Following the transport fields are additional logical layer fields, which in this example include a target write address for the payload. The actual write data follows this address.

All packets end with a 2-byte CRC field for end-to-end error checking. To provide sufficient error coverage for payloads larger than 80 bytes, an early 2-byte CRC field is inserted. Early CRC also simplifies cut-through routing in switches by allowing the integrity of the header fields to be determined before the entire packet is received. Should a defective packet be detected, a mechanism is defined to abort the packet.

4.2.1 Physical Layer

The RapidIO physical layer defines how to move each packet reliably across individual links. Links consist of one or more unidirectional signals in each direction. The physical layer includes the link and PHY layers. Currently, there are two RapidIO physical layers that offer different performance, pin count, and channel (link) length options. Together, they offer effective bandwidths ranging from 667 Mbps to 30 Gbps in each direction, depending on link speed and width.

The RapidIO link protocol guarantees reliable delivery across the link and manages three tasks: link initialization, packet transmission, and error recovery. Several mechanisms are used to implement the protocol.

First, individual packets participate in the link protocol through physical layer packet header fields. The AckID and CRC fields shown in Figure 10 ensure reliable delivery of packets. Packet loss is detected because the receiver must positively acknowledge each packet received. The AckID field allows an acknowledgement to be associated with each packet sent by the transmitter.

Second, a special link layer packet type is defined. Because two independent devices exist on either side of the link, there must be a way to coordinate link operation between them. This is accomplished using a small packet called a control symbol. Control symbols are error protected using parity or CRC protection depending on the physical layer. In addition, the link protocol itself provides error protection. To minimize the loop latency of the link protocol, control symbols can be embedded within packets. Control symbols perform the following basic protocol functions depending on the physical layer used:

- Packet acknowledgement
- Packet delineation (serial PHY only)
- Link maintenance and status
- Flow control
- Idle (parallel PHY only)

Control symbols acknowledge packets by passing the AckID of the packet back to the transmitter. During normal operation, a positive acknowledgement indicates the packet was received correctly. When an error

is detected, an error acknowledgement is sent instead. Except for certain rare error scenarios, an error acknowledgment causes hardware to resend the packet. Therefore, hardware automatically recovers most link errors without software intervention.

Packet AckIDs are numbered sequentially as they are transmitted and must be acknowledged in order as well. As a result, loss or corruption of packets or acknowledgment control symbols can be detected when they arrive with non-sequential AckIDs. Because RapidIO packets are not of fixed size, packet boundaries must be explicitly defined in the protocol. While a dedicated interface signal is used for this purpose in the parallel PHY, control symbols are used in the serial PHY. Control symbols also signal the premature end of a packet under certain error conditions.

Control symbols are also used to initialize a link at power on or after an unrecoverable error event. In addition, they allow status to be reported across the link and recovery initiated when certain control symbol or protocol errors occur.

A RapidIO link is never idle. When no packets are available for transmission, the parallel PHY sends a continuous stream of “idle” control symbols. The serial PHY continuously transmits the IDLE sequence (a specific set of 10-bit characters).

Because control symbols are defined at the physical layer, the specific control symbol format differs between the two defined physical layers. For delimiting, both physical layers implement a 32-bit control symbol. The parallel physical layer defines a single-function control symbol, and the serial physical layer defines a more efficient dual-function symbol that can act, for example, as both a handshake and framing delimiter.There are multiple forms of link-level flow control, including receiver and transmitter-based flow control and in the future a facility to communicate congestion events to upstream queues. In addition, the parallel physical layer defines a transmission throttle mechanism.

4.2.2 Transport Layer

The transport layer defines how endpoints are identified within a RapidIO fabric and how packets are routed to a desired destination. In general, endpoints are the only devices in a RapidIO fabric to source and sink packets. Switches only route packets from an input port to an output port. Each endpoint is assigned a device ID to identify it within the network. Switches have no device ID and are effectively invisible in a RapidIO fabric.

RapidIO is a destination-based interconnect. All packets carry a destination ID corresponding to the device ID of the desired destination endpoint and a source ID corresponding to the device ID of the source of the packet. The source ID is used when a response is required or when errors are reported. RapidIO switches are routing-table based. When a packet arrives at the input port of a switch, the switch examines the destination ID of the packet and looks up the output port associated with that destination in its routing table. The transport layer defines 8-bit (small) and 16-bit (large) device IDs supporting respectively 256 and 64K endpoints in the network. Implementations with large device IDs must also support small device IDs. Systems must choose globally at initialization which transport size to use. Runtime operation of endpoints supporting different device ID sizes within the same fabric is outside the scope of the specification.

Because switches have no device ID, special provision is made in the logical layer maintenance transaction to target switches for discovery, initialization, and configuration. The number of links are counted along a specific path through the network, and this hop count is included in the maintenance transaction header.

As maintenance packets follow the path determined by their destination IDs through the network, each switch decrements the hop count field in the packet by one. When the hop count equals zero upon arrival at a switch, the switch consumes the maintenance transaction.

The specification gives wide latitude to endpoint implementations. A single RapidIO device may contain multiple RapidIO endpoints, each with its own device ID. Such a device may have one or more ports associated with those endpoints. A physical device might also contain one or more endpoints and a switch.

4.2.3 Logical Layer

The logical layer is the highest protocol layer defined by the specification. Applications use operations defined here to move data across the interconnect fabric. The logical layer defines many operations in earlier interconnects plus additional ones helpful to embedded applications. One or more transaction types are used to implement an operation. Collectively they are divided into request and response packet types. Some requests require responses by the destination endpoint while others do not. Operations defined by RapidIO include:

- Read and writes
  - Write with response
  - Streaming write
  - Atomic (read-modify-write)
- Messaging
- Maintenance
- Data streaming
- Globally shared memory
- Flow control
- User-defined operations

4.2.3.1 Reads and Writes

RapidIO defines several forms of read and write operations, including byte-granularity reads and several atomic read-modify-write operations. Also defined are byte-granularity writes, a more efficient 8-byte granularity “streaming” write (format shown in Figure 10), and a write with response operation.

A variety of address sizes for read and write operations allow individual applications to minimize packet header overhead. RapidIO allows a source endpoint to address a total of 34, 50, or 66 bits of address space at the destination endpoint. The address fields are shown in Figure 10.

Reflecting its heritage as a control plane interconnect, RapidIO technology defines a variety of atomic read-modify-write transaction types that support memory semaphores. These include increment, decrement, compare-and-swap, test-and-swap, swap, set and clear. Only swap, compare-and-swap and test-and-swap require responses to the request.

4.2.3.2 Messaging

Messaging is a hardware-based mechanism for sending up to 4-Kbyte payloads to a remote destination. The source does not need to know the destination implementation or memory map. The specification defines the following messaging capabilities:

- Segmentation and reassembly
- Concurrent message transmission
- Multiple message priorities

RapidIO defines packet payloads up to 256 bytes; messages larger than 256 bytes are segmented into multiple message segments and reassembled at the destination even if they arrive out of order. Up to 16 multi-segment messages can be in flight between a source and destination. Each message request segment requires an end-to-end logical acknowledgement from the destination. The destination may respond with retry should hardware be temporarily unable to process the packet.

Figure 11 shows the format of a MESSAGE request packet. No addresses are carried by a message transaction. The location in memory of source and destination data is defined locally to the endpoint.

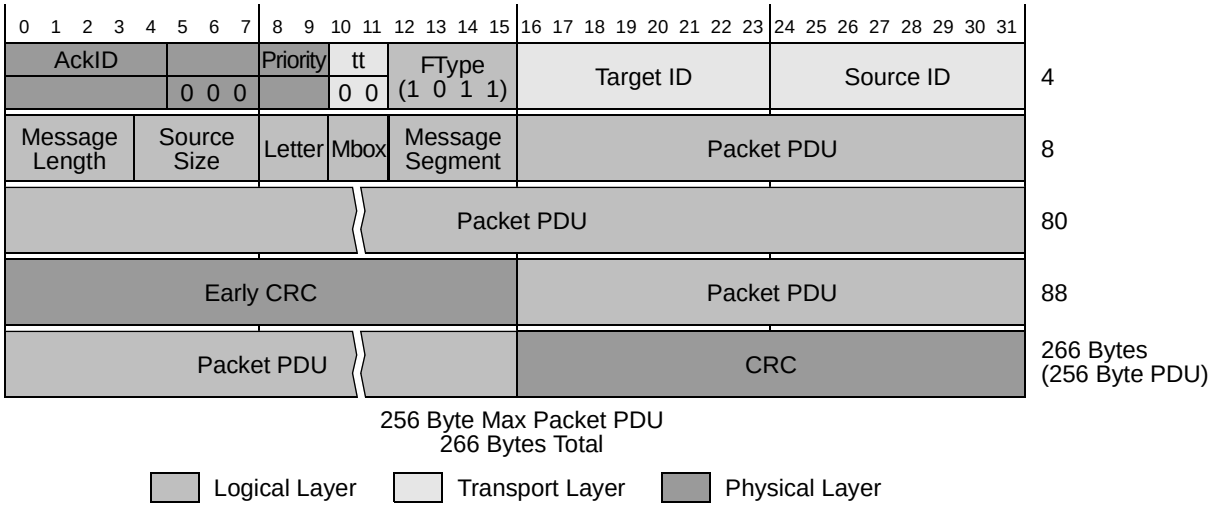

Figure 11. RapidIO Packet Format (MESSAGE)

RapidIO does not define a programming model or specific hardware implementation for messaging. Existing implementations consist of one or more hardware-managed queues at the source and destination. Upper layer services needing to send a message queue one or more messages in memory for transmission. Hardware reads a message from the queue, segments it when necessary, and issues the transactions to the interface for transmission. At the destination, hardware reassembles the segments into a memory-based queue structure and when complete notifies software.

4.2.3.3 Maintenance

As with many other interconnects, RapidIO defines a separate address space for accessing initialization, status, and configuration registers. There is an 8 Mbyte address space with regions allocated for required and implementation-specific registers. Maintenance read and write operations are fully acknowledged and implemented with a single distinct transaction type that supports both requests and associated responses.

This transaction type also implements the maintenance port-write. Because RapidIO switches do not have associated DeviceIDs, they cannot initiate normal transaction types to report errors or status. This transaction type has no response and is not guaranteed to be delivered because it may be dropped by the target endpoint.

4.2.3.4 Data Streaming

Relatively new to the specification are operations targeting data plane applications. Data streaming allows many different protocols to be encapsulated and transported concurrently across a RapidIO fabric. The encapsulation operation itself is protocol-independent and supports up to 64 Kbyte protocol data units (PDUs) using segmentation and reassembly hardware. Hundreds of traffic classes and thousands of streams of data can be identified between a given pair of endpoints so that concurrent transmission of multiple PDUs between endpoints is permitted.

4.2.3.5 Globally Shared Memory

RapidIO defines cache coherence and OS support operations that allow RapidIO to act as a coherent fabric for multiprocessor cache-coherent non-uniform memory access (ccNUMA) systems. These operations allow a processing device with a cache to keep its cache state coherent with others in the fabric. I/O devices can then request data that is coherent with system caches. Other functions allow remote management of processor translation lookaside buffers (TLBs).

4.2.3.6 Logical Level Flow Control

To reduce congestion within a fabric, a flow control facility is defined in the logical layer. This capability allows destination endpoints or fabric switches to shut off traffic from a specific source endpoint. This XON/XOFF messaging protocol applies to traffic within a specific logical flow. Soon, the specification will include additional end-to-end traffic management facilities, such as the ability to control the flow of thousands of individual traffic streams.

4.2.3.7 Logical Layer Ordering

Logical layer operations do not require packets to be transported in order across the fabric for proper operation. Where ordering must be restored at the destination, tagging fields in the packet header are used. However, many higher-level services require ordering, so the logical layer defines prioritized flows. Flows are an ordered sequence of request transactions between a single source and destination endpoint pair. Traffic between unrelated pairs of endpoints is unordered with respect to each other. The logical layer defines a set of ordering rules that are implemented and enforced at the physical layer.

Multiple transaction request flows between a given endpoint pair are defined. These flows are given a FlowID. Flows are labeled alphabetically A, B, C, and so on starting with A. Flows in RapidIO fabric are prioritized with flow A as the lowest priority. As they traverse the fabric, transactions that are part of a higher-priority flow between a pair of endpoints can pass transactions in a lower-priority flow. To avoid deadlock, transactions in a a lower-priority flow must never pass those of a flow with the same or a higher priority. Within a flow, the logical layer specifies the same ordering rules for read and write requests that are common to many applications. Destination endpoints must complete write requests in the same order

they were issued at the logical layer of the source. Read requests must push all earlier writes to completion at the destination endpoint. Responses are not part of any flow and are unordered with respect to requests that did not generate them as well as any other responses. Maintenance transactions exist outside of request flows or responses and are prioritized. Higher-priority maintenance packets can pass lower-priority ones taking the same path through the fabric.

4.3 Priority, Deadlock Avoidance, and Buffer Management

Deadlock avoidance in any system entails avoiding dependency loops. In a RapidIO system, these loops exist when logical layer operations require a response. For example, read transactions introduce the possibility of a dependency loop. When the entire loop between two endpoints is filled with read requests, specific deadlock avoidance measures are needed to ensure that responses to outstanding reads can complete and release their resources.

The logical layer defines prioritized request flows and associated ordering rules. Each packet has a 2-bit physical layer priority field and an optional Critical Request Flow (CRF) bit in the header. By assigning each packet to physical layer priorities, logical flows are identified, ordering rules are enforced, and deadlocks avoided.

Defining priorities at the physical layer greatly simplifies switches because, with the exception of maintenance transactions, there is no need to know the packet transaction type or its interdependency with other packets when making switching decisions. Ordering decisions are based only on the packet source ID, destination ID, and priority bits.

Assigning responses a higher priority than the associated request allows responses to make forward progress in the system. The physical layer requires switches and endpoints to implement mechanisms that allow higher-priority packets (and therefore associated responses) to make forward progress in the system. Table 3 shows the mapping of the four priorities to logical layer flows and their associated requests and responses. Note that the priority of a request packet defines its logical flow. Because each request flow must have its response assigned to a higher priority, up to three logical flows (A, B, and C) can be accommodated.

Table 3. Two-Bit Physical Priority and Logical Flows

Overall Physical Priority / PRIO Field	Flow A	Flow B	Flow C
3	Response	Response	Response
2	Response	Response	Request
1	Response	Request	
0	Request		

When the CRF bit is used, there are four additional physical priorities. These new priorities are defined to allow backward compatibility with systems that do not support this bit. As shown in Table 4, eight priorities accommodate three additional flows for a total of six flows.

Table 4. Two-Bit Physical Priority with CRF and Logical Flows

Overall Physical Priority		Flow A	Flow B	Flow C	Flow D	Flow E	Flow F
CRF Bit	PRIO Field						
1	3	Response	Response	Response	Response	Response	Response
0	3	Response	Response	Response	Response	Response	Response
1	2	Response	Response	Response	Response	Response	Request
0	2	Response	Response	Response	Response	Request	
1	1	Response	Response	Response	Request		
0	1	Response	Response	Request			
1	0	Response	Request				
0	0	Request					

To achieve priority-based deadlock avoidance at the physical layer, we define ordering rules for switch fabrics as they apply to packets travelling between any source and destination pair, as follows:

- Packets with a given priority *must not* pass those of the same priority.
- Packets with a lower priority *must not* pass those of a higher priority.
- Packets with a higher priority *may* pass those of a lower priority (implied by the previous rules).

The RapidIO protocol defines additional switch fabric ordering rules to guarantee that higher-priority packets (and associated response packets) are not blocked by lower-priority packets. Switches must implement buffer and/or reordering schemes that prevent packets of a given priority from blocking the forward progress of higher-priority packets. Therefore, as packets of lower priority congest a system, higher-priority packets must be reordered ahead of those of lower priority as they pass through the switch buffers.

Endpoints must allow responses to make forward progress. To generate responses to requests, endpoints must be aware of the transaction type of the packet. They must prevent inbound packets of a given priority from blocking higher-priority outbound response packets. They must not allow a packet with a given priority to be blocked when lower-priority packets encounter outbound congestion. Various methods are used to meet these requirements, including promoting the priority of outbound responses until outbound forward progress occurs.

In summary, ordering rules require that switches allow responses to make forward progress because their priority is at least one higher than requests at the same flow level. For endpoints, deadlocks are avoided because within each flow, responses must be able to make forward progress through the endpoint even in the presence of requests of the same priority.

Buffer management must coincide with these rules. Switches must manage their buffers to prevent lower-priority packets from filling the buffer and blocking acceptance of higher-priority packets. Prevention can be achieved in various ways, including allocating a minimum of one buffer to each priority. Many interconnect protocols that implement virtual channels require completely separate buffer pools per virtual channel. RapidIO systems can use this scheme as well by dedicating buffers to flows. However, a common solution is to allow a buffer assigned to priority *N* also to hold packets of higher priority. In this

manner, the RapidIO protocol achieves efficient sharing of the total buffer pool across flows by allowing packets of a given priority or higher to use the same buffer pool. Because more than one logical flow shares the same priority, as shown in Table 4, multiple flows share the same buffer pool.

4.4 Serial and Parallel PHY Options

The RapidIO protocol defines two different PHYs: a serialized embedded clock signaling scheme (SerDes)-based interface and a parallel source synchronous interface. Trade-offs between the two include latency, data rate, physical channel length, and number of differential pairs required.

For applications that require lowest pin counts and longer channel distances, the 1x/4x LP-serial specification uses a SerDes based on XAUI electricals from clause 47 of the IEEE 802.3 2002 Ethernet specification. Because of the small number of differential pairs required, this PHY is well suited for backplane fabric applications. In addition to a single differential pair (or lane) per port, the specification also defines use of four lanes per port. RapidIO technology extends the XAUI specification by defining a reduced voltage swing short-haul specification to accommodate a variety of channel lengths without unnecessarily dissipating power or radiating EMI.

The 8/16 LP-LVDS specification uses a parallel source-synchronous signaling method. Data is passed 8 or 16 bits at a time together with a framing bit and clock. The parallel PHY requires more signals than serial, and it is best-suited for board-level designs. Because the serial PHY must serialize and deserialize the data stream, the parallel PHY offers inherently less latency per clock. Therefore, the parallel PHY is ideal for applications, such as a processor front-side bus, that require very high bandwidth and lowest latency.

Table 5 summarizes the available PHY options. Both PHYs define two width options and the protocol auto-detects when a narrow PHY is connected to a wider one to allow interoperability. While the control symbol formats are relatively different between the two physical layers, packet formats are the same except for a few physical layer-specific header bits.

Table 5. RapidIO PHY Options

	Aggregate Data Rate[1]	Encoding	Symbol Rate	Number of Differential Pairs	Channel
8-bit LP-LVDS	4–16 Gbps	Source Synchronous DDR LVDS electricals	4–16 Gbaud	20	~50–80 cm of FR4 2 connectors
16-bit LP-LVDS	8–32 Gbps	Source Synchronous DDR LVDS electricals	8–32 Gbaud	38	~50–80 cm of FR4 2 connectors
1x LP-Serial	1, 2, 2.5 Gbps	8B10B + XAUI electricals (short reach level added)	1.25, 2.5, 3.125 Gbaud	4	~80–100 cm of FR4 2 connectors
4x LP-Serial	4, 8, 10 Gbps	8B10B + XAUI electricals (short reach level added)	1.25, 2.5, 3.125 Gbaud	16	~80–100 cm of FR4 2 connectors

[1]Does not include header and protocol overhead

4.5 Quality of Service

The RapidIO specification classifies traffic into as many as six prioritized logical flows, each of which requires a degree of quality of service. RapidIO provides a variety of QoS capabilities. As noted in Section 4.3, "Priority, Deadlock Avoidance, and Buffer Management," ordering rules at the physical layer require forward progress in the fabric for higher-priority transactions to ensure that responses make forward progress.

The degree to which prioritization results in lower average latency or latency jitter for a given flow is specific to the implementation. Aggressive switch designs make ordering decisions based on the specific flow by examining the priority, source, and destination ID fields. Less aggressive designs might choose to examine only the priority field. QoS is also affected by specific fabric arbitration policies. The specification explicitly defines prioritized flows but leaves implementations to choose arbitration policies that would avoid starvation of lower-priority flows. Often, well-known starvation avoidance schemes such as leaky-bucket are used. Even the least aggressive approach must provide improvement and therefore demonstrate better lower-average latency for higher-priority flows. To provide more aggressive QoS, additional capabilities are being defined, as described in Section 4.7, "New Dataplane Features."

4.6 Flow Control

RapidIO currently defines multiple flow control mechanisms, as follows:

- Physical layer
  - Receiver-controlled
  - Transmitter-controlled
  - Throttle symbols (parallel PHY only)
- Logical layer
  - Flow control packet

Physical layer flow control manages congestion at the link level, addressing short-term congestion events. Receiver and transmitter-controlled flow control are defined in both the serial and the parallel physical layers. With receiver-controlled flow control, the transmitter does not know the state of the receiver buffers. It determines independently when to transmit packets. The receiver determines when packets are accepted or rejected based on availability of its own buffers. Because receiver-controlled flow control causes packets to be resent (that is, retried), it can waste link bandwidth. Physical layer ordering rules require a switch to send available higher-priority packets before retrying the packet associated with the retry (subject to starvation-avoidance arbitration protocols).

Transmitter-based flow control allows the transmitter to decide whether to transmit a packet based on receive buffer status. This mechanism requires periodic receiver buffer status to be sent to the transmitter. Normal control symbol traffic is often used to send the number of available buffers using certain control symbols. The protocol requires a periodic update even if no control symbols are sent. During initialization, the transmitter determines a maximum number of buffers available at the receiver. It generally assigns priority watermarks at various buffer levels and uses these watermarks to determine when it can transfer packets with a given priority.

The parallel PHY defines a third link-level mechanism using a throttle control symbol to control packet transmission rate. Here, the receiver can request that the transmitter insert a selectable number of idle control symbols before resuming transmission of packets.

Logical layer level flow control is provided through a control packet and is used to avoid longer-term congestion. When congestion is detected along a particular flow path—at either a switch or endpoint—the device sends an XOFF message back to the source of the traffic. When congestion subsides, the congested device sends an XON to restart traffic flow.

4.7 New Dataplane Features

The dataplane extensions allow a RapidIO fabric to deliver higher throughput, usage, and QoS through a faster PHY and new protocol features.

Currently supported features in Rev 1.3 of the specification include:

- Protocol encapsulation
  - SAR support for up to 64 Kbyte payloads
  - 256 classes of service and 64 K streams

Dataplane extension features in the upcoming Rev 2.0 of the specification include:

- Next-generation physical layer
  - 5.0 and 6.25 Gbaud PHY
  - Lane widths of 1x, 2x, 4x, 8x, and 16x
  - 8 virtual channels with either reliable or best effort delivery policies
  - Enhanced link-layer flow control
- End-to-end traffic management with up to 16 million unique virtual streams between any two endpoints
  - 256 classes of service and 64 K streams

4.7.1 Encapsulation

Because data plane fabrics often carry other protocols, a vendor-interoperable methodology for encapsulating virtually any protocol using a new data streaming transaction type is defined in Rev 1.3 of the specification. This generic mechanism allows transport of up to 64 Kbyte protocol data units across a RapidIO fabric. A system-defined label is carried with each data unit to identify the protocol transported. Hardware-based SAR support is expected for most implementations.

To provide both higher fabric usage and better QoS, a foundation of specific traffic classification is added for the new data streaming transaction type in Rev 1.3 of the specification. Robust classification allows specific streams of traffic to get differentiated service either from reserved bandwidth, control of average worst-case latency, or latency jitter. Flows are expanded to allow much finer-grained differentiation of traffic between any two endpoints in the fabric. There can be up to 64 K individual streams of traffic within each of 256 traffic classes. Switches and endpoints provide differing classes of service for a stream or group of streams.

4.7.2 Next-Generation PHY

The next-generation PHY operates at 5 and 6.25 Gbaud as part of the dataplane extensions in Rev 2.0 of the specification. These rates correspond to 4 and 5 Gbps throughput (after 8B10B overhead but before accounting for protocol overhead) and allow 10 Gbps data streams to be carried when substantial fabric over-provisioning is desired. In addition, 2x, 8x, and 16x lane widths are defined for both the next-generation 5 and 6.25 Gbaud PHY as well as the existing 1.25, 2.5 and 3.125 Gbaud PHY. The 8x and 16x widths enable much larger bandwidths to be carried over a RapidIO link. Conversely, with support for a 2x width at the higher data rate, fewer links are required to carry the same data rate as 4×3.125 Gbaud.

4.7.2.1 Virtual Channels

Revision 1.3 of the RapidIO specification requires traffic to be transported across a fabric reliably. New extensions add virtual traffic channels that are either “reliable or continuous. Continuous traffic denotes data with real-time characteristics in which it is better to drop individual packets than incur the delay necessary to guarantee delivery. As in the Ethernet “best effort” policies, packets marked as continuous can be dropped if an error occurs or congestion develops. The link-level flow control mechanism is also enhanced to accommodate flow control per virtual channel rather than per link.

4.7.2.2 Link-level Flow Control

The existing serial RapidIO physical layer defines a link-level flow control feature that uses control symbols to communicate receive buffer status back to the transmitter. With Revision 2.0 of the specification, an optional capability enables virtual output queuing (VoQ) by defining how to send per-output port buffer congestion status backwards to upstream queues to allow congested traffic streams to be “sidelined” and thus reduce head-of-line blocking at the receiver. Like receive buffer status, congestion notification uses control symbols.

4.7.3 End-to-End Traffic Management

Congestion control at all layers of the fabric protocol is necessary to increase QoS. Flow control at the physical and logical layer is insufficient to control more than short and mid-term congestion events. The fundamental rate at which traffic is injected into a system at the ingress and egress endpoints must be controlled to manage overall fabric performance. Long-term congestion control requires higher-level end-to-end flow control negotiation. A new traffic management feature layered on top of the new data streaming transaction type adds a new mechanism to the existing RapidIO flow control capabilities by allowing ingress and egress endpoints to manage the amount of traffic injected into the fabric.

Endpoints can use simple traffic management approaches. More complicated network processor-style devices would use advanced traffic policing and shaping methodologies.

4.8 High Availability

RapidIO technology supports endpoints and redundant links, which can be either active or on standby. Traffic is redirected by changing switch routing tables. Often, these changes are required only at the closest switch. Multiple overlapping layers of error detection include use of error detection codes for both packets

and control symbols, positive acknowledgement of packets, and the protocol itself. All single-bit and some multiple-bit link errors are detected through CRC and parity mechanisms.

An end-to-end CRC is included in all packets and is calculated at the originating endpoint. At a minimum, packets are checked for errors at every link receiver. If a packet checks correctly at the destination endpoint, no intervening corruption occurred. All packets end with the 16-bit CRC field shown in Figure 10. In addition, an early 16-bit CRC field is present for non-maintenance packets with a length exceeding 80 bytes. This early CRC enhances error coverage for larger payloads and allows logic to check the header fields before the entire packet is received, which can simplify switch implementations that support cut-through. The CRC covers all but the first six bits of a packet, thus allowing the link-specific AckID field to be reassigned without recalculating the CRC as it traverses links in the network. The only exception is the maintenance packet CRC because the hop_count field must decrement as it passes through each switch. Protection for the uncovered AckID bits is provided by the protocol because the AckID value must be sequential over successive packets.

Control symbols are 32 bits long. Serial RapidIO control symbols are checked using a 5-bit CRC field. Control symbols in the parallel physical layer by append the inversion of the first two bytes, thus providing parity protection.

Single-bit errors in packets are the most common errors by many orders of magnitude and are corrected by retransmission of the packet. Errors in control symbols are recovered through a variety of protocol mechanisms depending on the affected bits. In all cases, errors are logged through standard registers, time-out timers, and thresholds. Should an individual link fail when a serial RapidIO port is running with four lanes, the port automatically falls back to a single lane to allow continued operation.

The specification does not define protection mechanisms to prevent corruption from babbling endpoints. Read and write transactions are especially vulnerable. However, their impact is often minimized when endpoints implement external-to-internal address mapping mechanisms to limit the size of the endpoint memory map visible to an inbound transaction. Datagram-style transactions, such as messages, have inherent protection from corruption because they must be examined by the SAR hardware, software, or both before local applications use them. Congestion caused by faulty endpoints is controllable through the available robust flow control hierarchy.

Many systems use an out-of-band management interconnect so that system managers can notify RapidIO-compliant devices communicating with or through a faulty device that they should stop sending packets and clean up any pending transactions. System managers can be notified of errors in-band due to the reliable transport and robust flow control available. Because switches only source and sink maintenance packets, in-band notification of switch-related errors is supported through a special maintenance packet called a port-write. Switches can use them to notify system managers of error conditions. Port-write transactions are lossy and may be dropped at any point along the path to the endpoint, including at the destination itself.

RapidIO technology supports hot swapping because the protocol and electrical connections support detection of link faults, link shutdown, pending transaction removal, FRU extraction and insertion, and reinitialization of the link and associated device. Many of these steps are standardized through registers specified by the error management specification. However, the exact process required to accomplish these steps is implementation-specific, especially removal of pending operations and initiation of link initialization.

5 Ethernet and RapidIO Comparison

The differences between Ethernet and RapidIO begin with the initial design constraints. Ethernet was initially designed to connect a large number of endpoints over a shared cable PHY with little hardware assistance available at the endpoints. This motivated a simple header with relatively large transport fields and a single transaction type. Hardware needed only to identify packet boundaries, and the rest of the work was left to software.

With a smaller network of tens of thousands of endpoints to address and much less expensive hardware, the size of the RapidIO header can be relatively efficient yet support multiple transaction types. Hardware is assumed to implement the entire protocol. Beyond the protocol, the electrical channel requirements differ significantly. The Copper Ethernet channel demands 100 m reach for the dominant structured wiring deployment model compared to 80–100 cm for RapidIO.

These contrasting constraints have led to profound differences in design complexity, protocol efficiency, and protocol partitioning between hardware and software.

5.1 Layering and Efficiency

Virtually any application layer service can be supported by either Ethernet or RapidIO technology. The difference between the two lies in how the header layers are optimized and the level of hardware support to process that portion of the header. The two interconnects have taken significantly divergent approaches.

Ethernet defines a very generalized header closely following the OSI stack. As shown in Figure 12, the header begins with the MAC layer and grows sequentially through Layer 5+. With a very simple header, the protocol is highly flexible as layers are added. However, even basic higher-level services require additional layered header fields and there is no opportunity to optimize the overall header for these services. The Ethernet specification motivated universal hardware support for a limited amount of header parsing. As discussed in Section 5.8, “Hardware/Software Trade-offs, this resulted in very different performance and usage characteristics as compared to RapidIO.

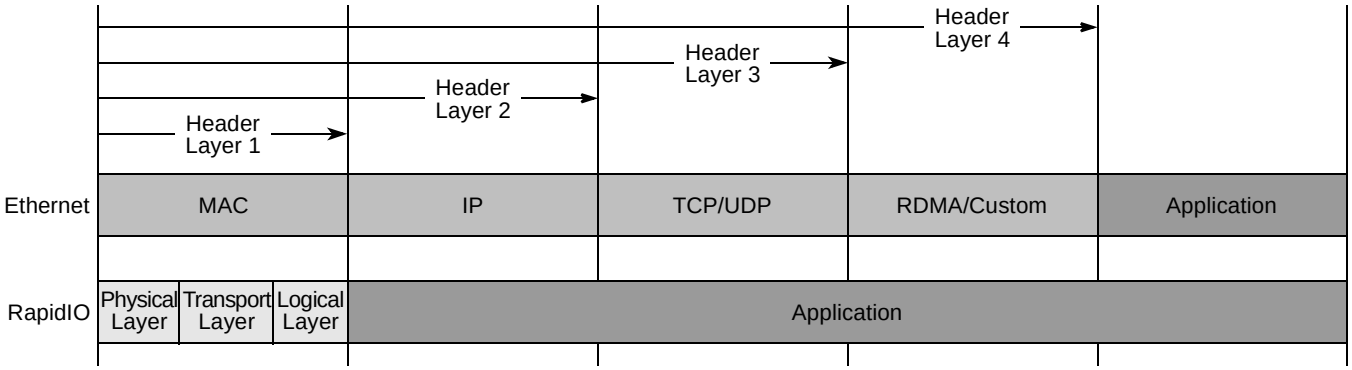

Figure 12. Header Layering

The RapidIO technology focus on header efficiency has encouraged several optimizations. First, the specification itself provides header support for common services that would otherwise require multiple header layers. For example, read and write transaction types are supported by the defined header, thus avoiding the need for additional fields. Second, redundant fields are removed and fields compressed where possible. For example, duplicate addressing schemes are avoided and their addressability limited.

The partitioned and sequential layering of the Ethernet header is immediately evident when additional services such as TCP/IP are supported. In contrast, the RapidIO layered headers are mixed in a way that minimizes the hardware needed to parse them.

Overall, the efficiency of RapidIO is significantly higher than Ethernet as shown in Figure 13. In this graph, efficiency is plotted between a 32-byte user PDU and the maximum packet PDU size of the protocol. For small 32-byte cache-line size packets and below, RapidIO is between two and four times as efficient as Ethernet.

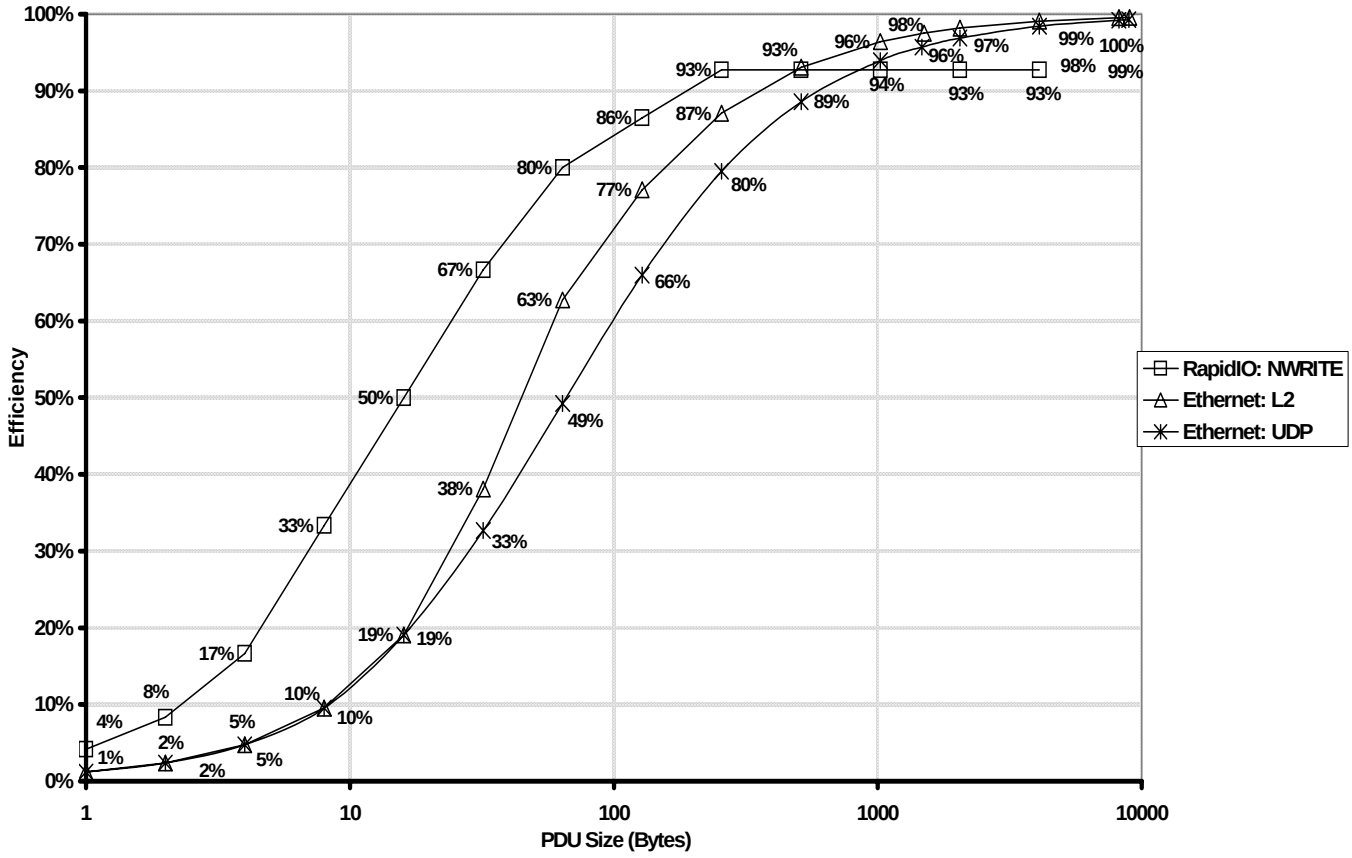

Figure 13. Ethernet and RapidIO Efficiency

Because multiple unoptimized header layers are required for many common Ethernet services, the complexity to parse the headers becomes significant regardless of whether hardware or software does the work. This complexity increases the number of gates required in hardware or the number of cycles required of a processor. Regardless of whether hardware parsing is used, latency increases. The value of an optimized header for all RapidIO application services can be easily seen. The result is a more straightforward and less costly design in comparison to similar Ethernet solutions such as those with hardware off-load features.

5.2 Logical Layer

As described in Section 2.3, “Logical Layer Requirements,” application layers use logical services to move data and control information between endpoints in a network. Table 6 compares these services.

Ethernet supports a significantly larger payload than RapidIO. The RapidIO 256-byte maximum payload size was judged to be a reasonable trade-off between the latency jitter that arises with large payload sizes and the lower efficiency of smaller payloads. As Figure 13 shows, RapidIO achieves well above 90 percent efficiency for 256-byte maximum payload packets while Ethernet efficiency is slightly higher at payloads over about 1000 bytes.

Operations such as read, write, messaging, and datagram are available in both standards. Ethernet RDMA provides read and write operations. However, as a layer 4 protocol, it has relatively high overhead and is not intended for small control-oriented load/store operations. In addition, the write with response and atomic operations available in RapidIO are missing.

TCP/IP services resemble RapidIO messaging. Important differences include payload size and the available granularity of traffic classification. RapidIO messaging supports 4 Kbyte messages while TCP/IP supports 64 Kbytes. TCP/IP classifies the message according to 16-bit input and output port fields where RapidIO messaging defines a 2-bit mailbox and letter designation for full 4 Kbyte messages. Perhaps the most significant difference is that RapidIO messaging is often fully implemented in hardware whereas TCP/IP is often a combination of hardware and software.

Datagram service is directly supported by the RapidIO data streaming transaction. The basic Ethernet layer 2 service is similar to a datagram service except that it is not guaranteed to be ordered. The two services also differ in terms of classification. Ethernet Layer 2 does not define a formal traffic classification field though the 16-bit Type/Length field provides limited classification. A RapidIO data streaming transaction defines a much more extensive classification capability with both an 8-bit class-of-service and a 16-bit StreamID field.

Only RapidIO defines a protocol for keeping caches coherent across the interconnect. With low header efficiency, high latency, and inconsistent levels of hardware support for the protocol, Ethernet cannot function as an effective coherent interconnect.

When read semantics are used in a system, the possibility of deadlock is introduced. Deadlock avoidance is inherent to RapidIO ordering rules. Read semantics in Ethernet exist only at Layer 4+, so upper-layer services must define mechanisms to avoid deadlocks. Unlike RapidIO flows, Ethernet fabrics do not require ordering. Upper-layer services that require ordering and use Ethernet must provide mechanisms to accommodate this behavior.

Table 6. Logical Layer Comparison

	Ethernet		RapidIO
	Layer 2	**Layer 3+**	**Logical Layer**
Payload size (bytes)	46-1500 (802.3) $46\text{-}9192_a$ (Jumbo)	26-1460 (802.3) $26\text{-}9172_b$ (Jumbo)	1–256
Memory-mapped R/W	No	RDMA	Yes
Memory-mapped write with response	No	No	Yes
Memory-mapped atomics	No	No	Yes
Memory-mapped address size (bits)	N/A	64 (RDMA)	34, 50, 66
Messaging	No	TCP	4 Kbyte user payloads, doorbells

Table 6. Logical Layer Comparison (continued)

	Ethernet		RapidIO
	Layer 2	Layer 3+	Logical Layer
Datagrams	Yes	Yes	64 Kbyte User Payloads
Globally Shared Memory Semantics	No	No	Yes
Deadlock Avoidance	N/A	L3+ issue	Pervasive hardware support
[a]Largest common jumbo frame size [b]Example assumes minimum 20-byte IP header [c]Dataplane extensions feature			

5.3 Transport Layer

Table 7 compares the Ethernet and RapidIO transport layers. The most significant difference between them is the number of addressable endpoints. Because Ethernet can accommodate very large networks with many endpoints, the address fields are much larger than those of RapidIO technology. Ethernet has 6 bytes of MAC address and, when IP is used, an additional 4 bytes of IPv4 address. In contrast, RapidIO technology uses one or two bytes of address. This difference accounts for a significant portion of the additional overhead in Ethernet, as illustrated in Figure 13.

Because endpoints and switches can drop packets in Ethernet, only a best effort service is provided. When reliable delivery is required, Ethernet must use higher-layer protocols such as TCP/IP to deal with packet loss. In contrast, RapidIO guarantees delivery by providing end-to-end error checking and retrying link errors. Switches are not allowed to drop packets. New dataplane extensions to the RapidIO specification define virtual channels that, in addition to guaranteed delivery, allow a lossy delivery mechanism.

The number of fields that must be modified as a packet traverses a switch from one port to another directly affects switch complexity and fall-through latency. Ethernet switches do not modify the packet unless they can insert or remove VLAN tags. When IP packets are routed, a number of fields along with the FCS must be updated and the FCS recalculated. RapidIO switches update only the AckID field and do not recalculate CRC because this field is not covered by CRC. The only exception is maintenance packets, where hop count and CRC must be recalculated as they pass through a switch.

Bridging between the Ethernet physical layers is easy because no specific header bits are associated with the physical layer. A packet is unchanged as it passes between a 10BaseT and 100BaseT port on a switch. Because there are header bits specific to the physical layer in RapidIO, bridging packets between the parallel and serial interfaces requires header changes. However, these bits are not covered by CRC and can be changed without recalculating the CRC.

Table 7. Transport Layer Comparison

	Ethernet		RapidIO
	Layer 2	Layer 3+	
Topologies	Any	Any	Any
Delivery service	Best Effort	Guaranteed (TCP, SCTP, others)	Guaranteed Best Effort[a]

Table 7. Transport Layer Comparison (continued)

	Ethernet		RapidIO
	Layer 2	Layer 3+	
Routing	MAC Address	IP Address	Device ID
Maximum number of endpoints	2^{48}	2^{32} (IPv4) 2^{128} (IPv6)	2^{8} (Small) 2^{16} (Large)
Fields that change link-to-link	None	TTL, MAC Addr, FCS	AckID Hop Count, CRC (Maintenance Only)
Redundant link support	Yes	Yes	Yes
[a]Dataplane extensions feature			

5.4 Physical Layer

Different channel requirements cause significant physical layer differences between Ethernet and RapidIO technology, as shown in Table 8. Several issues distinguish the two interconnect technologies:

- Offered data rates
- Number of required signals
- Channel characteristics

Probably the most significant difference between Ethernet and RapidIO is the data rate ranges defined by the standard. Ethernet has evolved at order-of-magnitude rates as it has moved from 10 Mbps to 10 Gbps. RapidIO defines more granularity. The LP-Serial PHY defines data rates at 1, 2, and 2.5 Gbps with rates quadrupled when four lanes are used. The parallel LP-LVDS PHY offers even greater breadth with data rates ranging from 4 to 32 Gbps depending on width and clock rate.This PHY requires a larger number of signals than the serial version but offers reduced latency at a given data rate by avoiding the serialization and deserialization steps necessary for the other PHYs

When accounting for board cost, routing, and device package limitations, there is a practical limit to the number of signals that can be devoted to a backplane interconnect. These limits have favored serialized interconnects. Both Ethernet and RapidIO offer single and four pair serial configurations depending on the PHY. Figure 14 illustrates the effective bandwidth achievable by Gigabit Ethernet and RapidIO assuming a backplane setting on which only four pairs of conductors are practical. RapidIO does not preclude cable or optical channels but in general assumes a maximum backplane or board-level channel of 100 cm using copper traces on FR4-like materials. The common deployment model for Ethernet is between a network closet and the desktop with a twisted-copper pair channel that is 100 times longer than RapidIO. Furthermore, Ethernet cabling assumes bundling with many other similar pairs, resulting in the need to tolerate significantly more cross-talk than the RapidIO board-level channel model. This difference has resulted in significantly higher PHY complexity and has encouraged system designers to consider a more suitable board-level Ethernet PHY standard. Only recently has the IEEE attempted to address these requirements under the IEEE 802.3ap working group that is tasked to define 1 Gbps and 10 Gbps Ethernet over backplanes. Meanwhile, vendors and original equipment manufacturers (OEMs) are adapting other standards. While no clear standard has emerged, one silicon vendor has chosen a single lane of XAUI for their Gigabit Ethernet switches. XAUI for Ethernet outlines a single electrical channel between a 10 Gbps

MAC and an external PHY. The serial RapidIO PHY extends XAUI with a second “short-reach” electrical specification that lowers the maximum transmitter amplitude to lower power and reduced radiated EMI.

Table 8. Physical Layer Comparison

	Ethernet		RapidIO	
	1000Base-T	**SERDES[1]**	**LP-LVDS**	**LP-Serial**
Channel	100 m cat5 cable	~50 cm FR4	~50-80 cm FR4 + 2 connectors	~80-100 cm FR4 + 2 connectors
Data Rate (Per signal pair)	1 Gbps	1 Gbps	500-2000 Mbps	1, 2, 2.5 Gbps
Symbol Rate (Per signal pair)	125 Mbaud	1.25 Gbaud	250-1000 Mbaud	1.25, 2.5, 3.125 Gbaud
Encoding	8 bits ⟶ 4 quinary symbols	8 bits ⟶ 10-bit symbols	DDR	8 bits ⟶ 10-bit symbols
Signalling	Multilevel Signaling (5 Layer PAM Code)	NRZ	NRZ	NRZ
Signal Pairs (Per direction)	4[2]	1	10, 19	1, 2[3], 4, 8[3], 16[3]
Electricals	Custom	XAUI	LVDS	XAUI (with long and short reach)
Clocking	Embedded	Embedded	Source Synchronous	Embedded

[1] This is one approach. Others are using 1000Base-CX electricals over backplanes.
[2] Each pair carries both Tx and Rx.
[3] 2x, 8x and 16x are dataplane extensions features.

5.5 Quality of Service

As previously discussed, achieving QoS metrics requires the fabric to:

- Identify individual traffic streams between two endpoints.
- Provide differentiated service to these traffic streams.
- Control congestion caused by traffic streams.

Based on classification of individual streams, fabric components must deliver varying levels of minimum guaranteed bandwidth, worst-case end-to-end latency, and latency jitter using effective flow control to manage congestion. Traffic in a fabric can always be identified on the basis of source and destination (for example, MAC address for Ethernet, DeviceID for RapidIO). The more important question is whether multiple streams can be identified between the same endpoint pair.

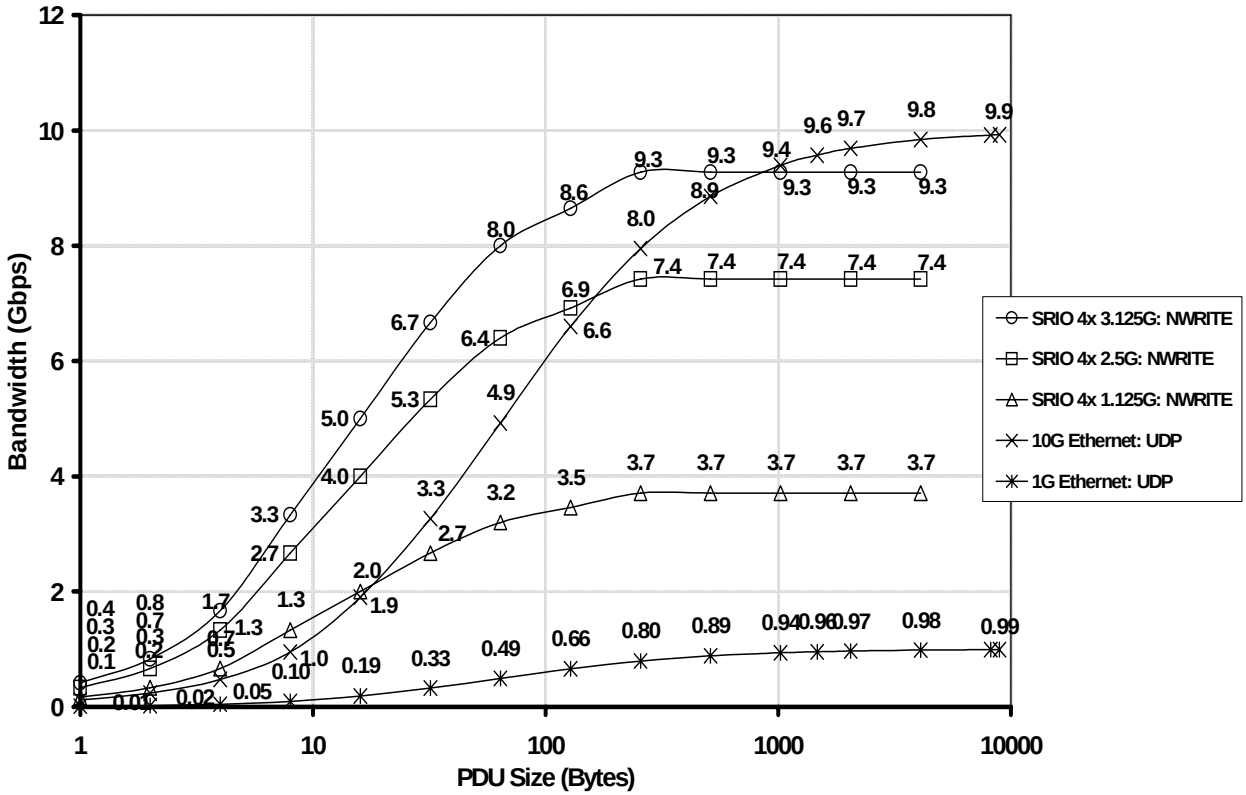

Figure 14. Effective Bandwidth

Table 9 quantifies the number of classes of service and streams supported by the Ethernet and RapidIO. Because these identifiers are carried together, the total number of differentiated streams is equal to ((Class-of-Service) × Streams).

Table 9. Classification Comparison

	Ethernet				RapidIO		
	Layer 2		Layer 3+		Revision 1.3		Dataplane Extensions
	Ethernet	802.1p+q	TCP/IP	TCP/IP + Diff-Serv	Non-Type 9	Type 9 Encapsulation	VC Additions
Classes of Service	1	8 (Priority)	1[1]	64	6 Flows	6 Flow × 256	8 VC × 256
Streams	None	4094[2]	Millions[3]	Millions[3]	None	Millions[4]	Millions[5]

[1] TOS field has not been generally used except when overloaded by Diff-Serv.
[2] VID is 12 bits in size. Two values pre-defined: VID=0 is used and VID=0xFFF is reserved.
[3] Streams are often based on 5-tuples {Src IP Addr, Dest IP Addr, Protocol, Src Port, Dest Port}.
[4] 6 flows × 256 Classes of Service × 65536 StreamIDs.
[5] 8 VCs × 256 Classes of Service × 65536 StreamIDs.

Ethernet IEEE 802.1p+q tagging allows multiple traffic streams to be differentiated between any endpoint pair. The tag contains a 3-bit priority field that can be used to define eight classes per VLAN. The 12-bit VLAN ID (VID) field can be used further to differentiate traffic within a class. For example, a DSLAM is a common traffic aggregation application. Individual DSL lines can be given their own VID, and intervening switches may need to apply differing levels of QoS based on VID. In this manner, Layer 2 switches with VLAN tag support can theoretically support thousands of differentiated streams.

For TCP/IP, layer 3+ routers commonly use a 5-tuple when defining specific traffic streams:

- 32-bit source IP address
- 32-bit destination IP address
- 16-bit source port
- 16-bit destination port
- 8-bit IP protocol

This methodology allows millions of individual streams between two specific TCP/IP endpoints with traffic differentiated by port number and protocol fields. However, no commonly used class of service (CoS) field exists. The type of service (ToS) field in the IP header originally served this purpose, but its meaning early on was not consistently used and the value of the field was significantly reduced. Later, DiffServ was proposed to add a more robust CoS designation by overloading ToS to define 64 classes of service. Use of DiffServ allows further traffic differentiation between endpoint pairs over the same port and protocol.

In the current revision of the specification, RapidIO defines up to six flows that can be considered prioritized classes of service. Each flow can carry a Type 9 encapsulation transaction with an explicit 8-bit CoS and 16-bit StreamID field to add 256 classes with 64 K StreamIDs per class. Therefore, millions of streams can be differentiated. In addition, new virtual channels are defined by the dataplane extensions that further multiply the total number of streams, as shown in Table 9.

All RapidIO networks must provide a minimum level of prioritized service because logical layer ordering and deadlock avoidance is implemented with physical layer priorities. Average latency improves because packets marked with a higher relative priority must make forward progress because they might be responses. Optionally, switches can examine packet source and destination ID to identify the flow of a buffered packet. The switch is then free to reorder packets from different flows and offer head-of-line blocking avoidance and other QoS features.

5.6 Flow Control

Flow control is key to managing short, medium, and long-term congestion events in a system. Table 10 summarizes the available flow control features of Ethernet and RapidIO.

Ethernet has few flow control mechanisms. As a best effort service, it commonly manages congestion by dropping packets. Unfortunately, packet loss introduces a number of QoS issues, not the least of which is latency jitter. Flow control is the task of upper-layer protocols. For example, TCP reduces its transmit rate when packet loss occurs. Because detection of packet loss implies high latencies, TCP flow control effectiveness is longer-term. Another weakness of TCP flow control is that it cannot proactively manage congestion to prevent packet loss. Therefore, when TCP flow control is invoked, latency jitter has already occurred in the system. Some systems manage long-term congestion more effectively with traffic

managers that negotiate the amount of traffic passing between the ingress and egress points and then shape the traffic accordingly.

Because Ethernet lacked short-term link-level flow control, endpoints were forced to grow their internal receive buffers to avoid overruns before traffic could be turned off through higher-level protocols. More recently. the IEEE 802.3x specification introduced the PAUSE frame to add a link-level flow control mechanism. One advantage of this mechanism is that it can be proactive and prevent packet loss rather than react to it. Unfortunately, it is sometimes implemented by software and as a result has limited short-term effectiveness. Without a full hardware-based mechanism, the PAUSE frame cannot address the shortest link-level congestion events and only reduces the size of these larger receive buffers.

No standard Ethernet flow control protocol effectively addresses mid-term duration congestion events. However, there is an effort under the IEEE 802.1Qau Task Force to define a backward congestion notification facility for VLAN-based systems.

As shown in Table 10, RapidIO technology defines multiple hierarchical flow control mechanisms. All endpoints must implement one or both of the link flow control mechanisms. Endpoints optionally support Type 7 and 9 mechanisms. Switches can support Type 7 flow control. With the exception of link-level retry, all flow control mechanism are proactive and allow congestion to be managed before there is a significant impact on network performance.

Table 10. Congestion Control Feature Comparison

Congestion Scale	Ethernet		RapidIO
	Layer 2	Layer3+	
Short-term (Microseconds)	• PAUSE frame	None	• Transmitter and receiver-based flow control • Per-VC transmitter-based flow control • VoQ Flow Control
Mid-term (Milliseconds)	• Backward congestion notification?[1]	None	• Type 7 flow control • VoQ flow control
Long-term (Seconds)	• Backward congestion notification?[1]	• TCP/IP Flow Control, Traffic Management	• Type 9 traffic management[2]

[1] Under development by IEEE 802.1Qau Task Force.
[2] Dataplane extensions feature.

5.7 High Availability

An important requirement of backplane-oriented embedded systems is that both RapidIO and Ethernet must have facilities to deal with soft and hard link errors, system-level protection from errant endpoints, and support for link redundancy.

5.7.1 Soft and Hard Link Errors

Ethernet and RapidIO differ in the way link errors are handled. Ethernet has no link-level protocol for error recovery, and networks must drop packets when faced with errors or local congestion. Because dropped packets are a relatively common event, system-level error recovery protocols are robust and well understood. Unfortunately, error detection and recovery always occur at the system level rather than on the

link level. Because timeouts in Ethernet exist only at layer 3 and above, they are managed by off-load hardware in the best case or by software in the worst case. This implies a much longer timeout, especially when software stacks are involved. This necessarily introduces significant latency jitter.

RapidIO defines a link-level protocol for error recovery and various link-to-link and end-to-end timeout mechanisms. Virtually all errors are recovered at the link level without software intervention. Defined end-to-end timeouts for responses are hardware-based. Link-level timeouts provide much shorter time-scale notification. With a much smaller control loop, the latency jitter is significantly lower.

RapidIO failure rates, defined as undetected packet or control symbol errors, are very low due to the strong protocol support. A failure rate of 0.84 failures in time (FIT) is calculated, assuming a bit-error-rate of 10^{-13} and a 128 lane switch operating at 3.125 GBaud. This figure is significantly less than the hard failure rate of the devices on either end of the link.

For hard link failures, error detection issues are made worse in Ethernet. No standards exist for hardware-based recovery methods such as retries and timeouts. Ethernet drops packets for a significant period of time before the failure is detected. This event is often detected through the exhaustion of jitter buffers or expiration of heart-beat timers. Protocols such as bidirectional forwarding detection (BFD) in Ethernet detect failures with sub-second resolution along the end-to-end path. This mechanism depends on regular exchange of packets and therefore imposes a bandwidth overhead. The speed with which it can detect the failure depends on the rate of packet exchange and therefore invokes a trade-off with bandwidth overhead. In contrast, because RapidIO links carry valid data at all times, a broken link is promptly detected locally at link level.

5.7.2 System-Level Error Protection

An important system-level consideration for high availability is protection of the network from compromised endpoints that babble into the network. Because Ethernet defines only a datagram transaction type and does not have the memory-mapped transactions that are available in RapidIO, it has inherent protection from babbling devices in a fabric. Because the endpoint controls the location where inbound frames are stored internally and usually examines each packet in turn, there is ample opportunity to avoid potential corruption. In addition, because the transport is lossy by nature, dropping questionable packets is an easy solution. Congestion issues arising from babbling are also relatively easy to deal with since they can be dropped anywhere in the network. It is fortunate that this mechanism exists because Ethernet lacks a robust flow control hierarchy to manage it any other way.

RapidIO provides a number of protection mechanisms. For read/write transactions, most endpoint implementations have an address mapping mechanism that limits access to the larger internal address space. For messaging, the protection is the same as in Ethernet because incoming packets are queued to a DMA engine that has local control over the placement of this data in memory. Congestion in a RapidIO network due to babbling can be dealt with using the existing hierarchical flow control mechanisms. Type 7 flow control is particularly useful in limiting the impact of a rush of inbound traffic from a babbler.

5.7.3 Link Redundancy

Ethernet and RapidIO both support link redundancy; redundant links can be added and removed from the network. Multiple paths between endpoints are allowed in Ethernet because ordering is not assured for a stream between endpoints. However when link aggregation is used, packets within a stream must be sent

over the same link to retain ordering. How a stream is defined depends on the implementation. RapidIO endpoints can support multiple links. However, for active links ordering is required, so multiple paths between two endpoints must be avoided. As a result, endpoints with multiple active links must present a different deviceID to the network for each link.

Both Ethernet and RapidIO allow traffic to be balanced across the links as desired. Shutting traffic off a failed link requires only a routing table change.

5.8 Hardware/Software Trade-offs

Very little hardware support is explicitly required by the Ethernet specification. Over time, as data rates increased and hardware became less expensive, the desire to off-load the processor work has grown more urgent. Based on the value of selectable header fields, modern Ethernet controllers perform layer 2, 3, and 4 check summing; coalesce packet-level interrupts; and move traffic to and from multiple queuing structures. More sophisticated controllers off-load nearly all the TCP/IP protocol, including control-oriented connection setup and tear down, segmentation, reassembly, and acknowledgement. Other layer 3+ protocols are accelerated in hardware, such as RDMA and iSCSI.

In contrast, RapidIO devices implement the protocol entirely in hardware. This results in the contrast illustrated in Figure 15, where significant amounts of software are often required to implement most Ethernet services.

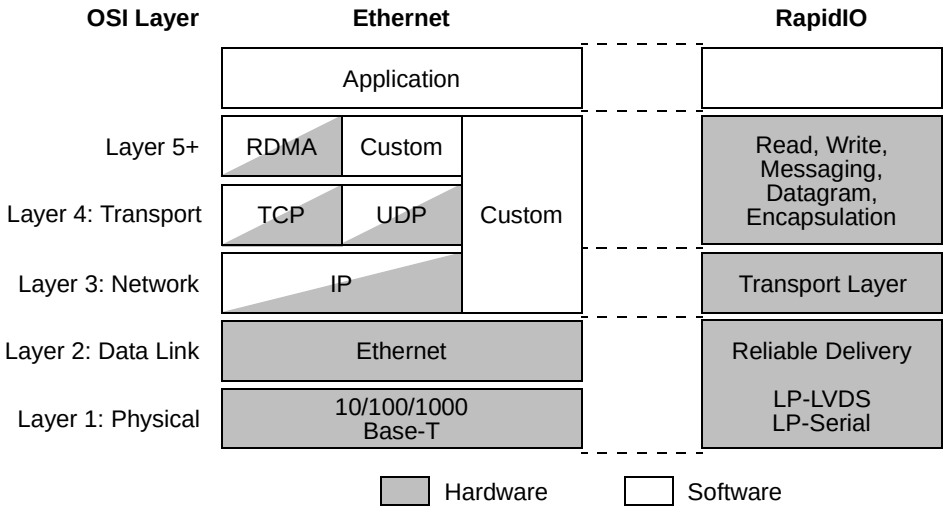

Figure 15. Protocol Layering Comparison

5.9 Software Usage Models

The fundamental task of an interconnect is to move a block of data from a source to a destination device. Historically, upper-layer applications use two common hardware paradigms to move data.

The first paradigm uses memory-mapped reads and writes. A specific address in the source system address space is used as a portal to the destination and, when written, data is automatically sent to the destination. Often, a range of addresses is defined and a processor or other hardware functional block targets read/write transactions to one or more addresses in this range. In most cases, the interconnect carries the transaction to the destination where the address can again be remapped to the local address space of the endpoint. Early

generations of interconnects provided only this address-mapped capability. Often, processors executed load and store instructions to this address space to move data to the destination.

To move ever larger amounts of data across the interconnect, a second common paradigm arose to off-load data movement from the processor. The processor programs a hardware DMA engine to read data from one location and write it to another. When the desired transfer completes, the DMA engine notifies the processor. A basic DMA engine imposes no structure on the data moved. This task is left to higher layers of the system-level hardware and software protocols. To further off-load work from the processor, DMA engines have been created that are aware of the structure of the data.

Both of these hardware paradigms work with Ethernet and RapidIO but with differing levels of hardware support for the protocols. As a result, significantly different levels of processor overhead are often required. The widespread and early adoption of Ethernet in UNIX systems has led to multiple APIs that applications use to transport data between processes and across networks. The most common is socket services. The sockets API was initially defined to sit on top of TCP/IP services. Because of its widespread use, the sockets API has been ported to a variety of other underlying transports, including RapidIO.

UNIX **ioctl** commands are sometimes used to enhance sockets by adding features. Also, the overhead inherent to TCP/IP has motivated alternative APIs that are in some cases vendor-specific and in others OS independent. Examples include the message passing facilities of the QNX OS from QNX Software Systems and the transparent inter-process protocol (TIPC).

5.9.1 Ethernet

Most Ethernet MAC implementations use a DMA engine to move data to and from the Ethernet port. Often, the engine is commanded by chains of descriptor commands set up by the processor in local memory. When additional application services are needed, new protocol layers are built on top of this basic capability. Because the Ethernet specification does not specify the user-visible DMA interface, there is no standard and each implementation requires a vendor-specific stack.

Many custom protocol stacks on top of Ethernet layer 2 have been written to deliver custom services to the application or to be more efficient than standard IP-based protocols. For example, layer 2 assumes a datagram use and therefore lacks direct support for address-based reads and writes. When this usage model is desired, layer 4+ protocols such as RDMA are used. The cost of supporting custom stacks across multiple vendors or even multiple generations of the same vendor hardware can become prohibitive. Therefore, many systems rely on standard layer 2, UDP, and TCP/IP stacks to preserve application software although the lower layers of the stack that communicate with the hardware are still vendor-specific.

5.9.1.1 Layer 2 MAC

Applications that need to minimize processor and hardware overhead and can tolerate the lossy nature of layer 2 Ethernet services can use basic MAC functionality. Data plane applications carrying real-time data that can tolerate packet loss are obvious candidates for this service. Many Ethernet MAC implementations offer some QoS by directly supporting hardware insertion and deletion of VLAN tags and multiple transmit and receive queues based on configurable header state such as VLAN tags.

5.9.1.2 UDP

Applications that need port-based multiplexing, multicasting, or end-to-end error checking in addition to IP services and can tolerate a lossy, connectionless service between endpoints can use UDP. Because it lacks guaranteed delivery and does not need to track and recover from lost packets, it has lower software overhead and stack latency. Some data plane applications use UDP instead of a custom stack on top of layer 2 when extra services are required and performance is not bottlenecked by the processor overhead of the UDP/IP stack.

5.9.1.3 TCP/IP

Applications that need reliable service are candidates for TCP/IP. Modest control plane applications that can tolerate high transport latencies can leverage this protocol. Without adequate hardware off-load, the lack of sufficient processor performance to run the stack can sometimes limit TCP/IP throughput.

5.9.2 RapidIO Technology

Unlike Ethernet, RapidIO directly supports address-based reads and writes. Like Ethernet, many RapidIO implementations include a DMA engine for use in general data transport and messaging. In the near future, streaming and encapsulation are also likely to use a DMA engine for the basic transport mechanism. RapidIO does not yet define a RapidIO-specific application layer API. However, Linux implementations of the sockets API use RapidIO messaging as an underlying transport. In the future, RapidIO-specific services such as atomic operations, messaging interrupts, QoS, and encapsulation features will likely be supported in sockets using IOCTLs or through some OS-specific and agnostic APIs.

5.9.2.1 Writes, Reads, and DMA

Most RapidIO endpoint implementations support reads and writes to and from remote devices over RapidIO. DMA engines within the device are often used for large block moves. Many systems have multiple interconnects between devices. For example, a control or maintenance interconnect may exist in addition to the data plane. Data is moved to a destination through DMA block writes over the data plane interconnect. When the DMA transfer completes, notification that the data is ready for use is sometimes sent over the second control interconnect. In this scenario, there is a race condition between the commitment at the destination of the last datum on one interconnect and the notification to the destination on the other because no ordering is implied between the two interconnects. To mitigate this issue, RapidIO defines a write with response transaction. The DMA engines of some RapidIO devices use write with response for the last write transaction and do not notify the local processor that the transfer is complete at the destination until this response is returned.

5.9.2.2 Messaging

The usage model for RapidIO messaging is almost always based on a DMA engine to queue up outgoing messages and receive incoming ones. The local processor is notified when packets are sent or received. Because all segments of a message must be positively acknowledged before a new mailbox message is sent, some implementations include multiple mailboxes to increase the messaging throughput by allowing multiple simultaneous messages to be sent. Inbound message hardware often dedicates an inbound DMA

queue to each mailbox. Doorbell messages allow a short message to be sent together with an interrupt. Some implementations use dedicated DMA engines for this function as well.

5.9.2.3 Streaming and Encapsulation

In the future, support for streaming and encapsulation using Type 9 packets is likely to be based on a sophisticated DMA engine that is content aware and capable of moving large amounts of data. This DMA engine would manage dozens or more streams of traffic queued separately. The DMA engine would multiplex and demultiplex the link data stream based on specific configuration and header information. An example encapsulation is a bridge between Ethernet and RapidIO where Ethernet is carried across a RapidIO network encapsulated with Type 9 packets. In this case, the DMA engine might automatically map MAC or IP address to RapidIO destination or stream IDs.

6 Practical Considerations

As with most ubiquitous technologies, Ethernet has been applied to applications that were not an original design goal. In some cases, economies of scale and being “good enough” have resulted in successful applications. However, the fit becomes increasingly strained as application requirements diverge from Ethernet capabilities. Finding the line between good and not good enough requires consideration of a number of issues. Therefore, this section discusses Ethernet and RapidIO in the context of a variety of issues system designs must confront on a daily basis, such as the impact of standardization, throughput, and power as well as economic considerations of cost and availability.

6.1 Standards and Interoperability

With an ever present push to lower engineering and system costs, leveraging industry standards has gradually increased in importance. An ecosystem of vendors can exist around a standard, increasing competition and the variety of possible solutions. These trends result in lower costs as seen by OEMs and the end customer.

Where standards do not exist or are not administered by the same standards body, a variety of inefficiencies can develop. For example, protocols for Ethernet and the Internet (which is built largely on Ethernet) have been defined by multiple standards bodies. Under the Internet Society (ISOC) are a number of other bodies that propose and manage the Internet infrastructure, including protocols carried by Ethernet. These organizations include, among others:

- Internet Engineering Task Force (IETF)
- Internet Engineering Steering Group (IESG)
- Internet Architecture Board (IAB)
- Internet Assigned Numbers Authority (IANA).

Many L3+ protocols are proposed and discussed through the IETF and are documented through “RFCs.” Underlying this ISOC structure is the Institute of Electrical and Electronics Engineers (IEEE), which has worked on a succession of derivative protocols to move data over a wide variety of physical media. As a direct result of this diversity of standards organizations and interests, no single unified specification is uniformly implemented. Instead, there are many optional protocols resulting in a diversity of actual implementations. This has led to increased Ethernet stack complexity.

There have been occasional violations in compartmentalizing the various protocol layers. For example, the TCP checksum is calculated not just over the TCP header and payload but over elements of the previously-built IP header as well. In IPSec encryption, the IPSec security protocol inserts header shims between existing layers and then encrypts only some of the header fields. These cases and others like it make it difficult to process the protocol in pipeline fashion. Interim hardware pipeline stages are obligated to parse fields up and down the layer hierarchy and thus greatly complicate the implementation of off-load hardware.

Another challenge for Ethernet going forward is how and when protocol off-load engines will be standardized. Today, with the exception of Microsoft TCP Chimney Offload for the Windows/PC world, there is no standard driver-level interface for hardware off-load. This has resulted in many proprietary implementations, each with a proprietary Ethernet stack. Adopters of a particular off-load engine face being locked into that software stack going forward.

Another standards issue for Ethernet is backplane PHY technology for Gigabit and 10 Gigabit Ethernet. As previously noted, an IEEE effort is underway to standardize these electrical elements. However, today there is no standard SERDES PHY for backplane use although a single lane of XAUI exists as the actual standard supported by some Gigabit Ethernet switch vendors.

On the other hand, the RapidIO specification is a single uniform protocol with limited options and consistent protocol layering and governed by a single standards organization (the RapidIO Trade Association). The result is lower implementation cost and reduced overall complexity. With most of the protocol implemented in hardware, software drivers are far more simple than a typical Ethernet TCP/IP stack and can depend on the existence of standardized services.

6.2 Throughput and Latency

Ethernet has high overhead and limited flow control, and it is allowed to drop packets. For control plane applications that cannot tolerate packet loss, this means an Ethernet fabric must be significantly over-provisioned to function well. Over-provisioning may also apply to data plane applications because their tolerance of packet loss and associated latency jitter is limited.

How much over-provisioning is necessary depends on the expected sustained and peak traffic demands of a given system. A range of between 25–35 percent usage is probably the case for many applications. This means a sustainable effective throughput for layer 2 traffic of ~250 Mbps for Gigabit Ethernet and 2.5 Gbps for 10 Gbps Ethernet, depending on average packet size. Over-provisioning causes the fabric to be under-utilized, which reduces end-to-end latency and latency jitter. Unfortunately, end-to-end latency can still be as high as milliseconds because traffic must traverse software stacks at the source and destination endpoint. Latency jitter remains an issue because soft errors still exist in an under-used fabric and occasionally impose significant short term latency to the traffic.

With robust flow control and guaranteed delivery, RapidIO can deliver much higher fabric usage in complex topologies—well in excess of 50 percent. For control plane applications, link-level error correction minimizes the latency jitter incurred as a result of soft errors. End-to-end latency is much lower, potentially much less than 500 ns depending on the topology and component implementation. By minimizing or even eliminating software stacks at the endpoints, RapidIO greatly reduces effective latency in comparison to Ethernet.

6.3 Power

Power can be an important consideration in some applications. There has been considerable debate about the merits of SERDES interconnects when compared to earlier slower single-ended parallel interconnects. A less complex single-ended parallel interconnect controller clearly uses fewer gates and therefore less power. However, the power of single-ended IOs must also be considered and often dominate. For this reason, there is often little incremental power dissipation moving to a SERDES interconnect.

A single-lane XAUI-like PHY similar to that of some Ethernet or RapidIO switches dissipates anywhere from 70–200 mW at 3.125 Gbaud. Such a wide range reflects the process-specific nature of PHY design and the degree of power optimization. As might be expected, 1000Base-T PHY dissipation is much higher at between 640–950 mW. Surprisingly, actual implementations have demonstrated only a modest power savings moving to slower link baud rates on RapidIO. One estimate puts the power reduction from 3.125 Gbaud at slightly lower than linear with respect to frequency. By this measure, a 70 mW design would see less than a 14 mW power drop moving to 2.5 Gbaud.

Another consideration when comparing the power dissipation of an Ethernet versus a RapidIO implementation is that most Ethernet implementations run a software stack and the total interconnect power must account for the processor dissipation required for this stack. With the rule of thumb being a hertz of CPU clock rate per bit of terminated TCP/IP performance, the power required to terminate a line-rate Gigabit Ethernet link would have to include a Gigahertz class processor. Off-load would help considerably here, but the need to have a processor is guaranteed to increase the power by watts.

6.4 Economics

The economics of any silicon-based technology depend on the underlying silicon die size/cost, the volume of silicon shipped, and the size of the silicon and support ecosystem. A casual observer of Ethernet and RapidIO would probably conclude that with these metrics, Ethernet would win on every count. Surprisingly, the picture is considerably more complicated. For example, one RapidIO endpoint (including a messaging function) was only 25 percent larger than the size of a Gigabit Ethernet controller on the same processor. This comparison is not apples-to-apples because the Gigabit Ethernet controller did not implement full TCP/IP off-load (TOE). A fairer comparison would probably show Ethernet with TOE to be at least comparable to RapidIO in size and more likely larger.

Costs of the PHY must be carefully compared. As a direct result of the challenging 1000Base-T channel, this PHY is very large: about 20 mm^2 in 130nm technology. A more apt comparison is between a single-lane XAUI PHY used on some Gigabit Ethernet switches and RapidIO. Because Serial RapidIO is also XAUI-based, the underlying PHY costs are the same. When four lanes of RapidIO are used, a four-lane SERDES is only around 50 percent larger than a single XAUI lane. This would suggest that silicon complexity and area for endpoints are comparable between the RapidIO and Gigabit Ethernet.

A casual observer might quickly conclude that Gigabit Ethernet has a larger ecosystem in terms of switch vendors and much higher volume per device. This is likely true for 4–8 port Gigabit Ethernet switches shipped in very large volume. However, a 12–24 port ATCA-like backplane environment RapidIO looks far more competitive. An apples-to-apples comparison requires an Ethernet switch with VLAN QoS and SERDES PHYs. This quickly reduces the potential vendors and significantly lowers the shipping volume for an Ethernet solution, again making RapidIO much more competitive.

With potentially comparable endpoint costs with Ethernet, RapidIO technology offers 2.5 times more bandwidth per link than Gigabit Ethernet. In addition, today the cost per port of a 16-port RapidIO switch is half as much as the cost of a similar Gigabit Ethernet switch.

When fabric bandwidth requirements exceed 1 Gbps, the only alternative for Ethernet is 10 Gbps. Here, the situation is more stark. Switch solutions based on 802.3ap are still in the future and promise to be more expensive than today's Gigabit Ethernet switches for the foreseeable future. However, today RapidIO offers higher effective bandwidth for payloads less than 1024 bytes, as shown in Figure 14, at lower cost. This comparison ignores the added cost imposed on Ethernet endpoints to deal with Ethernet stacks running at 10 Gbps.

7 Conclusion

System requirements fundamentally determine whether Ethernet or RapidIO is most suitable for a given embedded application. There is no general answer to this question. Ethernet will clearly dominate the larger WAN and the Internet. However, market trends continue to demand increasing performance with higher levels of QoS, suggesting that RapidIO will be increasingly important in meeting many system requirements going forward.

This document has reviewed and compared many specific features of Ethernet and RapidIO as fabric interconnects. The question of which interconnect to use becomes an analysis of the value proposition to evaluate how each interconnect best meets specific application requirements. The following list summarizes these value propositions:

Ethernet:

- Undisputed standard for wide area network (WAN). Provides seamless interworking when the embedded system must interact with a WAN.
- Widely supported by a broad ecosystem that includes endpoints, switches, and software stacks.
- Can be inexpensive for networks with low performance and QoS requirements when applications leverage devices such as high volume endpoints and switches.
- Flexible and adaptable software protocol stack that is easily changed to support new upper-layer protocols.

RapidIO technology:

- Lower CPU overhead due to standardized hardware protocol off-load.
- Significantly higher effective bandwidth than Gigabit Ethernet and higher than 10G Ethernet for payloads < 1KB.
- Quality-of-Service superior to that of many Ethernet implementations:
  - All devices must deliver a minimum level of QoS.
  - Lower latency and latency jitter.
- Standardized backplane electrical elements suitable for embedded systems:
  - Demonstrated margin for embedded backplane designs.
  - Long and short reach options to minimize EMI.
- Multiple link width and speed choices to match applications for cost and power.

- Price competitive against Ethernet as an embedded system fabric:
  - Similar in cost to Gigabit Ethernet but much higher throughput.
  - Significantly lower cost today than 10G Ethernet.
  - Multiple switch vendors.

8 Revision History

Table 11 provides a revision history for this document.

Table 11. Document Revision History

Rev. No.	Date	Substantive Change(s)
0	10/2006	Initial draft release.

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN3088
Rev. 0
02/2007

freescale™ semiconductor