White Paper

Synopsis of the Data Streaming Logical Specification (Phase I)

Based on:

RapidIO Specification
Part X: Data Streaming Logical Specification
Rev. 1.2, 08/2004

Abstract

The Data Streaming specification provides a series of robust components that facilitate the efficient movement of data through a system. This extends the RapidIO standard so that it may be leveraged for many high utilization data plane, as well as control plane, applications.

The focus in this paper is on Phase I of the Data Streaming specification, which defines encapsulation, protocol interworking, and the traffic management framework. The details of the traffic management implementation will be defined in Phase II of the Data Streaming specification and are not discussed in this paper.

Disclaimer

Introduction

RapidIO technology provides a point-to-point, switched interconnect and fabric that can be used for chip-to-chip, board-to-board and chassis-to-chassis communication. RapidIO technology is built upon a modular physical layer that includes both parallel (8/16b up to 2 GHz) and serial (single and quad-lane 1.25/2.5/3.125 GHz) PHYs and a roadmap to higher speed PHYs in the future. All RapidIO packets use a common transport protocol that allows switches to route packets without having to be protocol aware. This enables existing switches to be forward compatible and to operate with all existing RapidIO protocols – even new ones that come along, such as data streaming.

The Message Passing protocol, which has already been in use in the RapidIO interconnect, is an ideal protocol for lossless fabrics or control plane traffic because of its robustness. The RapidIO physical layer acknowledges every packet on a point-to-point basis. In the absence of this acknowledge, the hardware automatically resends the packet. In other control plane buses, like Ethernet, this type of robustness is implemented at layer 4 (TCP/IP) and consumes cycles on already over-burdened CPUs. The end-to-end acknowledge in message passing adds a bit of overhead, but for applications that require a high-level of reliability, that overhead is worth it.

RapidIO interconnect also supports four separate traffic priorities that allow multiple traffic types to share a common fabric. For example, control plane, voice, video, data and multicast can be spread across RapidIO priorities according to an application's unique system requirements.

RapidIO message passing can be used for encapsulation of PDUs up to 4Kb and supports Segmentation and Reassembly (SAR). This has allowed Message Passing to penetrate the data plane market particularly for medium utilization fabrics. RapidIO flow control extensions increased the utilization and efficiency of RapidIO fabrics by allowing receiving devices and switches to provide feedback to traffic sources.

Data plane applications that are required to achieve very high levels of utilization often add fine-grained traffic management capabilities. As target utilization of fabric bandwidth approaches 100%, the fabric may be required to drop traffic when congestion occurs. The Data Streaming Logical Layer has been created to provide this capability. Data Streaming has also incorporated a number of other features that are critical to applications that wish to use RapidIO fabric in the data plane. Data Streaming is a key piece of the standard that enables RapidIO technology to become a full-featured communications fabric.

Overview of Data Streaming

Data Streaming is the latest RapidIO logical protocol extension. It defines a new packet type and other system-level features that are targeted at the data plane. Data Streaming is compatible with all merchant market RapidIO switches because it runs on top of the common transport layer. Data Streaming can also be run over any RapidIO parallel or serial physical layer PHY. Physical and transport layers are not logical protocol aware in the RapidIO architecture.

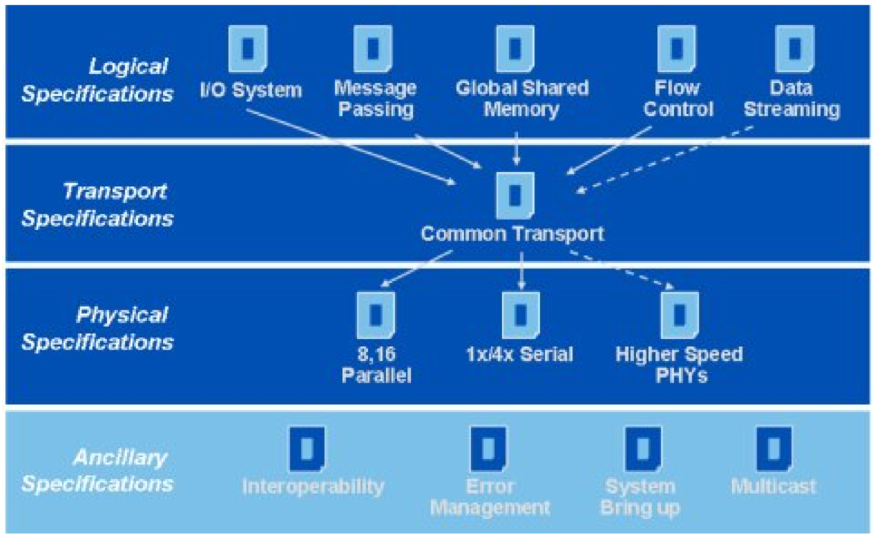

Figure 1: *RapidIO Architecture Specification Layers*

The design goal of the data streaming specification is to combine the need for efficiency, flexibility, and protocol independence in order to minimize the resources necessary to support a dataplane fabric. It enables more intelligence within the system fabric and relieves the system processing resources.

Further, the specification seeks to enable fabric and endpoint implementations spanning a variety of price and performance points ranging from a high-end data plane fabric with advanced traffic management to a simple interworking application where encapsulated data is passed across a basic fabric.

To accomplish this, the data streaming logical specification defines a mechanism for encapsulating and transporting an arbitrary protocol over a standard RapidIO interface. It provides an interconnection between elements in an end-to-end data communications circuit. The data streaming protocol is completely independent of the native protocol of the traffic. Data streaming transactions do not receive responses. Therefore, the protocol makes efficient use of the fabric bandwidth as it maximizes data throughput. It is assumed that the higher layer protocol will provide for end-to-end control if it is required.

The defined encapsulation methodology provides for the multiplexing of different network-layer protocols simultaneously over the same link and provides a common solution for easy connection of a wide variety of hosts, bridges and switches. A RapidIO system will be capable of carrying a wide variety of data types, supporting a diverse set of protocol regimens concurrently.

The data being passed through the RapidIO system may not be directly generated or consumed by the device connected to the RapidIO fabric, but instead by a distant end user. This necessitates the addition of a new protocol to the RapidIO logical layers, which is what the data streaming encapsulation protocol provides.

Functional Features

- o Protocol encapsulation, independent of the protocol that is being encapsulated. Examples include but are not limited to CSIX, ATM, Ethernet, SPI, etc.
- o Support for Protocol Data Units (PDUs) of up to 64k bytes using Segmentation and Reassembly (SAR)
- o Support for hundreds of traffic classes
- o Support for thousands of traffic streams between end points
- o Support for concurrent interleaved PDUs between end points
- o Compatibility with currently shipping RapidIO devices

Physical Features

- o Packet definition is independent of the choice of physical layer interconnection to other devices on the interconnect fabric
- o The protocols and packet formats are independent of the physical interconnect topology (daisy-chain, bus, switched multi-dimensional network, duplex serial connection, etc.)
- o No dependencies exist on the bandwidth or latency of the physical fabric
- o Certain devices have bandwidth and latency requirements for proper operation. The data streaming logical layer specification does not preclude an implementation from imposing these constraints within the system.

Performance Features

- o Packet headers are small, in order to minimize the control overhead.
- o Packet sizes can be limited (MTU size) to control variability in latency.
- o Multiple transactions are allowed concurrently in order to maximize the system throughput, and, multiple traffic flows are supported for high fabric utilization.

The *data streaming logical layer* is intended to support data from a variety of hardware and processing devices. These devices may have various different interfaces, protocols, and degrees of sophistication that can be efficiently transported on the RapidIO fabric.

Data Streams

Data "streams" represent logical connections between an ingress port and an egress port. A connection spans the transfer of multiple data units. The transfer of data units may be separated by discrete intervals of time, based on the arrival of data at the ingress. Transfer between an ingress process and an egress process is unidirectional. An I/O device may be bi-directional containing both an ingress process and an egress process. These processes are usually completely independent consisting of separate streams in each direction.

A given ingress may service hundreds, thousands, even millions of streams at any given time depending on how specifically a unit of data is classified. Traffic may be lumped into a single stream, or classified by user and application to form millions of data streams.

The movement of data path data through a system needs to exhibit significantly different behavior than the movement of control path and traditional DMA transactions. Data streaming transactions differ from these other RapidIO transactions in two ways:

- o They must accommodate larger, variably sized data transfers.
- o The transactions are not acknowledged with a logical layer response packet.

This style of data movement is typically not address-based as with DMA type I/O, and consequently follows a queue based message passing paradigm. Also, data path data movement has significantly more complex requirements in the area of class (or quality) of service than control path communications, and generally requires managing a number of queues at the egress of the system. There is also a need to be able to identify and manage many thousands of data traffic streams that pass through a RapidIO based data path system.

Since many data movement protocols guarantee data delivery in an upper layer protocol, the generation of responses indicating completion is not needed. This allows for more efficient use of bandwidth for data traffic streams.

Data streaming offers a better fit than the RapidIO Message Passing scheme *(Part II: Message Passing Logical Specification*) for applications that necessitate a high degree of traffic management and low overhead. However, the two types of traffic can be integrated using a RapidIO bridge to switch frames between the data streaming interfaces and the message passing interfaces. Generally speaking, end points that contain data streaming capabilities are being designed to provide for this bridging functionality as a built-in feature of the devices.

Traffic Streams

Traffic streams are a type of data stream. A unit of data that contains a discrete identifier is called a Protocol Data Unit (PDU), and a traffic stream is a series of PDUs that have an ordering relationship between each other.

A stream identifier (streamID) distinguishes a traffic stream between two endpoints of RapidIO encapsulated data. For example, a stream ID would identify a stream between a producer (such as a web server) and a consumer (such as a personal computer). Stream identifiers vary with protocol and may include multiple fields from the various networking layers included in the protocol. An ordering relationship is only required for the flow of PDUs within a traffic stream (that is, those that have the same streamID). PDUs with the same producer and consumer pair but with a different streamID might not have an ordering relationship.

The data streaming logical layer uses a virtual stream identifier (VSID) to allow multiple end-to-end traffic streams of PDUs to be uniquely identified and managed concurrently within the RapidIO system. The VSID allows the traffic to be reassociated with an appropriate application at the egress, without having to perform a second protocol specific classification. A VSID may consist of a source or destination address (depending which side of the fabric the packet is located), a class of service and a streamID.

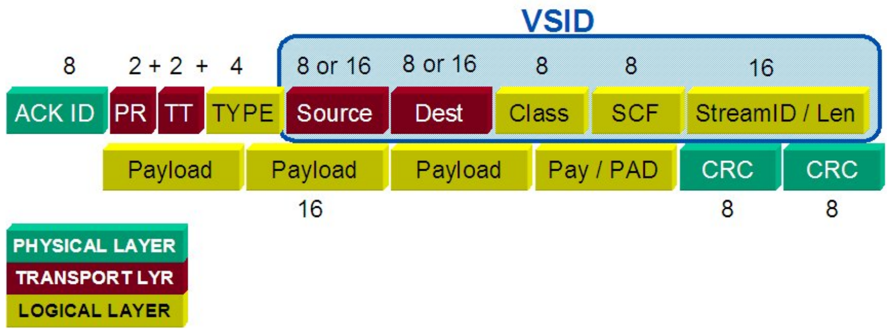

Figure 2: *Data Streaming Packet with Virtual Stream Identifier*

The VSID allows the protocol specific activity to take place one time at the ingress to the fabric. After that, the handling of the PDU is protocol independent. The VSID is used at the destination to 'reclassify' the PDU. This sorts the data back into contexts that can now be protocol specific again. This virtual addressing model eliminates the need for the source and the destination to align the use of buffers and other resources.

Therefore, the VSID can be used to carry a wide variety of information about a stream through the system as required for a specific application. For example, the VSID could carry the indicator for the protocol being encapsulated, the demultiplexing exit port ID instructions, or very fine grained buffer management information. The RapidIO specification leaves the semantics of the VSID flexible in order to accommodate various implementations and still achieve the highest level of efficiency.

Only one PDU from any given stream will be transmitted at a time at the source, but fabric conditions may result in multiple PDUs in transit. The fabric must guarantee that delivery of the PDUs (or segments of PDUs) remain in order. This allows for load balancing of multiple streams; however, a single stream would not be load balanced since it would not guarantee PDU order. PDU ordering immensely simplifies PDU reassembly and keeps the header size compact.

The VSID mechanism provides multiple features condensed in a single key (32 or 40 bits long). These features include:

- A mechanism to manage traffic for ingress to the fabric
- A mechanism to manage traffic in transit within the fabric
- A protocol independent tag to reclassify packets on fabric egress
- A flexible "sub-port" addressing mechanism
- Independence in buffer management

Class of Service and Virtual Queues

Data streaming systems may support thousands, even millions, of active traffic streams. These streams are eventually interleaved onto the single physical fabric channel. The process for deciding which streams may share common resources is sometime referred to as virtual queuing. To facilitate virtual queuing at the ingress and/or egress of the fabric, and to provide for more sophisticated management of traffic streams, the *data streaming logical layer* provides a Class of Service (CoS) identifier. The RapidIO standard offers 256 classes of service, which provides for comprehensive traffic management.

The CoS field forms a specific hierarchy for transitioning packets from highly individualized streams to coarser groupings of traffic. At the fabric ingress, egress, and potentially at interim points (where competition for resources may occur) the traffic may be segregated and queued by class.

The class of service identifier (CoS ID) is a subset of the VSID. At the ingress to the RapidIO fabric, queuing should be based on the destination address and the class of service portions of the VSID, which allow for thousands of streams to be combined into fewer virtual output queues (VoQs). Egress queuing should be based on source ID and class of service. This scheme allows the class of service to be specific to the source and destination pairing.

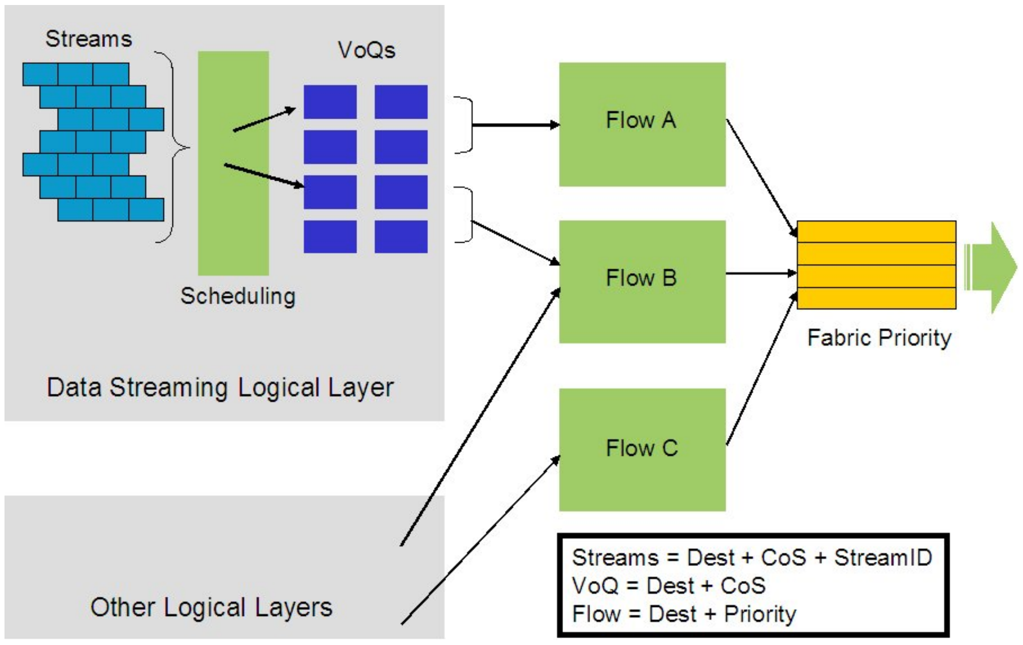

Figure 3: *Relationship of Streams, VoQs and Flows*

Packet Ordering and Classification

Each traffic stream is mapped onto a transaction request flow, which is defined as an ordered sequence of request transactions comprising a specific PDU from a given source to a given destination. There may be multiple transaction request flows between a given source and destination pair. When multiple flows exist between a source and destination pair, an indicator referred to as a "flowID" is used to distinguish the flows from one another. When multiple transaction request flows exist between a given source and destination pair, transactions of a higher priority flow may pass transactions of a lower priority flow, but transactions of a lower priority flow may not pass transactions of a higher priority flow.

The data streaming logical specification requires that all outstanding transactions to another processing element with the same flowID are delivered in the issued order. There is no provision for returning packets that are delivered out of order back into the original issued order, and these packets would be dropped if the error is detected. At the transport layer, switching decisions are made by inspecting the destination ID and the flowID. There is more information on the flowID in other RapidIO specifications, particularly *Part I: Input/Output Logical Specification*.

All PDUs require some form of classification for ingress to the fabric. The application would examine the protocol and create routing information and it would use this to produce the VSID. This VSID contains detailed stream information, which is more comprehensive than simply an 8-bit port address. At the destination, this can be used to re-associate the PDU with a target buffer by using direct addressing or by using a single key table lookup.

This mechanism provides a finely grained and protocol independent way to sort traffic, and a virtual mechanism for buffer pool management. Without a virtual tag, the packet would have to undergo a re-classification based on the protocol specific portion of the PDU. In multi-service platforms, this could save numerous and complex processes, which would prevent the duplication of what was already processed at the source. The overall result is much greater efficiency in the system.

Segmentation and Reassembly

The RapidIO fabric has a 256 byte limit on the packet data payload. Since most data packets exceed this limit, the packets must be broken up into smaller units for transmission across the RapidIO fabric. The mechanism used to do this is commonly referred to as "Segmentation and Reassembly" (SAR).

A PDU that is to be transmitted from the initial producer to the final consumer is broken up (segmented) into a series of blocks of data. The consumer reassembles the data (segments) back into the original PDU. The block size used for the segmentation process is specified by the Maximum Transmission Unit (MTU). The MTU is a system-wide parameter agreed to by all processing elements participating in the SAR process. By managing the MTU size for the system, system latency and jitter can be controlled. The SCF (Segmentation Control Field) of the data streaming packet contains the control bits for the SAR functionality. These bits specify whether the PDU is a start segment, end segment, the segmentation context for which the PDU belongs, as well as some other control information for the SAR process.

The transmission of a PDU for any given stream may result in one or more transactions. A typical sequence is made up of three types of transactions, a start segment, some number of continuation segments, and an end segment. A single segment is considered to be both a start and an end segment.

System Examples

The following examples describe how the data streaming layer may be applied to a Digital Subscriber Line Access Mulitplexer (DSLAM) application and a Voice over IP (VoIP) application.

DSLAM Application

Assume that each line card contains 128 user ports. The system could expose each of these as independent destinations to the RapidIO fabric, requiring the use of an excessively large number of destination IDs in the system, and imposing the associated cost in overhead.

Using data streaming, the ATM traffic can instead be encapsulated into 128 VSIDs, one for each port. The line card would then expose a single port to the RapidIO fabric. The VSID would be used as the address to fan out the traffic on various UTOPIA buses to the user ports. This also has an advantage for fault recovery. Should a line card fail, a single port entry in routing tables in the fabric needs to be updated rather than all 128 sub-ports.

VoIP Application

By incorporating the features of data streaming, the VSID can be used to separate the traffic into just two channels. One channel would be dedicated to a control processor (which would handle the control messages), and the other channel would forward to a network processor (which would distribute the traffic into a bank of DSPs). The VSID could contain the address of the target DSP to further off-load the network processor on distribution. The VSID could also contain the user channel within the DSP de-multiplexing the traffic even further.

Virtual Output Queuing – Fabric On-ramp

Applications involving larger numbers of flows can use the class field to regulate the ingress to the fabric (known as virtual output queuing). For example, the RapidIO fabric interface could contain 256 queues for 64 destination ports with 4 traffic classes. Traffic for each destination of the same class is fairly weighted. The weighting between classes can be application unique.

The traffic is kept sorted by destination. If traffic was just dumped into four queues and a destination port was to fail, the traffic could head of line block the traffic to the other ports, or it would have to be discarded while the port is being recovered or re-routed. By keeping the traffic sorted by destination at the fabric ingress, that destination can be re-routed with minimal traffic loss.

Virtual output queuing can be expanded to 2K or even 16K buffers depending on how large the fabric is and how many different traffic classes are involved. This fabric ingress management can be a simple mechanism to add some quality of service to a system using the destination ID and the class portion of the VSID. Note that this can be done separately from the use of the streamID at the destination for de-multiplexing.

Conclusion

The data streaming specification (Phase I) provides a series of robust components that facilitate the efficient movement of data through a system. This extends the RapidIO standard so that it can be leveraged for many data plane as well as control plane applications. This is facilitated by the use of additional features designed to meet the needs of a data plane application, such as protocol encapsulation, traffic management, classification, and flow priorities.

Data streaming provides for independent protocols to be carried over the RapidIO fabric via the use of protocol encapsulation. Packets are kept to a small, fixed length through the use of segmentation and reassembly. This allows for high efficiency while maintaining minimal overhead. The traffic is deterministic and latency is predictable, allowing for the maximum usage of the bandwidth, while providing for high quality of service.

With the extensions provided in the Data Streaming specification, the RapidIO standard offers a complete solution to deliver high quality, feature rich and cost-effective benefits to a data plane application.