RapidIO™ Interconnect Specification Part 9: Flow Control Logical Layer Extensions Specification

4.1, 6/2017

RapidIO.org

Revision History

Revision	Description	Date
1.0	First release	06/18/2003
1.3	No technical changes, revision changed for consistency with other specifications Converted to ISO-friendly templates	02/23/2005
2.0	Technical changes: new features showing 05-07-00001.003	06/14/2007
2.1	No technical changes	07/09/2012
2.2	Technical changes: errata showing 10-08-00000.003	05/05/2011
3.0	Changed RTA contact information. Technical changes: Addition of references to Dev32 in packet format descriptions.	10/11/2013
3.1	No technical changes.	09/18/2014
3.2	No technical changes.	01/28/2016
4.0	No technical changes.	06/15/2016
4.1	No technical changes.	06/30/2017

Questions regarding RapidIO.org, specifications, or membership should be forwarded to:

RapidIO.org
8650 Spicewood Springs #145-515
Austin, TX 78759
512-827-7680 Tel.

Table of Contents

Chapter 1 Flow Control Overview

1.1 Congestion Management ... 9
1.1.1 Introduction ... 9
1.1.2 Requirements ... 10
1.1.3 Problem Illustration ... 10
1.2 Flow Arbitration ... 12
1.2.1 Fixed / Static Resource Allocation ... 13
1.2.2 Dynamic Resource Allocation Protocol ... 13

Chapter 2 Logical Layer Flow Control Operation

2.1 Fabric Link Congestion ... 15
2.2 Flow Arbitration ... 15
2.2.1 Arbitration Protocol ... 16
2.2.2 Number Of Outstanding Requests ... 18
2.3 Flow Control Operation ... 19
2.4 Physical Layer Requirements ... 20
2.4.1 Fabric Topology ... 20
2.4.2 Flow Control Transaction Transmission ... 20
2.4.2.1 Orphaned XOFF Mechanism ... 21
2.4.2.2 Controlled Flow List ... 21
2.4.2.3 XOFF/XON Counters ... 22
2.4.3 Priority to Transaction Request Flow Mapping ... 22
2.4.4 Flow Control Transaction Ordering Rules ... 23
2.4.5 End Point Congestion Management Rules ... 23
2.4.6 Switch Congestion Management Rules ... 24
2.4.7 Endpoint Rules for the Arbitration Protocol ... 24
2.4.8 Abnormal De-allocation of Resources ... 25

Chapter 3 Packet Format Descriptions

3.1 Introduction ... 27
3.2 Logical Layer Packet Format ... 27
3.3 Flow Arbitration Message Fields (FAM) ... 29
3.4 Transport and Physical Layer Packet Format ... 29

Chapter 4 Logical Layer Flow Control Extensions Register Bits

4.1 Introduction ... 31
4.2 Capability Registers (CARs) ... 31
4.2.1 Processing Elements Features CAR ... 31

Table of Contents

4.2.2 Port n Control CSR 32

Annex A Flow Control Examples (Informative)

A.1 Congestion Detection and Remediation 33
A.2 Orphaned XOFF Mechanism Description 34
A.3 Discussion on Flow Arbitration 35

List of Figures

1-1 Interconnect Fabric Congestion Example 11
2-1 Single PDU Transfer Scenario 16
2-2 Multi-PDU Transfer Scenario 17
2-3 Multi-PDU scenario with receiver based de-allocation scheme 18
2-4 Pipelined Requests 19
2-5 Flow Control Operation 20
3-1 Type 7 Packet Bit Stream Logical Layer Format 28
3-2 LP-Serial Flow Control Packet 30

List of Figures

Blank page

List of Tables

2-1 Prio field to flowID Mapping 23
3-1 Specific Field Definitions and Encodings for Type 7 Packets 28
3-2 Flow Arbitration Protocol Commands....................................29
4-1 Bit Settings for Processing Elements Features CAR....................................31
4-2 Bit Settings for Port n Control CSR....................................32

List of Tables

Blank Page

Chapter 1 Flow Control Overview

RapidIO transacts operations in “flows”. A flow is defined as the nexus of a source, a destination, and a physical channel. The physical channel is a virtual channel and/or a priority within a virtual channel. Since a large number of simultaneous connections can exist within a fabric, resources within the fabric and at the endpoints can be a constraint. The protocols defined in this specification permit the management of resources on a flow basis.

The protocol consists of two functions, congestion management and flow arbitration. Congestion management may be implemented by endpoints or switches independent of the flow arbitration protocol. The flow arbitration protocol only applies to endpoints. Implementation of this specification is optional.

1.1 Congestion Management

1.1.1 Introduction

A switch fabric based system can encounter several types of congestion, differentiated by the duration of the event:

- Ultra short term
- Short term
- Medium term
- Long term

Congestion can be detected inside a switch, at the connections between the switch, and other switches and end points. Conceptually, the congestion is detected at an output port that is trying to transmit data to the connected device, but is receiving more information than it is able to transmit. This excess data can possibly “pile up” until the switch is out of storage capacity, and then the congestion spreads to other devices that are connected to the switch’s inputs, and so on. Therefore, contention for a particular connection in the fabric can affect the ability of the fabric to transmit data unrelated to the contested connection. This is highly undesirable behavior for many applications.

The length of time that the congestion lasts determines the magnitude of the effect the congestion has upon the system overall.

Ultra short term congestion events are characterized as lasting a very small length of time, perhaps up to 500 or so nanoseconds. In a RapidIO type system these events are adequately handled by a combination of buffering within the devices on either end of a link and the retry based link layer mechanism defined in the RapidIO Part 4: 8/16 LP-LVDS Physical Layer and RapidIO Part 6: 1x/4x LP-Serial Physical Layer Specifications. This combination adds “elasticity” to each link in the system. The impact of ultra short term events on the overall system is minor, if noticeable at all.

Short term congestion events last much longer than ultra short term events, lasting up into the dozens or hundreds of microseconds. These events can be highly disruptive to the performance of the fabric (and the system overall), in both aggregate bandwidth and end to end latency. Managing this type of congestion requires some means of detecting when an ultra short term event has turned into a short term event, and then using some mechanism to reduce the amount of data being injected by the end points into the congested portion of the fabric. If this can be done in time, the congestion stays localized until it clears, and does not adversely affect other parts of the fabric.

Medium term congestion is typically a frequent series of short term congestion events over a long period of time, such as seconds or minutes. This type of event is indicative of an unbalanced data load being sent into the fabric. Alleviating this type of congestion event requires some sort of software based load balancing mechanism to reconfigure the fabric.

Long term congestion is a situation in which a system does not have the raw capacity to handle the demands placed upon it. This situation is corrected by upgrading (or replacing) the system itself.

This specification addresses the problem of short term congestion.

1.1.2 Requirements

The flow control mechanism shall fulfill the following goals:

- Simple - excess complexity will not gain acceptance
- React quickly - otherwise the solution won’t work
- Robust - same level of protection and recovery as the rest of RapidIO
- Scalable - must be able to extend to multi-layer switch systems
- Compatibility with all physical layers

1.1.3 Problem Illustration

The *RapidIO Part 1: Input/Output Logical Specification* defines a transaction request flow as a series of packets that have a common source identifier and a common destination identifier at some given priority. On a link, packets of a single transaction request flow can be interleaved with packets from one or more other transaction request flows.

No assumptions are made on the underlying switch architecture for this discussion of the short term congestion problem. Also for the purposes of this discussion, an idealized output queued switch is assumed, which in literature is also used to compare the performance of a particular switch under study. Packet buffers are associated with the output of the switch. An example switch topology showing output buffers is illustrated in Figure 1-1 below. A point of congestion is therefore associated with an output buffer of such a switch.

The problem that is to be addressed by this specification is caused by multiple independent transaction request flows, each with burst and spatial locality characteristics that typically do not exceed the bandwidth capacity of links or end points. Due to the statistical combination of such transaction request flows, usually in the middle of multistage topologies, the demand for bandwidth through a particular link exceeds the link’s capacity for some period of time, for example, Data Flows a, b, and c for an output port of Switch 3 as shown in Figure 1-1. As a result, the output buffer for this port will fill up, causing the link layer flow control to be activated on the links of the preceding switch stages. The output packet buffers for Switches 1 and 2 then also fill up. Packets for transaction request flows, such as data flow d, in these same output buffers not destined for the output port with the full buffer in Switch 3 are now also waiting, causing additional system performance loss. This phenomenon is known as higher order head of line blocking.

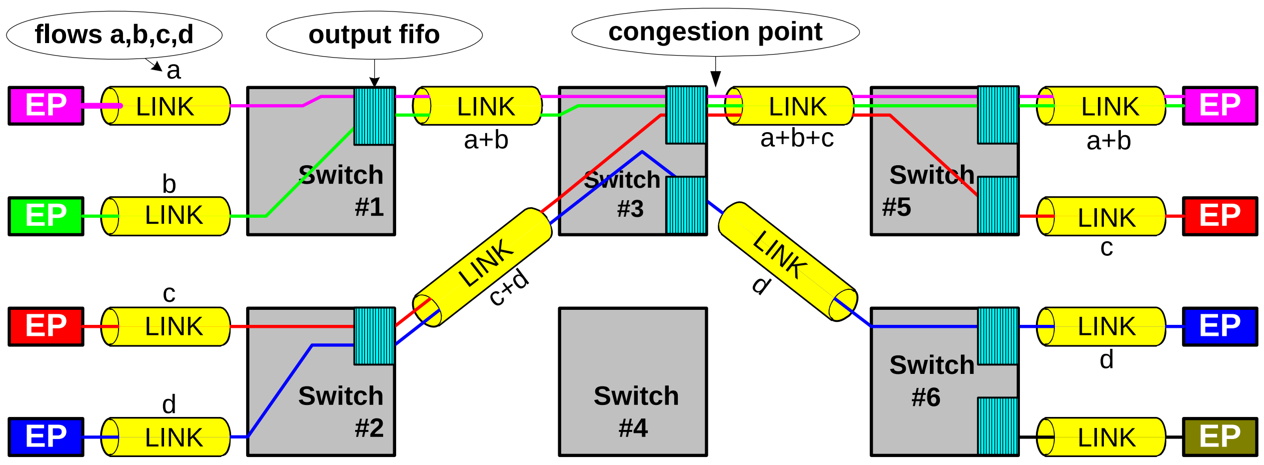

Figure 1-1. Interconnect Fabric Congestion Example

A second problem, less frequently a contributor to system performance loss, occurs when an end point cannot process the incoming bandwidth and employs link layer flow control to stop packets from coming in. This results in a similar sequence of events as described above.

The problem described in this section is very well known in the literature. The aggregate throughput of the fabric is reduced with increased load when congestion control is not applied (see reference [1]). Such non-linear behavior is known as ‘performance-collapse’. It is the objective of this specification to provide a logical layer flow control mechanism to avoid this collapse. Research also shows that relatively simple “XON/XOFF” controls on transaction request flows can be adequate to control congestion in fabrics of significant size.

The reason for the described non-linear behavior is illustrated with a saturation tree. The point at which a single transaction request flow that causes link bandwidth to be exceeded and causes buffer overflow is referred to as the root of the saturation tree. This tree grows backward towards the sources of all transaction request flows going through these buffers, and all buffers that these transaction request flows pass through in preceding stages, causing even more transaction request flows to be affected.

An important design factor for interconnect fabrics is the latency between a congestion control action being initiated and the transaction request flow source acting in response. This latency determines, among other factors, the required buffer sizes for the switches. To keep such buffers small, the latency of a congestion control mechanism must be minimized. For example, 10 data flows contribute to a buffer overflow (forming what is known as a “hotspot”). If it takes 10 packet transmission times for the congestion notification to reach the sources and the last packets sent from the sources to reach the point of congestion after the sources react to the congestion notification, up to 100 packets could be added to the congested buffer. The number of packets added may be much smaller depending on the rate of oversubscription of the congested port.

Reference

[1] “Tree saturation control in the AC3 velocity cluster interconnect”, W. Vogels et.al., Hot Interconnects 2000, Stanford.

1.2 Flow Arbitration

Protocols such as the RapidIO Data Streaming Logical Layer are designed to carry Protocol Data Units (PDUs) of lengths greater than 256 bytes by utilizing Segmentation and Reassembly (SAR). Coherency of the segmentation and reassembly process is enforced by RapidIO’s ordering rules for a packet flow. Since a flow (the nexus of a source, destination, and physical channel) must deliver packets in order, an endpoint ensures coherency of a segmentation / reassembly process by only introducing one complete PDU into a specific flow at a time.

However, an endpoint can potentially connect to up to 64K other endpoints, with 4 or 8 or even 16 physical channels available between each endpoint. As such, an endpoint could have to potentially support millions of dedicated SAR contexts and reassembly buffers. For large PDUs having dedicated reassembly buffers per endpoint could be costly.

The Data Streaming Segmentation and Reassembly contexts are one example of flow based resources that may be a limited resource. Other logical layer functions, like DMA contexts, can also run into resource constraints.

Managing limited resources can be done in a variety of ways, the use of an arbitration protocol is not mandatory:

1.2.1 Fixed / Static Resource Allocation

Fixed allocation of resources occurs by system design. Systems with smaller topologies, or with endpoint resources sufficient for all anticipated flows, do not require any specific management. In larger systems, some portion of the resources can be fixed and assumed to be always available, reducing the number of resources that might have to be further managed.

Resources may also be statically allocated on an individual connection basis. These resources would only be allocated via the overall connection admission algorithm. This additional layer of protection prevents flows from being admitted to the fabric that do not have corresponding resources on the receiving end.

1.2.2 Dynamic Resource Allocation Protocol

The dynamic arbitration protocol is designed to arbitrate and allocate resources to flows for short durations of time. It allows a fewer number of resources to be dynamically shared among a larger number of flows. The system may still require the use of these resources to be intelligently managed in order to achieve desired system performance. The dynamic arbitration of resources will prevent data loss caused by overrunning the receiver.

The congestion management commands affect flows on a packet boundary basis. The arbitration protocol commands affect flows at PDU boundaries (a PDU can consist of one or more packets). Endpoints must have the same understanding of PDU boundaries.

Blank page

Chapter 2 Logical Layer Flow Control Operation

This chapter describes the logical layer flow control mechanisms.

2.1 Fabric Link Congestion

In compliant devices, logical layer flow control methods shall be employed within a fabric or destination end point for the purpose of short term congestion abatement at the point in time and location at which excessive congestion is detected. This remediation scheme shall be enacted via explicit flow control messages referred to as transmit off (XOFF) and transmit on (XON) congestion control packets (CCPs) which, like any other packet, require link-level packet acknowledgements. The XOFF CCPs are sent to shut off select flows at their source end points. Later, when the congestion event has passed, XON CCPs are sent to the same source end points to restore those flows.

The method used to detect congestion is implementation specific and is heavily dependent upon the internal packet buffering structure and capacity of the particular switch device. In the example output port buffered switch from “Section 1.1.3, Problem Illustration”, on page 10, congestion occurs when some output buffer watermark is exceeded, but this is not the only way of detecting congestion. Several possible implementation methods are described in Appendix A. These described methods are purely exemplary and are not intended to be an exhaustive list of possible methods.

2.2 Flow Arbitration

The flow arbitration protocol extends the Congestion Control Packet (CCP) protocol first introduced in Revision 1.3 of this specification. In addition to the XON/XOFF congestion management functionality the arbitration protocol adds the following commands:

- REQUEST
- XOFF to indicate un-availability of resources.
- XON to allow and grant use of resources
- RELEASE

2.2.1 Arbitration Protocol

The protocol is illustrated in the following diagrams, using Data Streaming allocation of SAR resources as examples. There are two request messages pertaining to single PDU and multi-PDU transfers. The single PDU case is illustrated in Figure 2-1. The transmitting endpoint sends a single PDU request. The receiving endpoint will respond with either a XON(ARB) or XOFF(ARB) message depending on whether it has buffer and context resources available.

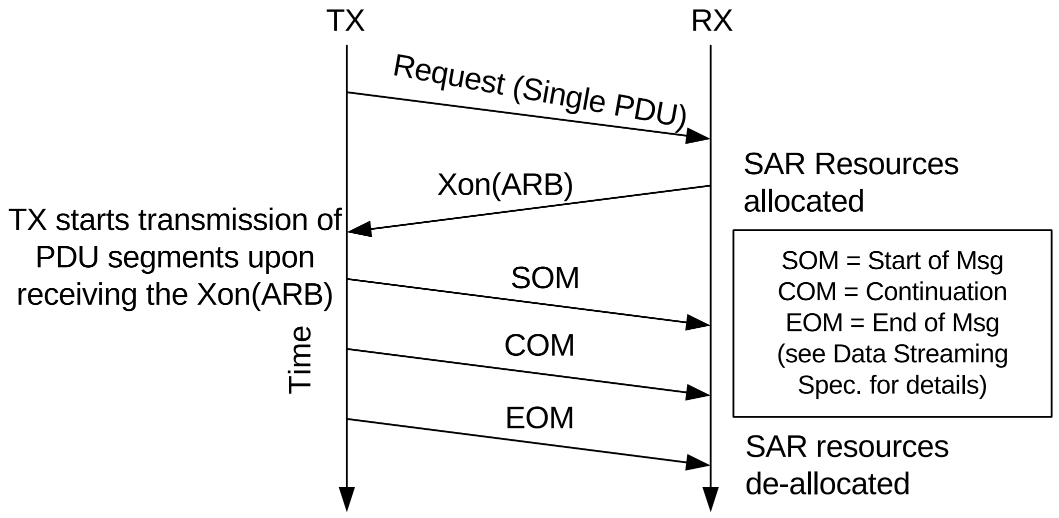
TX
RX
Request (Single PDU)
Xon(ARB)
SAR Resources allocated
TX starts transmission of PDU segments upon receiving the Xon(ARB)
SOM
COM
EOM
Time
SOM = Start of Msg
COM = Continuation
EOM = End of Msg
(see Data Streaming Spec. for details)
SAR resources de-allocated

Figure 2-1. Single PDU Transfer Scenario

In the single PDU transfer case, if the receiving endpoint responds with a XOFF(ARB), the transmitting endpoint can send a new REQUEST message to ask for resources. If the receiving endpoint responds with a XON(ARB), the transmitting endpoint can start transmitting the PDU segments once it receives the XON(ARB) message. The receiver will automatically de-allocate resources once it receives the last packet for the PDU.

When the transmitting endpoint sends a request pertaining to the transfer of multiple PDUs the receiving endpoint, similarly to the single PDU case, shall respond with either the XON(ARB) or the XOFF(ARB) protocol depending on the availability of buffering resources. If the receiving endpoint responds with a XON(ARB) message, as shown in Figure 2-2, the transmitting endpoint can start sending the PDU segments once it receives the XON(ARB) message. The transmitter can send multiple PDUs without having to renegotiate the resources. The receiver will hold the allocated resources until it receives a RELEASE message from the transmitting endpoint.

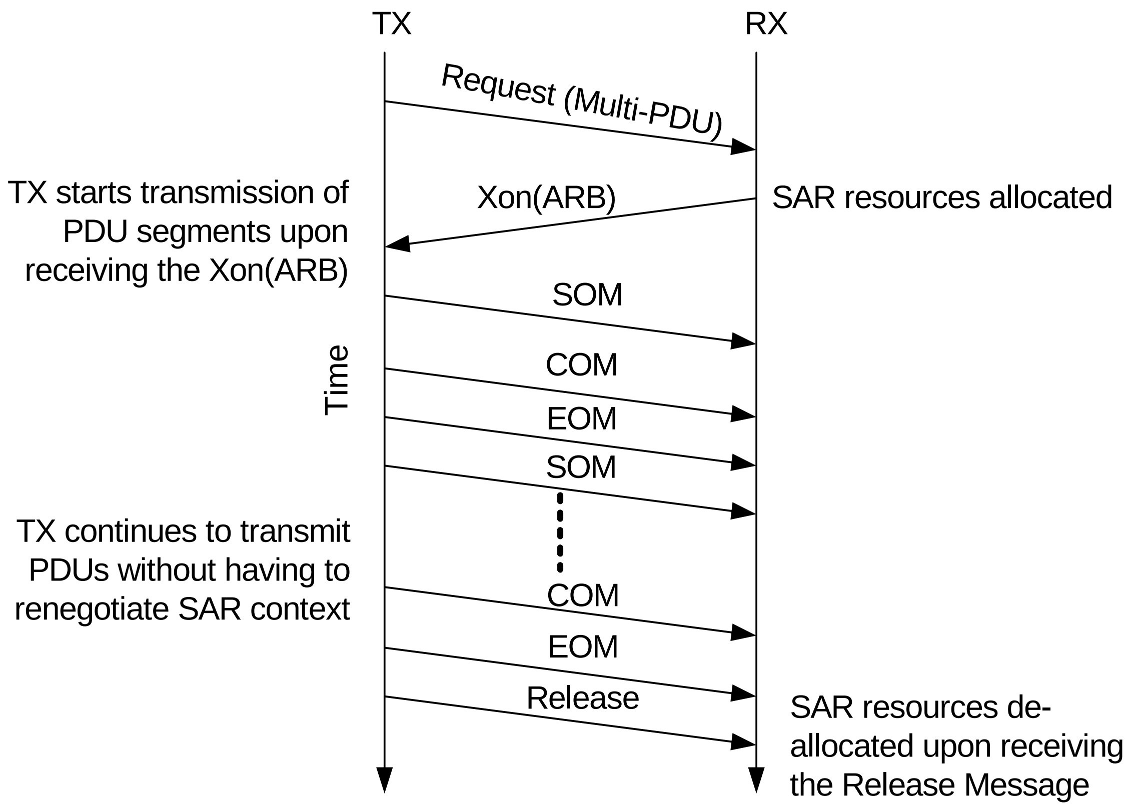
TX
RX
Request (Multi-PDU)
Xon(ARB)
SAR resources allocated
TX starts transmission of PDU segments upon receiving the Xon(ARB)
SOM
Time
COM
EOM
SOM
TX continues to transmit PDUs without having to renegotiate SAR context
COM
EOM
Release
SAR resources de-allocated upon receiving the Release Message

Figure 2-2. Multi-PDU Transfer Scenario

The receiver can also inform the transmitting endpoint of its desire to de-allocate the resources, by sending a XOFF(ARB) message. The transmitting endpoint, after sending the last packet at the current PDU boundary, will send a RELEASE message. The receiver shall de-allocate the resources only when it has received the RELEASE message. This scenario is illustrated in Figure 2-3.

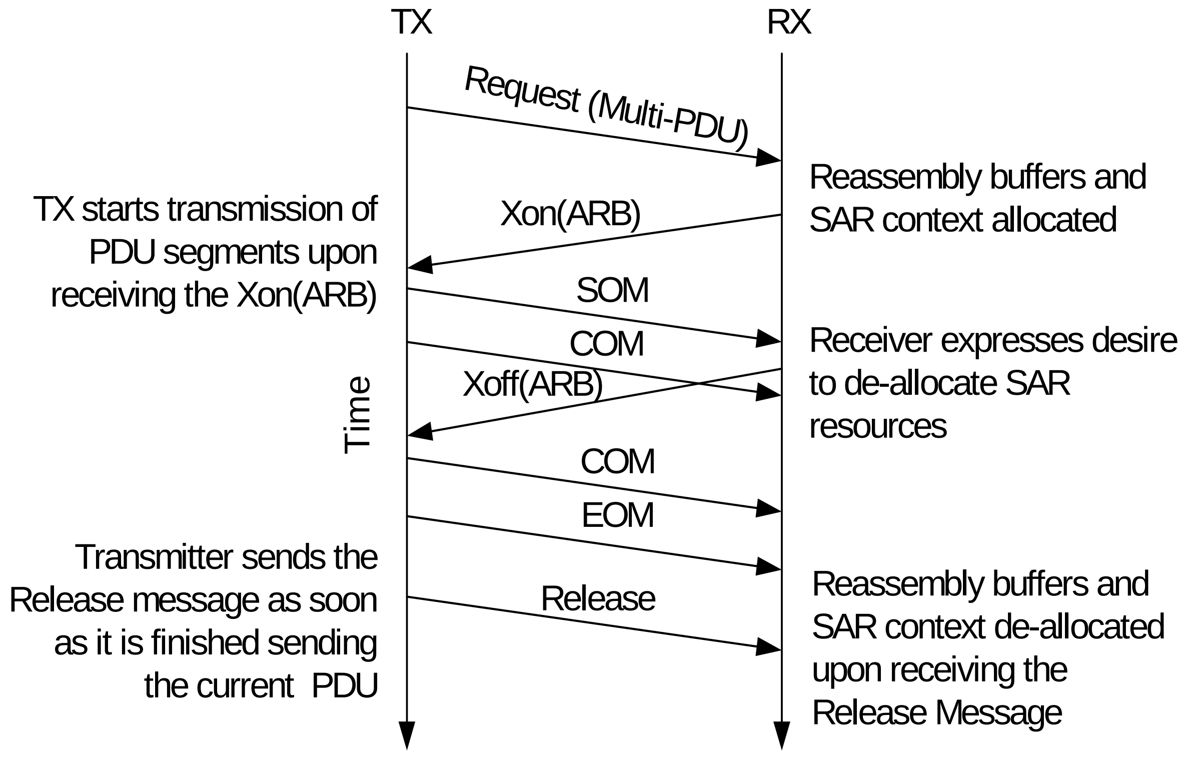
TX
RX
Request (Multi-PDU)
Reassembly buffers and SAR context allocated
TX starts transmission of PDU segments upon receiving the Xon(ARB)
Xon(ARB)
SOM
COM
Receiver expresses desire to de-allocate SAR resources
Xoff(ARB)
Time
COM
EOM
Transmitter sends the Release message as soon as it is finished sending the current PDU
Release
Reassembly buffers and SAR context de-allocated upon receiving the Release Message

Figure 2-3. Multi-PDU scenario with receiver based de-allocation scheme

2.2.2 Number Of Outstanding Requests

In the arbitration protocol the transmitting endpoint has to wait at least a round trip time after it has sent the request message before it can start transmitting. This delay may be undesirable in high performance systems. Therefore, in order to overlap the request phase with the data transmission phase, the transmitter is allowed to have a maximum of one outstanding request in the system, that is, it can pipeline requests to increase the efficiency of the system. The requests and the corresponding responses are identified by a 1 bit sequence number. This pipelining of requests is allowed for both single PDU and multi-PDU requests.

Consider the exchange shown in Figure 2-4 below. The transmitting endpoint issues a request. The request is processed by the receiving endpoint and a XON issued. Once the transmitter receives the XON message, it can start transmitting the data and it may also pipeline another request. The pipelined request shall not be honored until the current transaction has been completely received and resources are available for the next transaction. In Figure 2-4 Request_1 is sent after the transmitting endpoint has received the XON_0(ARB) message for the previous request (Request_0). If the receiving endpoint for some reason cannot queue/process the requests, it can send a XOFF (ARB) message immediately to indicate lack of resources.

The pipelining of requests is managed by the source. It only issues the next request when the current request has been acknowledged. The destination only acknowledges the next request when the current request has completed. So, the destination only has to queue up one outstanding request per flow. This pipelining also offers the destination an opportunity to use the pending requests to get a look at the incoming traffic and make better allocation decisions should there be limited resources.

A single level of pipelining of requests is adequate because this is on a per flow basis. flow may only have a single open context at a time, so the current context must complete before the flow can be used for another transaction.

NOTE: Context Definition

As a reminder, a "context" is a group of individual transactions that must remain ordered, and may not have intervening transactions from a different context in the same flow.

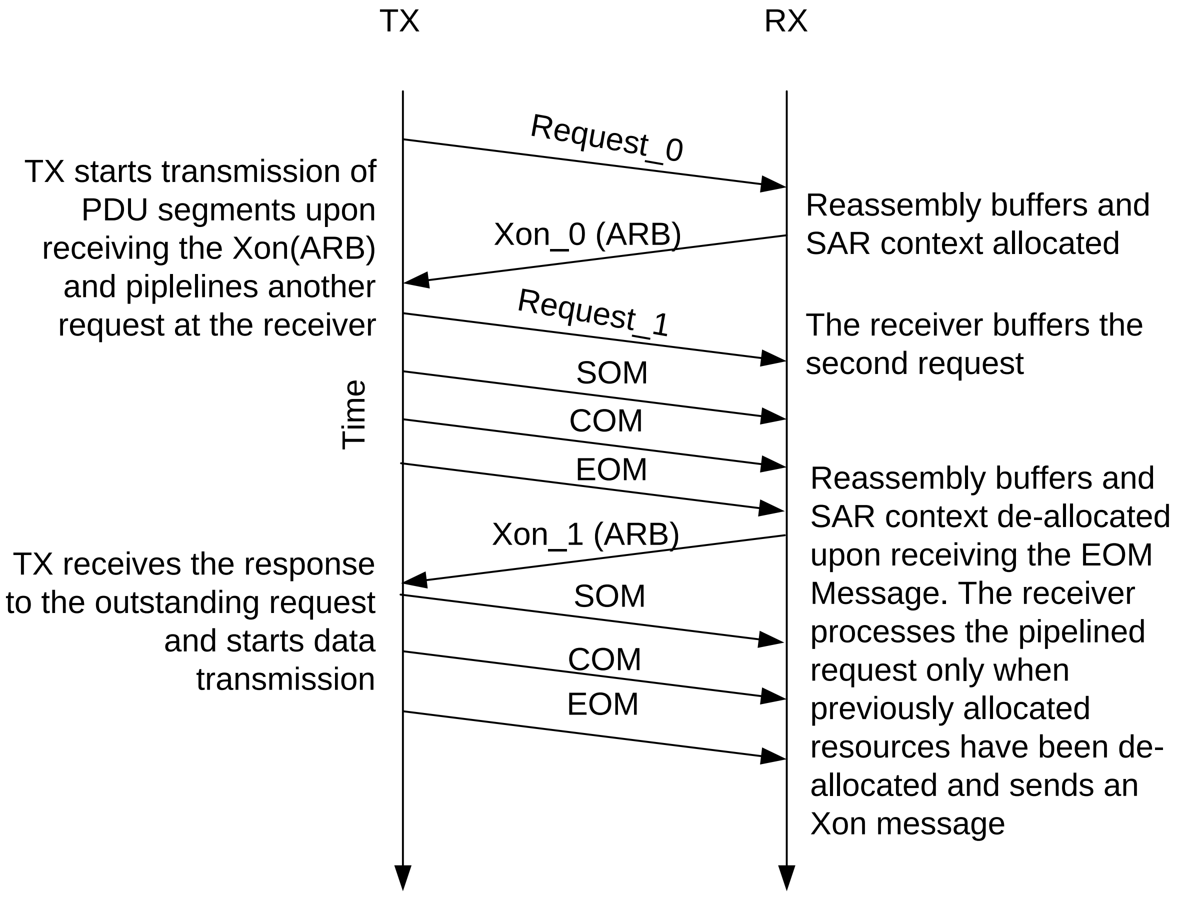
TX
RX
Request_0
TX starts transmission of PDU segments upon receiving the Xon(ARB) and piplelines another request at the receiver
Reassembly buffers and SAR context allocated
Xon_0 (ARB)
Request_1
The receiver buffers the second request
SOM
COM
EOM
Time
Reassembly buffers and SAR context de-allocated upon receiving the EOM Message. The receiver processes the pipelined request only when previously allocated resources have been de-allocated and sends an Xon message
Xon_1 (ARB)
TX receives the response to the outstanding request and starts data transmission
SOM
COM
EOM

Figure 2-4. Pipelined Requests

2.3 Flow Control Operation

The flow control operation consists of a single FLOW CONTROL transaction as shown in Figure 2-5. The FLOW CONTROL transaction is issued by a switch or end point to control the flow of data. This mechanism is backward compatible with RapidIO legacy devices in the same system.

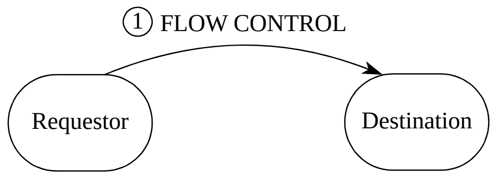

Figure 2-5. Flow Control Operation

While FLOW CONTROL packets do not contain response packets, the arbitration protocol does consist of multiple transactions between the source of a data flow and the destination. Some of the transactions flow from the destination to the source.

2.4 Physical Layer Requirements

This section describes requirements put upon the system physical layers in order to support efficient logical layer flow control.

2.4.1 Fabric Topology

The interconnect fabric for a system utilizing the logical layer flow control extensions must have a topology such that a flow control transaction can be sent back to any transaction request flow source. This path through the fabric may be back along the path taken by the transaction request flow to the congestion point or it may be back along a different path, depending upon the requirements of the particular system.

2.4.2 Flow Control Transaction Transmission

Flow control transactions are regarded as independent traffic flows. They are the most important traffic flow defined by the system. Congestion management transactions are always transmitted at the first opportunity at the expense of all other traffic flows if possible. For the 8/16 LP-LVDS and 1x/4x LP-Serial physical layer specifications, this requires marking flow control packets with a “prio” field value of 0b11, and a “CRF” bit value of 0b1, if supported. Flow arbitration has additional requirements for some transactions to be transmitted in the same flow as the data packets. All of these transactions use a normal packet format for purposes of error checking and format.

Because an implicit method of flow restoration was simulated and found to be impractical for RapidIO fabrics due to lack of system knowledge in the end point, an explicit restart mechanism using a XON transaction is used. In the CCP flow back to the source end point, XOFF and XON CCPs may be dropped on input ports of downstream elements in the event of insufficient buffer space.

2.4.2.1 Orphaned XOFF Mechanism

Due to the possibility of XON flow control packets being lost in the fabric, there shall be an orphaned XOFF mechanism for the purpose of restarting orphaned flows which were XOFF'd but never XON'd in end points. Details of this mechanism are implementation specific, however the end point shall have sufficient means to avoid abandonment of orphaned flows. A typical implementation of such a mechanism would be some sort of counter. A description of a possible implementation is given in Appendix A. The Orphaned XOFF Mechanism is intended to work with the rest of the XON/XOFF CCPs to handle the short term congestion problem as previously described, and so shall operate such that software intervention is not required or inadvertently invoked.

Counter mechanisms for arbitrated flows are also an implementation decision, but care should be taken before just enabling transmission on a flow. Unlike the congestion management protocol, arbitrated flows are expected to remain off until explicitly enabled. Timeouts at source end points should result in retrying requests, not just arbitrarily starting a flow.

2.4.2.2 Controlled Flow List

It is required that elements which send XOFFs keep a list of flows they have stopped, along with whatever flow-specific information is needed to select flows for restart, such as per-flow XON watermark level, or relative shut off order. This information shall be stored along with flow identification information in a "controlled flow list", a memory structure associated with the controlling element. It shall be permissible in the time following the sending of a XOFF CCP for the flow control -initiating element to re-evaluate system resources and modify the flow restart ordering or expected XON watermark level within the controlled flow list to better reflect current system state. It shall not however be permissible to abandon the controlled flow by "forgetting" it, either due to lack of controlled flow list resources or other factors. In the event that limited controlled flow list resources cause the congested element to have insufficient room to issue another XOFF CCP which is deemed more important than a previously-XOFF'd controlled flow, then that previously-XOFF'd controlled flow may be prematurely XON'd and removed from the controlled flow list. The new, more important flow may be XOFF'd and take its place in the controlled flow list.

Details of the controlled flow list are implementation specific, though at the very least it shall contain entries for each currently XOFF'd flow, including flow identification information. It is likely that some state information will be required, such as expected time of flow restart, or per-flow restart watermark levels. The controlled flow list size is selected to provide coverage for short term congestion events only. Remediation for medium and greater -term congestion events is beyond the scope of logical layer flow control as these events likely indicate systemic under-provisioning in the fabric.

Arbitrated resources must also be associated with a flow list to keep track of the flow the resources are allocated to. Should the source fail to utilize the resource in an expected interval, the destination may take action to recover the resource. The arbitration protocol provides a method to attempt to deallocate the resource in concert with the source to avoid packet loss (see “Section 2.2.1, Arbitration Protocol”, on page 16). Should that fail, asynchronous de-allocation of the resource may be used, with the understanding that packet loss could result. The implementation of such a mechanism is not specified here. Care should be taken to account for fabric latencies and not cause excessive packet loss during higher latency intervals.

2.4.2.3 XOFF/XON Counters

XOFF/XON counters shall be instantiated for some number of output flows at the end point. Since the number of flows may be large or unpredictable, the number of counters and how flows are aggregated to a particular counter is implementation dependant. However, all flows must be associated with a counter. For simplicity, the following behavioral description assumes a single flow associated with a single counter. The counter is initialized to zero at start up or when a new DestinationID and given Priority is initialized. The counter increments by one for each associated XOFF CCP and decrements by one for each associated XON CCP, stopping at zero. Only when this counter is equal to zero is the flow enabled. In no event shall the counter wrap upon terminal count. If the orphaned XOFF mechanism activates, the counter is reset to zero and the flow is restarted.

2.4.3 Priority to Transaction Request Flow Mapping

When a switch or end point determines that it is desirable to generate a flow control transaction, it must determine the associated flowID for the (non-maintenance and non-flow control) packet that caused the flow control event to be signalled. Maintenance and flow control transaction request flows must never cause the generation of a flow control transaction. For the 8/16 LP-LVDS and the 1x/4x LP-Serial physical layer specifications, the flowID of a transaction request flow is mapped to the “prio” bits as summarized in Table 1-3 of the 8/16 LP-LVDS specification and Table 5-1 of the 1x/4x LP-Serial specification. Determining the original transaction request flow for the offending packet requires the switch to do a reverse mapping.

It is recognized that mapping a particular response to a particular transmission request may be inaccurate because the end point that generated the response is permitted in the physical layer to promote the response to a priority higher than would normally be assigned. Deadlock avoidance rules permit this promotion. For this reason the choice of which flow to XOFF is preferably made using request packets, not response packets, as responses release system resources, which also may help alleviate system congestion.

Additionally, the CRF bit should also be used in conjunction with flowID to decide whether or not a particular transaction request flow should be targeted with a XOFF flow control transaction. A switch may select for shut off a packet with CRF=0 over a packet with CRF=1 if there are two different flows of otherwise equal importance. Correspondingly, an end point may choose to ignore a flow control XOFF request for a transaction request flow that it regards as critical.

The reverse mappings from the transaction request flow prio field to the CCP flowID field for the 8/16 LP-LVDS and 1x/4x LP-Serial physical layers are summarized in Table 2-1.

Table 2-1. Prio field to flowID Mapping

Transaction Request flow prio Field	Transaction Type	System Priority	CCP flowID
0b00	request	Lowest	A
0b00	response	Illegal	
0b01	request	Next	B
0b01	response	Lowest	A
0b10	request	Highest	C or higher
0b10	response	Lowest or Next	A or B
0b11	request	Illegal	
0b11	response	Lowest or Next or Highest	A, B, C or higher

2.4.4 Flow Control Transaction Ordering Rules

The ordering rules for flow control transactions within a system are analogous to those for maintenance transactions.

1. Ordering rules apply only between the source (the original issuing switch device or destination end point) of flow control transactions and the destination of flow control transactions.
2. There are no ordering requirements between flow control transactions and maintenance or non-maintenance request transactions.
3. A switch processing element must pass through flow control transactions between an input and output port pair in the order they are received.
4. An end point processing element must process flow control transactions from the same source (the destination of the packet that caused the flow control event) in the order they are received.

2.4.5 End Point Congestion Management Rules

There are a number of rules related to flow control that are required of an end point that supports the logical layer flow control extensions.

1. A XOFF flow control transaction stops all transaction request flows of the specified priority and lower targeted to the specified destination and increments the XON/XOFF counter associated with the specified flowID.
2. A XON flow control transaction decrements the XON/XOFF counter associated with the specified flowID. If the resulting value is zero, the transaction request flows for that flowID and flowIDs of higher priority are restarted.
3. An end point must be able to identify an orphaned XOFF'd flow and restart it.
4. A destination end point issuing a XOFF Flow Control transaction must maintain the information necessary to restart the flow with a XON flow control transaction when congestion abates.
5. Upon detection of congestion within one of its ports, the destination end point shall send required CCP(s) as quickly as possible to reduce latency back to the source end point.

2.4.6 Switch Congestion Management Rules

There are a number of rules related to flow control that are required of a switch that supports the logical layer flow control extensions.

1. Upon detection of congestion within a port, the switch shall send a CCP (XOFF) for each congested flow to their respective end points.
2. If a switch runs out of packet buffer space, it is permitted to drop CCPs.
3. A switch issuing a XOFF Flow Control transaction must maintain the information necessary to restart the flow with a XON flow control transaction when congestion abates.

2.4.7 Endpoint Rules for the Arbitration Protocol

Transmitters shall not transmit on an arbitrated flow unless a resource is available for reception of the PDU. Assumption of an available resource can either be fixed, statically allocated, or dynamically allocated. If dynamically allocated, the protocol must obey the following rules:

1. The transmitter shall issue a REQUEST when it wishes the receiver to allocate a resource to a particular flow. Note that this does not imply a PDU is immediately ready for transmission. The algorithms for resource allocation are up to the implementation.
2. The receiver shall respond to all REQUEST messages with a XOFF(ARB) or XON(ARB) depending on the availability of resources and the arbitration policy at the receiver.
3. The transmitter may send a new REQUEST message if: 1) it did not receive a response to the previous REQUEST message and timed out, or 2) it received a XOFF(ARB) message from the receiving endpoint.

4. Sequence numbers shall remain coherent for each individual flow. The sequence number shall remain the same for REQUESTs reissued for a given flow, without having received a response. The sequence number shall advance for any new REQUESTs on a given flow.
5. If a single PDU request is granted, the receiver may deallocate the resources at any time after: 1) it receives the last segment for the PDU, or 2) it does not receive a packet and an idle counter for the session times out (see rule l). The transmitter shall assume the context is no longer available upon sending the last segment for the PDU.
6. If the resources were granted in response to a multi-PDU request the transmitter may transmit PDUs continuously on that flow until the resource is de-allocated.
7. The transmitter may relinquish a multi-PDU context by sending a RELEASE message after completion of the current PDU.
8. The receiver may send a XOFF(ARB) message during the multi-PDU transfer to indicate its desire to deallocate resources. The transmitter, upon receiving a XOFF(ARB) message during the multi-PDU transfer, shall complete transmission of the current PDU and send a RELEASE message to allow the receiver to de-allocate the resources. The receiver may not deallocate the resources until the RELEASE message is received.
9. The receiver may delay sending responses to the REQUEST commands to consider which REQUESTs to grant or reject. There is no ordering requirement for processing requests from different flows.
10. Only a single instance of resources shall be allocated to a flow at any point in time.
11. The transmitter may issue a single additional REQUEST in advance of completion of the current PDU. However, the transmitter shall not have more than one outstanding REQUEST at any point in time.
12. REQUEST, XON(ARB), and XOFF(ARB) messages may be sent in any flow (such as a high priority channel). RELEASE messages shall be sent in the same flow that the context is allocated for.

2.4.8 Abnormal De-allocation of Resources

Abnormal de-allocation of Resources will occur under the following circumstances:

1. If the resources were allocated in response to a single PDU request:

– The PDU may be aborted according to the exceptions defined in the logical layer rules for the segmentation process. An aborted PDU results in the de-allocation of the resources.

– If the receiver does not receive a packet from the transmitter before the idle counter for the session times out, the resources would be de-allocated. Note that the use of a timer is implementation specific. Incorrect use of a timer may result in packet loss.

2. If the resources were allocated in response to a Multi-PDU request:

– A PDU may be aborted according to the exceptions defined in the logical layer. The resources will still not be de-allocated until a RELEASE message is received.

– If the receiver does not receive a packet from the transmitter before the idle counter for the session times out, the receiver shall first attempt to use the XOFF(ARB) / RELEASE handshake to deallocate the context. If a subsequent timeout is encountered, the SAR resources are asynchronously de-allocated.

NOTE: Timeouts for Arbitration Protocol are Optional

Use of timers with the arbitration protocol is optional. Timer intervals are specific to system implementation and performance goals. Aggressive timer intervals may result in retrying operations that were simply slowed down due to system congestion. Aggressive recovery of resources may also result in packet loss. Conservative time intervals may result in poor performance if transactions are lost or corrupted. It is up to the implementer to determine the correct behavior for the specific system environment.

Chapter 3 Packet Format Descriptions

3.1 Introduction

This chapter contains the definition of the flow control packet format. The type 7 FLOW CONTROL packets are used by the switch or the end points to signal congestion buildup within the node (switch or endpoint) or exchange flow arbitration protocol messages.

3.2 Logical Layer Packet Format

The type 7 FLOW CONTROL packet formats (Flow Control Class) are used by a RapidIO switch or end point processing element to stop (XOFF) and start (XON) the flow of traffic to it from a targeted RapidIO end point processing element. A single transaction request flow is targeted with a CCP. Type 7 packets do not have a data payload and do not generate response packets. The origin of a flow control packet shall set the SOC (Source of Congestion) bit to (SOC=0) if it is a switch or (SOC=1) if it is an end point. The SOC bit is informational only but may be useful for system software in identifying a failing end point.

The Flow Arbitration Message (FAM) field is used to modify the XON or XOFF message for the purposes of flow arbitration.

Definitions and encodings of fields specific to type 7 packets are provided in Table 3-1.

Table 3-1. Specific Field Definitions and Encodings for Type 7 Packets

Type 7 Fields	Encoding	Definition
XON/XOFF	0b0	XOFF For devices not supporting flow arbitration: Stop issuing requests for the specified and lower priority transaction request flows For devices supporting flow arbitration (see Table 3-2)
	0b1	XON For devices not supporting flow arbitration: Start issuing requests for the specified and higher priority transaction request flows For devices supporting flow arbitration (see Table 3-2)
flowID	 0000000 (Flow0A) 0000001 (Flow 0B) 0000010 (Flow 0C) 0000011 (Flow 0D) 0000100 (Flow 0E) 0000101 (Flow 0F) 1000001 (Flow 1A) 1000010 (Flow 2A) 1000011 (Flow 3A) 1000100 (Flow 4A) 1000101 (Flow 5A) 1000110 (Flow 6A) 1000111 (Flow 7A) 1001000 (Flow 8A)	Highest priority affected transaction request flow for VC0 transaction request flow A transaction request flow B transaction request flow C transaction request flow D transaction request flow E transaction request flow F and higher For VC1-VC8 the following flow IDs will result in the VC1-VC8 flow control Flow 1A and higher for VC1 Flow 2A and higher for VC2 Flow 3A and higher for VC3 Flow 4A and higher for VC4 Flow 5A and higher for VC5 Flow 6A and higher for VC6 Flow 7A and higher for VC7 Flow 8A and higher for VC8 Remaining encodings are reserved for the 8/16 LP-LVDS and the 1x/4x LP-Serial physical layers.
destinationID	—	Indicates which end point the CCP is destined for (sourceID of the packet which caused the generation of the CCP).
tgtdestinationID	—	Combined with the flowID field, indicates which transaction request flows need to be acted upon (destinationID field of the packet which caused the generation of the CCP).
SOC	0b0	Source Of Congestion is a Switch
	0b1	Source Of Congestion is an End Point
FAM		See Section 3.3
rsrv	0b0000	Reserved

Figure 3-1 displays a CCP packet with all its fields. The field value 0b0111 in Figure 3-1 specifies that the packet format is of type 7. Dev8 (tt=0b00) and Dev16 (tt=0b01) Transport Formats are shown in the figure, additionally there is the Dev32 (tt=0b10) transport size.

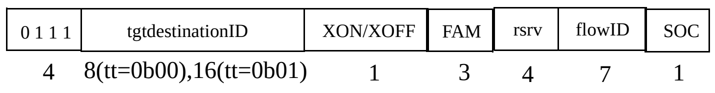

Figure 3-1. Type 7 Packet Bit Stream Logical Layer Format

3.3 Flow Arbitration Message Fields (FAM)

The flow arbitration protocol uses the 3 FAM bits along with the XON/XOFF bit to identify the messages. A device that does not support the SAR protocol ignores the FAM bits. It should be noted that a device which supports the flow arbitration protocol when communicating with an end point that does not support SAR protocol should default to the congestion management (XON/XOFF) functionality and not send other messages as they would be mis-interpreted. The CAR bits define whether the device supports the flow arbitration protocol or not. The bit "Y" is the sequence number bit previously described.

Table 3-2. Flow Arbitration Protocol Commands

XON/XOFF	FAM	Definition
0	0b000	XOFF: Transmit off (Congestion management) Stop issuing requests for the specified and lower priority transaction request flows
	0b010	XOFF(ARB): Flow Request Rejected. Message with sequence number 0 (LSB) in response to REQUEST with sequence number 0.
	0b011	XOFF(ARB):Flow Request Rejected. Message with sequence number 1(LSB) in response to REQUEST with sequence number 1.
	0b10Y	RELEASE: Release message informs the receiving endpoint to de-allocate the buffer space reserved by the receiving endpoint. The release message should be used in conjunction with the request.
1	0b000	XON: Transmit on. (Congestion management) Start issuing requests for the specified and higher priority transaction request flows
	0b01Y	XON(ARB): Flow Request Granted. Reassembly buffer space is now available and allocated.
	0b10Y	REQUEST: Request Flow Single PDU. Buffer space will be de-allocated once the end transaction is received for that PDU.
	0b11Y	REQUEST: Request Flow Multi-PDU. This request message informs the receiving endpoint to reserve the buffer space until it receives the release message. The buffer space will be de-allocated once the release command is received by the receiver.

3.4 Transport and Physical Layer Packet Format

Figure 3-2 shows a complete flow control packet, including all transport and 1x/4x LP-Serial physical layer fields except for delineation characters. The destinationID field of the CCP packet is the sourceID field from packets associated with the congestion event, and is the target of the flow control transaction. The tgtdestinationID field is the destinationID field from packets associated with the congestion event, and was the target of those packets. The tgtdestinationID field is used by the target of the flow control packet to identify the transaction request flow that needs to be acted upon. For all undefined flowID encodings, there is no action required and the tgtdestinationID is ignored. Field size differences for 8 bit address Dev8 Transport Format (tt=0b00) vs. 16 bit address Dev16 Transport Format (tt=0b01) are shown, additionally the Dev32 (tt=0b10) format can be used. Note: when tt=0b01 there will be a pad after the CRC.

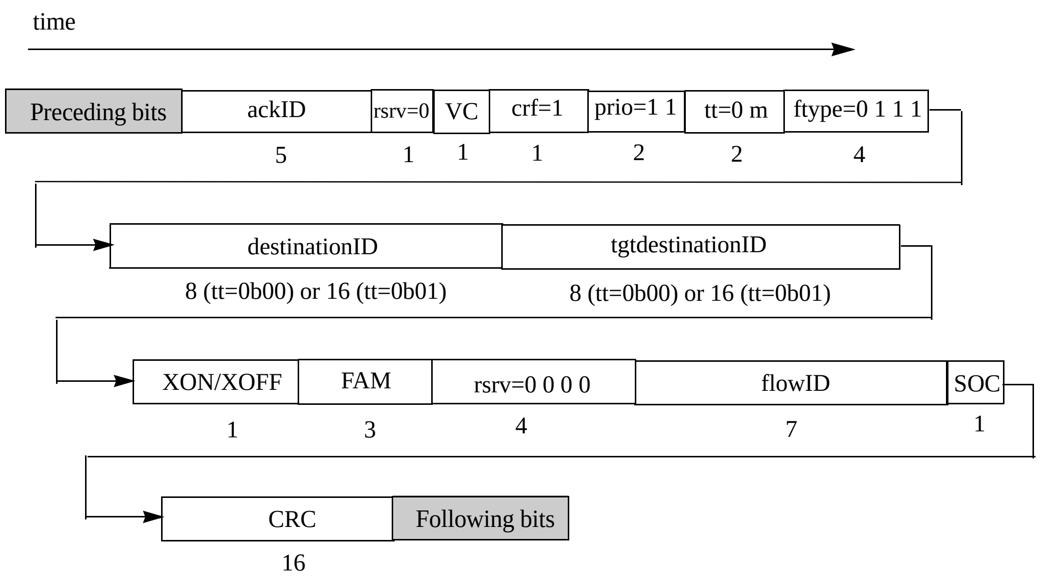

Figure 3-2. LP-Serial Flow Control Packet

Chapter 4 Logical Layer Flow Control Extensions Register Bits

4.1 Introduction

This section describes the Logical Layer Flow Control Extensions CAR and CSR bits that allow an external processing element to determine if a switch or end point device supports the flow control extensions defined in this specification, and to manage the transmission of flow control transactions for a switch processing element. This chapter only describes registers or register bits defined by this specification. Refer to the other RapidIO logical, transport, physical, and extension specifications of interest to determine a complete list of registers and bit definitions for a device. All registers are 32-bits and aligned to a 32-bit boundary.

4.2 Capability Registers (CARs)

4.2.1 Processing Elements Features CAR (Offset 0x10 Word 0)

The Processing Elements Features CAR contains 31 processing elements features bits defined in various RapidIO specifications, as well as the Flow Control Support bit, and Flow Arbitration Participant bit are defined here.

Table 4-1. Bit Settings for Processing Elements Features CAR

Bit	Name	Reset Value	Description
0-19	—		Reserved (defined elsewhere)
20	Flow Arbitration Support	*	Support for flow arbitration 0b0 - does not support flow arbitration 0b1 - supports flow arbitration
21-23	—		Reserved (defined elsewhere)
24	Flow Control Support	*	Support for flow control extensions 0b0 - Does not support flow control extensions 0b1 - Supports flow control extensions
26-31	—		Reserved (defined elsewhere)

* Implementation dependant

4.2.2 Port *n* Control CSR (Block Offsets 0x5C, 7C, ... , 23C)

The Port *n* Control CSR contains 30 bits specifying individual port controls defined in various RapidIO specifications, as well as the Flow Control Participant and Flow Arbitration Participant bits.

Table 4-2. Bit Settings for Port *n* Control CSR

Bit	Name	Reset Value	Description
0-12 (serial)	—		Reserved (defined elsewhere)
13 (serial)	Flow Control Participant	0b0	Enable flow control transactions 0b0 - Do not route or issue flow control transactions to this port 0b1 - Route or issue flow control transactions to this port
14 (serial	—		Reserved (defined elsewhere)
15 (serial)	Flow Arbitration Participant	0b0	Enable Flow Arbitration Transactions: 0b0 - do not route or issue flow arbitration transactions to this port 0b1 - route or issue flow arbitration transaction to this port
16-31 (serial	—		Reserved (defined elsewhere)

Annex A Flow Control Examples (Informative)

A.1 Congestion Detection and Remediation

The method used to detect congestion is implementation specific and is heavily dependent upon the internal packet buffering structure and capacity of the particular switch device. In the example output port buffered switch from "Section 1.1.3, Problem Illustration" on page 10, congestion occurs when some output buffer watermark is exceeded. As long as the watermark is exceeded the output port is said to be in a congested state. The watermark can have different levels when entering the congested state and leaving the congested state.

Fabric elements should monitor their internal packet buffer levels, comparing them on a packet by packet basis to pre-established, locally-defined watermark levels. These levels likely would be configurable depending upon the local element's position within the fabric relative to source endpoints and its particular architecture. On the high watermark side, a level should be selected which is low enough that the remaining buffer space is adequate to provide ample storage for packets in-flight, given a worse-case latency for XOFF CCPs to travel back to the source endpoint and shut off the flow in the endpoint. On the low watermark side (if a watermark is used for XON), a yet-lower level should be selected which meets the following criteria;

a) Provides sufficient hysteresis. When considered in context with the high watermark, it should not be so close as to provide a high flow of XON/XOFF CCP traffic back to the source endpoint.

b) Is set high enough that the switch output buffer does not run dry (underflow) in the typical live-flow case (one or more packets are present in the source endpoint output buffer waiting to be sent when the flow is restarted), given the latency of XON CCP travel back to the source endpoint and restoration of the shut-off flow in the endpoint.

The following two examples are provided to show possible methods for detecting and reacting to congestion:

1. Histogram analysis:
   - — The switch keeps track of packet quantities for the different transaction request flows for which packets are stored in its output buffer.
   - — The switch sorts the transaction request flows according to the number of packets.
   - — The switch selects the 1 to 5 transaction request flows with the most

packets stored in the buffers.

- — The switch sends an XOFF flow control request to those transaction request flow sources when the watermark threshold is exceeded, as long as flow control transaction routing is enabled on that switch port. Handling of system critical flows intending to bypass the flow control operation is outside the scope of this document.
- — The CCP-targeted sources stop transmitting packets for the indicated transaction request flow and all lower priority transaction request flows.
- — The switch sends a flow control XON request to those transaction request flow sources when the watermark drops below the threshold.
- — The CCP-targeted sources begin to transmit packets for the indicated transaction request flow and all higher priority transaction request flows.

2. Simple threshold:
   - — The switch sends an XOFF flow control to the source of every new transaction flow it receives as long as the watermark is exceeded, provided flow control transaction routing is enabled on that switch port. Handling of system critical flows intending to bypass the flow control operation is outside the scope of this document.
   - — The CCP-targeted sources stop transmitting packets for the indicated transaction request flow and all lower priority transaction request flows.
   - — The switch sends a flow control XON request to those transaction request flow sources when the watermark drops below the threshold.
   - — The CCP-targeted sources begin to transmit packets for the indicated transaction request flow and all higher priority transaction request flows.

Note that the first method is reasonably fair in that it targets the source of the data flows that are consuming most of the link bandwidth, and that the second method is unfair in that it indiscriminately targets any source unfortunate enough to have a packet be transmitted while the link is congested.

A.2 Orphaned XOFF Mechanism Description

This timer may take the form of a low precision counter in the end point which monitors the oldest XOFF'd flow at any given time. When a flow first becomes the oldest flow (reaches top of an XOFF'd flow FIFO list within the end point) the timer is reset to its programmed value and begins to count down with time. If it is allowed to elapse without a change to the oldest XOFF'd flow, that flow will be presumed to be orphaned due to lost XON CCP and be restarted as if an XON CCP had been received, with the orphaned flow entry removed from the top of the list and the counter reset to count down for the next oldest XOFF'd flow. The length of the count should be long enough to insure that significant degradation of the flow control function does not occur, on the order of several times the width of the fabric expressed in terms of packet transit time, yet not so large that it would fail to elapse between uncorrelated congestion events. The length of this count shall be programmable through an implementation-dependent register in the end point. The orphaned XOFF mechanism is intended solely as a last-resort mechanism for restarting orphaned flows. It will not be adequate for the purpose of implicit controlled flow reinstatement owing to inherent fairness issues as well as burstyness due to uncontrolled simultaneous multi-flow restart.

A.3 Discussion on Flow Arbitration

The objectives of the flow arbitration protocol are:

1) Conserve resources at the end point, and allow for limited resources to be utilized in a larger system context.
2) Conserve system resources, minimizing protocol overhead.
3) Provide robustness against failures and lost messages as well as provide for low implementation complexity.

Flow arbitration allows for managing resources that are critical to “flows”. A flow is a nexus of a source, a destination, and a physical channel. With the priorities and virtual channels that exist at the physical layer, even a medium system with a few nodes could have 100s or 1000s of flows. Most RapidIO transactions are self contained (as with an IO_WRITE) and thus have a limited “context”. But the data streaming logical type can have a context that spans multiple transactions, and thus needs a persistent resource. The segmentation/reassembly resource is one example of a resource that may be in limited supply in a large system. But, SAR resource management is not the only use of this protocol. Any resource provided by the logical interface to help offload the system may use contexts that span multiple transactions.

As described in the introduction, resources may be managed in three ways:

- The system designer can use end points with enough resources for the worst case combination of flows (fixed)
- The system designer can provide enough resources for the worst case number of flows based on expected traffic, allocating them on a connection by connection basis (static)
- The limited resources can be shared among a larger number of connections on a PDU by PDU basis (dynamic)

It is important to provide some management of resources because an overrun will cause packet loss, at least for the data streaming protocol.

Dynamic allocation is what the protocol defined in this specification provides, but it is not expected to be the sole method of resource allocation. If all the contexts were to use a dynamic protocol, goals #2 and #3, as stated above, might not be met. It is expected that some number of flows will be fixed or static, and only a portion of the lower quality of service flows will arbitrate for some portion of the resources.

System designers are responsible for selecting components that match their strategy for resource management.

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book.

C

Congestion. A condition found in output ports of switch and bridge elements characterized by excessive packet buildup in the buffer, when packet entry rate into the buffer exceeds packet exit rate for a long enough period of time.

CCP. (Congestion Control Packet). A packet sent from the point of congestion in the fabric back to the source endpoint of particular flows instructing the source to either turn on or off the flow.

Controlled Flow List. A memory structure associated with controlling elements which holds a list of currently controlled flows, used by the element to turn back on controlled flows.

CRF. Critical Request Flow. For packets or packets of a given priority, this bit further defines which packet or notice should be moved first from the input queue to the output queue (see *RapidIO Part 4: 8/16 LP-LVDS Physical Layer Specification*, Section 1.2.2 and *RapidIO Part 6: 1x/4x LP-Serial Physical Layer Specification*, Section 5.3.3).

F

flowID. Transaction request flow indicator (see *RapidIO Part 1: Input/Output Logical Specification*, Section 1.2.1).

L

Long Term Congestion. A severe congestion event in which a system does not have the raw capacity to handle the demands placed upon it in actual use.

M

Medium Term Congestion. A congestion event in which a frequent series of short term congestion events occur over a long period of time such as seconds or minutes, handled in RapidIO systems by reconfiguration of the fabric by system-level software.

O

Orphaned XOFF Mechanism. A mechanism in an end point which is used to restart the oldest controlled flow within the end point after a certain period of time has elapsed without the flow being XON'd.

P

Performance Collapse. Non-linear behavior found in non- congestion controlled fabrics, whereby reduced aggregate throughput is exhibited with increased load.

S

Saturation Tree. A pattern of congestion identified within the fabric which grows backward from the root buffer overflow towards the sources of all transaction request flows passing through this buffer.

Short Term Congestion. A congestion event lasting up into the dozens or hundreds of microseconds, handled in RapidIO by Logical Layer Flow Control.

T

Topology. The structure represented by the physical interconnections of a switch fabric.

Transaction Request Flow. A series of packets that have a common source identifier and a common destination identifier at some given priority.

U

Ultra Short Term Congestion. A congestion event lasting from dozens to hundreds of nanoseconds, handled in RapidIO by Link Level Flow Control.

Underflow. A condition within output buffers of switches in which the buffer runs dry.

W

Watermark. A predetermined buffer occupancy level indicating either congestion (high watermark) or abatement of congestion (low watermark).

X

XOFF (Transmit Off). A congestion control packet sent from the point of congestion back to the source of a particular flow, telling the source endpoint to shut off the flow.

XON (Transmit On). A congestion control packet sent from the point of congestion back to the source of a particular flow, telling the source endpoint to restart a controlled flow.