School of Computing
DUNDEE

SpaceNet - SpaceWire-RT

Initial Protocol Definition

Revision: Draft A Issue 2.1
Date: 30th October 2008
ESA Contract Number 220774-07-NL/LvH
Ref: SpW-RT WP3-200.1

Space Technology Centre
School of Computing
University of Dundee
Dundee, DD1 4HN
Scotland, UK
spacetech.computing.dundee.ac.uk

Document Authors

Steve Parkes

Albert Ferrer Florit

Document Change Log

Date	Revision No	Comments
30th October 2008	Draft A Issue 2.1	Update following ESA and SciSys comments (track changes on). Description of complete SpaceWire protocol stack added with service interfaces.
8th September 2008	Draft A Issue 2.0	Update following early prototyping and inputs from SpaceWire Working Group
12th May 2008	Draft A Issue 1.1	Update following comments from ESA
25th March 2008	Draft A Issue 1.0	First issue

A comprehensive list of the changes made to this document in each major revision is provided in section 7.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 AIMS AND OBJECTIVES

The aim of this document is to present an initial set of protocols that meet the SpaceWire-RT Requirements.

The various techniques that are being considered to meet the SpaceWire-RT requirements are considered and described first. Then each function required to implement the protocols is defined in some detail.

WARNING

This current document is an early draft of the proposed standard and is for discussion purposes only. It will change after prototyping work has been completed. Applicable documents may also change.

DO NOT USE THIS DOCUMENT TO DESIGN DEVICES OR SYSTEMS!

1.2 BACKGROUND

The requirements for SpaceWire-RT were provided in WP2-100.1 SpaceWire-RT Requirements [AD8]. Note that RT stands for Real-Time (or alternatively Reliable and Timely). A first draft of the initial protocols was provided in May 2008 (draft issue 1.1). These protocols were presented to the SpaceWire working group for comment and various parts of the protocols prototyped by University of Dundee. The current document has been updated according to the results of the prototyping, taking into account the comments from the SpaceWire working group.

1.3 GUIDE TO DOCUMENT

Section 2 lists the terms and definitions relevant to the SpaceWire-RT set of protocols.

Section 3 provides an overview of the SpaceWire-RT protocols and various techniques that are proposed for their implementation.

Section 4 provides the specification for an asynchronous network.

Section 5 provides the specification for a scheduled network.

Section 6 provides an initial look at the network configuration parameters required for SpaceWire-RT. Substantially more work needs to be done on this section.

1.4 ACRONYMS AND ABBREVIATIONS

AD	Applicable Document
BACK	BFCT Acknowledgement
BFCT	Buffer Flow Control Token
CCSDS	Consultative Committee for Space Data Systems
DP	Data PDU
ECSS	European Cooperation for Space Standardization
ESA	European Space Agency
ESTEC	ESA Space Technology and Research Centre
I/F	Interface
PC	Personal Computer
PDU	Protocol Data Unit
PUS	Packet Utilization Service
QoS	Quality of Service
RD	Reference Document
RMAP	Remote Memory Access Protocol
RMW	Read/Modify/Write
SDU	Service Data Unit
SOIS	Spacecraft Onboard Interface Services
SpW	SpaceWire
TCONS	Time-Critical Onboard Network Services
UDS	User Data Segment
UoD	University of Dundee

1.5 REFERENCE DOCUMENTS

The documents referenced in this document are listed in Table 1-1.

Table 1-1: Reference Documents

REF	Document Number	Document Title
RD1	UoD-SpaceNet v7, 23rd April 2007	Proposal for SpaceWire Network and Future Onboard Data-Handling, Technical, Management and Administrative Proposal
RD2	TEC-ED/WG/2005.15	SpaceWire Network "SpW-Net" SpaceWire and Future Onboard Data Handling SpaceNet Statement of Work Annex1

1.6 APPLICABLE DOCUMENTS

The documents applicable to this document are listed in Table 1-2.

Table 1-2: Applicable Documents		
REF	**Document Number**	**Document Title**
AD1	ECSS-E50-12A, January 2003	SpaceWire: Links, nodes, routers and networks
AD2	ECSS-E50-11A Draft 0.5	SpaceWire Protocols Feb 2008
AD3	CCSDS 850.0-G-R1.1 Draft Green Book	Spacecraft Onboard Interface Service Draft Informational Report Jan 2008
AD4	CCSDS 851.0-R-1.1 Draft Red Book	Spacecraft Onboard Interface Service Subnetwork Packet Service Draft Recommended Practice Jan 2008
AD5	CCSDS 852.0-R-1.1 Draft Red Book	Spacecraft Onboard Interface Service Subnetwork Memory Access Service Draft Recommended Practice Jan 2008
AD6	CCSDS 853.0-R-1.1 Draft Red Book	Spacecraft Onboard Interface Service Subnetwork Synchronisation Service Draft Recommended Practice Jan 2008
AD7	CCSDS 854.0-R-1.1 Draft Red Book	Spacecraft Onboard Interface Service Device Discovery Service Draft Recommended Practice Jan 2008
AD8	CCSDS 855.0-R-1.1 Draft Red Book	Spacecraft Onboard Interface Service Subnetwork Test Services Draft Recommended Practice Jan 2008
AD9	ISO 7498-1 1996	Information Technology – Open Systems Interconnect – Basic Reference Model: The Basic Model Available from http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html
AD10	SpW-RT WP3-100.1	SpaceWire-RT Requirements, February 2008
AD11	ECSS-E-ST-50-11C	SpaceWire Protocols Draft 1.3 July 2008

2 TERMS AND DEFINITIONS

In this section the terms and definitions proposed for the SpaceWire-RT standard are provided in the form required by ECSS.

2.1 DEFINITIONS FROM THE OPEN SYSTEMS INTERCONNECTION (OSI) BASIC REFERENCE MODEL

layer subdivision of the architecture, constituted by subsystems of the same rank

protocol data unit (PDU) unit of data specified in a protocol and consisting of protocol-control-information and possibly user data.

service capability of a layer (service provider) together with the layers beneath it, which is provided to service-users.

service data unit (SDU) an amount of information whose identity is preserved when transferred between peer entities in a given layer and which is not interpreted by the supporting entities in that layer.

2.2 TERMS DEFINED IN THIS RECOMMENDATION

For the purposes of this Recommendation, the following definitions also apply. Many other terms that pertain to specific items are defined in the appropriate sections.

ACK acknowledgement

acknowledgement control PDU used to indicate reception of a data PDU without error and in the correct sequence

asynchronous network a network where there is no control of network bandwidth, packets are sent when generated, subject to prioritisation, provided that the network resource is not busy

asynchronous system system that uses a SpaceWire asynchronous network for communication between elements of the system

BACK control PDU used to indicate reception of a buffer flow control token

BFCT buffer flow control token

buffer flow control token control PDU used to indicate availability of space in a destination buffer

byte 8-bits

channel identifier for network resources comprising source destination buffer, links on path through the SpaceWire network and source buffer

CRC cyclic redundancy code

cyclic redundancy code code used to check for errors in a packet

data PDU PDU containing data user data segment from an SDU

delimited having a known and finite length

destination destination node

destination channel buffer buffer in the destination node associated with a specific channel between a particular source and destination from which data, sent by the source and received at the destination, can be read

destination logical address logical address of the destination node

destination node node that a SpaceWire packet is being sent to

destination subnetwork service access point service access point that identifies the user entity to which a Packet Service SDU is required to be delivered

DLA destination logical address

DP data PDU

DSNSAP Destination Subnetwork Service Access Point

duplicate packet copy of a packet that has already been received once

epoch repeat cycle time for time-slot identifiers

input node node on a SpaceWire network that puts data into a channel for transfer across the network to another node (output node)

maximum data unit maximum size of user data put into a segment by SpaceWire-RT

MDU maximum data unit

output node node on a SpaceWire network that receives data from a channel

packet delimited byte aligned data unit

priority the relative precedence for sending of a data packet relative to other data packets so that a higher priority data PDU normally gets sent before a lower priority one

QoS Quality of Service.

Quality of Service level of service that is requested and provided

Reliable able to be trusted and depended on i.e. SpaceWire-RT will ensure that the data gets to the destination if it is at all possible even if there are transitory errors or even permanent errors provided that the network has some appropriate redundancy

scheduled network synchronous network that uses time-slots for dividing network bandwidth

scheduled system system that uses a SpaceWire scheduled network for communication between elements of the system

segmentation division of service data units by SpaceWire-RT into shorter sections (segments) that are short enough to be sent over the SpaceWire-RT network. SpaceWire-RT is responsible for reassembling the segments back into link service data units at the target.

sequence number incrementing number given to each unique packet sent which is use to detect missing and duplicate packets

service access point an interface to SpaceWire-RT that is identified by an initiator or target address and from which a user application can access the SpaceWire-RT services

SLA source logical address

source source node

source channel buffer buffer in the source node associated with a specific channel between a particular source and destination into which data is written to be sent data across the network from source to destination

source logical address logical address of the source node

source node node that is sending a SpaceWire packet

source subnetwork service access point service access point that identifies the user entity that wishes to access a SOIS service

SSNSAP Source Subnetwork Service Access Point

timely done at an appropriate time or within a specific time constraint

time-slot smallest unit of time division in a SpaceWire-RT network

UDS user data segment

user application a software or hardware system that is using the services of SpaceWire-RT

user data data that a user of SpaceWire-RT wishes to send across a SpaceWire network

user data segment a piece of a SDU that fits into a data PDU

2.3 JUSTIFICATION AND OTHER NOTES ON TERMINOLOGY

2.3.1 Maximum Data Unit

The Maximum Data Unit (MDU) is required to ensure that different sources of data get fair access to the transmission medium, by multiplexing traffic on a packet by packet basis. When a large data unit is being sent, other sources can gain access to the transmission medium after each segment of the large data unit has been sent.

3 PROTOCOL OVERVIEW

In this section an informative description of the SpaceWire-RT protocols is provided as an introduction to the subsequent normative specifications in sections 4, 5, and 6.

3.1 COMMUNICATIONS MODEL

In this section the SpaceWire-RT communications model is described. First the communications model for SpaceWire is reviewed.

3.1.1 SpaceWire

SpaceWire uses point to point links to directly connect one node to another node as illustrated in Figure 3-1. SpaceWire networks can be constructed using wormhole routing switches. Nodes can then be indirectly connected via one or more routing switches as illustrated in Figure 3-2.

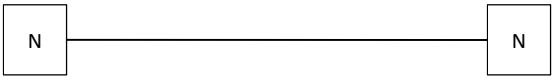

Figure 3-1 SpaceWire Point-to-Point Links

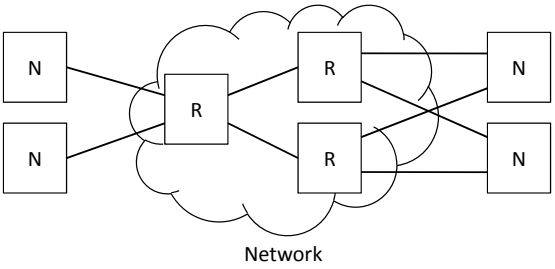

Figure 3-2 SpaceWire Network

SpaceWire provides a stream service. The inputs and outputs to a SpaceWire point to point link are FIFOs. Data presented to an input FIFO is transferred across the SpaceWire link to the other end where it appears in and can be read from an output FIFO. A link level flow control mechanism is defined in SpaceWire which prevents overflow of the output FIFO. When the output FIFO becomes full no more data can be transferred across the SpaceWire link until some data is read out of the output FIFO making space available.

There is no management of memory at the destination i.e. there is no mechanism in SpaceWire to reserve memory or other resources in the destination node. If the destination user decides to stop reading data from an output FIFO the corresponding SpaceWire link will block.

SpaceWire provides a very basic quality of service.

A SpaceWire network provides a means of routing or switching data from a source node to a destination node, using logical addressing or path addressing respectively. The interface to the network and the service provided is the same as for a point to point link. The quality of service is again very basic.

The SpaceWire communications model is streamed data across a SpaceWire point to point link or network.

3.1.2 SpaceWire RMAP

SpaceWire Remote Memory Access Protocol (RMAP) is a service that runs over SpaceWire. It provides a means of reading from and writing to memory in a destination node across a SpaceWire interface. RMAP provides a mechanism for managing access to memory in the remote node, reserving memory prior to writing to it.

3.1.3 SpaceWire-RT

SpaceWire-RT aims to provide a quality of service layer for SpaceWire that gives SpaceWire reliability and timeliness properties. The intention is for this to be done in such a way that any application that used SpaceWire would be able to run over SpaceWire-RT and gain benefit from the QoS provided. Specifically this application would be able to run with other applications which required improved QoS (either reliability or timeliness or both) without impairing the QoS of these other applications. To be able to run existing SpaceWire applications over SpaceWire-RT the interface to SpaceWire-RT has to be conceptually the same as that to SpaceWire. The communications model for SpaceWire-RT is thus one of virtual point-to-point connections, referred to as channels, across a SpaceWire network each of which connects a source channel buffer (input FIFO) in one node to a destination channel buffer (output FIFO) in another node. This is illustrated in Figure 3-3.

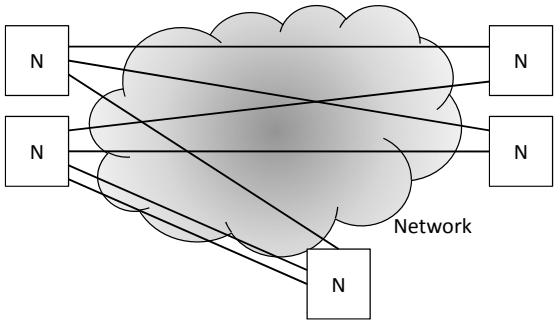

Figure 3-3 SpaceWire-RT Virtual Channels

SpaceWire-RT adds QoS to the SpaceWire paradigm, by providing a stream service over virtual point to point links. Any SpaceWire packet can be sent over a SpaceWire-RT virtual point to point link (channel) receiving the QoS of that channel.

SpaceWire-RT does not provide management of user memory or other resources. It only manages the source and destination buffers (FIFOs).

3.1.4 SpaceWire-PnP

SpaceWire-PnP (Plug and Play) is another protocol which is currently being developed by the SpaceWire Working Group. It provides mechanisms for configuring elements on a SpaceWire network in a consistent way and for discovering the presence on nodes on a network. SpaceWire-PnP has been designed to use the RMAP protocol in order to reduce the amount of hardware (or software) needed to support a complete SpaceWire protocol stack.

3.2 Layered Protocol Stack

SpaceWire-RT is part of the layered protocol stack for SpaceWire which is illustrated in Figure 3-4.

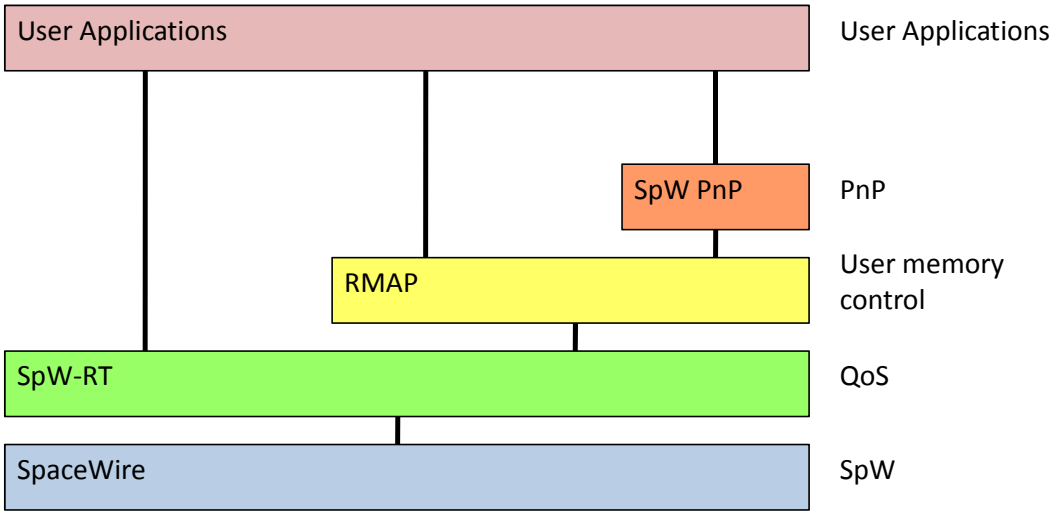

Figure 3-4 SpaceWire Layered Protocol Stack

SpaceWire is at the bottom of the protocol stack sending SpaceWire packets across the SpaceWire network from source to destination. Immediately on top of SpaceWire is SpaceWire-RT providing quality of service. All traffic has to pass through SpaceWire-RT otherwise the reliability and timeliness QoS cannot be ensured. User applications can talk directly to SpaceWire-RT. RMAP provides a mechanism for reading from and writing to memory in a remote node. SpaceWire-PnP (plug and play) uses RMAP for configuration and discovery of nodes on the SpaceWire network. User applications can use the services provided by RMAP or SpaceWire-PnP as well as talking directly to SpaceWire-RT.

The CCSDS Spacecraft Onboard Interface Services (SOIS) working group has defined a set of common communication services for use onboard a spacecraft. The SOIS subnetwork layer and three of the services provided are illustrated in Figure 3-5. The SOIS Packet Service provides for delivery of packets across a subnetwork, the Memory Access Service for the access of memory devices on the subnetwork, and the Device Discovery Service supports plug-and-play capability with notification services.

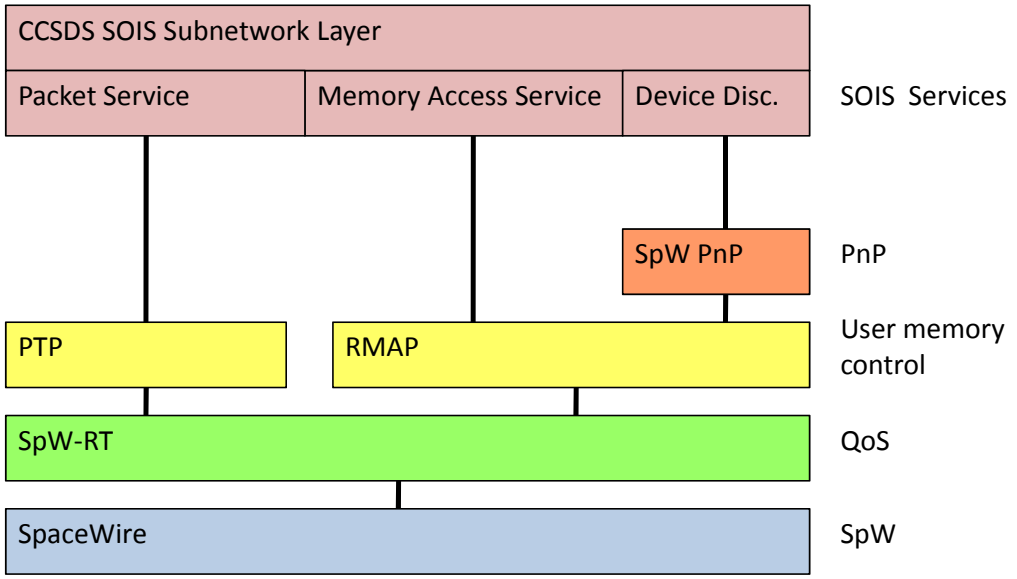

Figure 3-5 SOIS to SpaceWire Layered Protocol Stack

RMAP and SpaceWire-PnP provide the Memory Access Service and Device Discovery Services. The SOIS Packet Service is provided by a SpaceWire Packet Transfer Protocol that sends packets across the SpaceWire network, providing buffer management and flow control.

To send a packet across a SpaceWire network using the CCSDS SOIS Packet Service the packet is presented to the SOIS Packet Service and passed to the SpaceWire Packet Transfer Protocol (SpaceWire-PTP) which manages the destination buffer to avoid sending the packet when there is no room for it in the destination. SpaceWire-PTP passes the packet as a stream of characters to SpaceWire-RT which chops the stream into segments, sends them across the network, and reassembles the SOIS packet at the destination. The SOIS packet emerges from SpaceWire-RT as a stream of characters which are read and put into appropriate packet buffer space by SpaceWire-PTP. The complete buffered packet is then made available to the user application via the SOIS Packet Service interface.

SpaceWire protocols are designed for efficient implementation, high performance and both hardware and software implementation. The SOIS services are designed for generic implementation over various buses or networks, providing a standard software interface.

3.3 SERVICE INTERFACES

In this section the various service interfaces exposed at each layer of the SpaceWire protocol stack are introduced.

3.3.1 SOIS Service Interfaces

The three SOIS services supported by SpaceWire are:

- Packet service
- Memory access service
- Device discovery service

In future the SOIS test and synchronisation services may be included.

The three supported SOIS services are described in relation to SpaceWire in the following subsections.

3.3.1.1 Packet Service

The SOIS Subnetwork Packet Service is defined in AD4. It transfers SDUs from one end-point on a SpaceWire network to another end-point on the same data link/subnetwork, using the SpaceWire Packet Transfer function to move the information across the SpaceWire network.

There are three primitives used by this service:

- PACKET_SEND.request, which requests to send an SDU;
- PACKET_RECEIVE.indication, which indicates that one or more Data PDUs containing an SDU have been received and which passes the SDU to the End System;
- PACKET_FAILURE.indication, which indicates a failure to provide an assured or guaranteed service.

3.3.1.2 Memory Access Service

The SOIS Subnetwork Memory Access Service is defined in AD5. It provides a means for a user entity to retrieve or change data located in memory hosted by a node on a SpaceWire network.

There are five primitives used by this service:

- READ.request which requests to retrieve the contents of memory;
- READ.indication which returns the retrieved contents of memory;
- WRITE.request which requests to change the contents of memory;
- READ/MODIFY/WRITE.request which invokes an atomic Read/Modify/Write cycle at the memory;
- MEMORY_ACCESS_FAILURE.indication which informs a user of the failure of a memory access operation.

3.3.1.3 Device Discovery Service

The SOIS Subnetwork Device Discovery Service is defined in AD6. It provides a means for a user entity to receive notification of data systems' presence on the SpaceWire network.

There are two primitives used by this service:

- DEVICE_DISCOVERY.request which requests that device identities be retrieved from the subnetwork;
- DEVICE_DISCOVERY.indication which returns device identities.

3.3.2 SpaceWire-PTP Service Interface

The SpaceWire-PTP service interface and protocols have yet to be defined. This is an initial definition.

There are three primitives used by this service:

- PACKET_SEND.request, which requests to send a SDU;
- PACKET_RECEIVE.indication, which indicates that one or more Data PDUs containing a SpaceWire-PTP packet have been received and which passes the corresponding SDU to the SpaceWire-PTP user;
- PACKET_FAILURE.indication, which indicates a failure to provide an assured or guaranteed service.

There is a one to one mapping between these primitives and the SOIS Packet Service.

3.3.3 SpaceWire-RMAP Service Interface

The RMAP service interface and protocols are specified in AD11. There are three sets of primitives provided by this service:

- Write memory
- Read memory
- Read-Modify-Write memory

RMAP and SOIS Memory Access service use different paradigms for memory access. RMAP is peer to peer with the memory being accessed treated as a user or application of the SpaceWire network. SOIS Memory Access is asymmetric with the memory being access being treated as part of the subnetwork rather than a unit that uses the SOIS subnetwork services.

The three sets of RMAP primitives are summarised in the following subsections.

3.3.3.1 Write Memory

The RMAP write memory command provides a means for one node, the initiator, to write zero or more bytes of data into a specified area of memory in another node, the target on a SpaceWire network.

The service primitives associated with writing to memory are;

- WRITE.request which is passed from the RMAP Initiator Write service user to the service provider to request that data is written to memory in a target across the SpaceWire network;
- WRITE.confirmation which is passed from the RMAP Initiator Write service provider to the RMAP Initiator Write Service user to confirm that data has been written to memory in a target across the SpaceWire network or to report that an error occurred;
- WRITE.authorisation.indication which is passed from the RMAP Target Write service provider to the RMAP Target Write service user to ask for authorisation to write to memory in the target;
- WRITE.authorisation.response which is passed from the RMAP Target Write service user to the RMAP Target Write service provider to give permission or deny permission to write to memory in the target;
- WRITE.data.indication which is passed from the RMAP Target Write service provider to the RMAP Write service user to write data to memory in the target;
- WRITE.data.response which is passed from the RMAP Target Write service user to the RMAP Write service provider when data has been written to memory in the target;
- WRITE.indication which is passed from the RMAP Target Write service provider to the RMAP Write service user to indicate that data has been successfully written to memory in the target.

3.3.3.2 Read Memory

The read memory command provides a means for one node, the initiator, to read zero or more bytes of data from a specified area of memory in another node, the target on a SpaceWire network. The data read is returned in a reply packet which normally goes back to the initiator.

The service primitives associated with reading from memory are:

- READ.request which is passed by the RMAP Initiator Read service user to the RMAP Initiator Read service provider to request that data is read from memory in a target across the SpaceWire network;
- READ.confirmation which is passed by the RMAP Initiator Read service provider to the RMAP Initiator Read service user to confirm that data has been read from memory in a target across

the SpaceWire network and to provide that data to the RMAP Initiator Read service user, or to report that an error occurred;

- READ.authorisation.indication which is passed by the RMAP Target Read service provider to the RMAP Target Read service user to ask for authorisation to read data from memory in the target;
- READ.authorisation.response which is passed by the RMAP Target Read service user to the RMAP Target Read service provider to give permission or deny permission to read from memory in the target;
- READ.data.indication which is passed by the RMAP Target Read service provider to the RMAP Target Read service user to read data from memory in the target;
- READ.data.response which is passed by the RMAP Target Read service provider to the RMAP Target Read service user to provide data read from memory in the target;
- READ.indication which is passed by the RMAP Target Read service provider to the RMAP Target Read service user to indicate that data has been successfully read from memory in the target.

3.3.3.3 Read-Modify-Write Memory

The read-modify-write command provides a means for one node, the initiator, to read a memory location in another node, the target, modify the value read in some way and then write the new value back to the same memory location. The original value read from memory is returned in a reply packet to the initiator.

The service primitives associated with read-modify-writing memory are:

- RMW.request which is passed by the user of the RMAP Initiator read-modify-write service to the service provider to request that data is read-modify-written to memory in a target across the SpaceWire network;
- RMW.confirmation which is passed by the RMAP Initiator read-modify-write service provider the RMAP read-modify-write service user to confirm that data has been read-modify-written to memory in the target across the SpaceWire network;
- RMW.authorisation.indication which is passed by the RMAP Target RMW service provider to the RMAP Target RMW service user to ask for authorisation to read-modify-write memory in the target;

- RMW.authorisation.response which is passed by the user of the RMAP Target RMW service to the RMAP Target RMW service provider to give permission or deny permission to read-modify-write memory in the target;
- RMW.read.data.indication which is passed by the RMAP Target RMW service provider to the RMAP Target RMW service user to read data from memory in the target;
- RMW.read data.response which is passed by the RMAP Target RMW service provider to the RMAP Target RMW service user to provide the data read from memory in the target;
- RMW.write.data.indication which is passed by the RMAP Target RMW service provider to the RMAP RMW service user to modify and write data to memory in the target;
- RMW.write.data.response which is passed by the user of the RMAP Target RMW service to the RMAP RMW service provided when data and mask have been combined with the data previously in memory and the new result written to memory in the target;
- RMW.indication which is passed by the RMAP Target RMW service provider to the RMAP RMW service user after read-modify-writing memory in the target.

3.3.4 SpaceWire-PnP Service Interface

The service interface for SpaceWire-PnP is currently being defined and will be added to a future revision of this document.

3.3.5 SpaceWire-RT Service Interface

The SpaceWire-RT service interface and protocols are the subject of this document. There are six primitives currently defined for the SpaceWire-RT service:

- Send_Data.request (channel, source address, destination address, priority, cargo) which requests to send a Service Data Unit (SDU) from the source node where the request is being made to a destination node on a SpaceWire network;
- Receive_Data.indication (channel, source address, destination address, priority, cargo) which indicates that a SpaceWire-RT packet has been received and which passes the SDU it carried to the SpaceWire user;
- Notify_Delivered.indication (channel, source address, destination address, SDU_ID) which indicates to the user that issued a Send_Data.request over a channel that provided assured or guaranteed services that the SDU was safely delivered to the destination.

- Notify_Error.indication (channel, source address, destination address, SDU_ID, error metadata) which indicates to the user that issued a Send_Data.request that there was a problem delivering the SDU over a channel that provided assured or guaranteed services.
- Configure.request (channel, configuration information) which configures the channel parameters.
- Redundancy_Invocation.indication (channel, reliability metadata) which indicates that one or more retries or redundancy switching were invoked for a channel.

3.3.6 SpaceWire Service Interface

SpaceWire is specified in the SpaceWire ECSS-E50-12A standard [AD1].

SpaceWire provides seven primitives:

- Send_Packet.request (destination address, cargo) which requests to send a SpaceWire packet from the source node where the request is being made to a destination node on the SpaceWire network;
- Receive_Packet (cargo) which indicates that a SpaceWire packet has been received and which passes the packet cargo to the SpaceWire user;
- Error_Packet (cargo) which indicates that a SpaceWire packet terminated by an EEP has been received and which passes the possibly erroneous and incomplete packet to the SpaceWire user;
- Time-Code Tick-In (time-code value) which requests to send a time-code;
- Time-Code Tick-Out (time-code value) which indicates that a time-code has been received and which passes the value of this time-code to the SpaceWire user.
- Link_Error.indication (error metadata) which indicates that an error has occurred on the link resulting in the disconnection and restarting of the link.
- Link_Configuration.request (configuration information) which sets the SpaceWire link operating parameters.

3.4 Operation of the SpaceWire Protocol Stack

In this section the operation of the SpaceWire protocol stack is illustrated using two examples: reading from memory and transferring a packet.

3.4.1 Example: Packet Transfer

A sequence diagram illustrating how a SOIS packet it transferred across a SpaceWire network using the SpaceWire-PTP, SpaceWire-RT and SpaceWire protocols is shown in Figure 3-6.

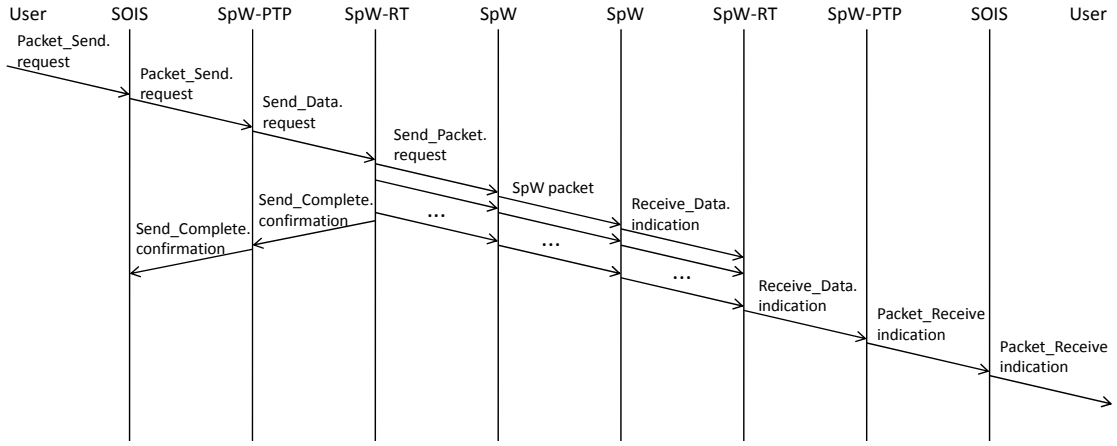

Figure 3-6 Packet Transfer Example – Best Effort Service

The user passes a Packet_Send.request to the SOIS packet service which then passes a Packet_Send.request to the SpaceWire Packet Transfer Protocol. The SpaceWire Packet Transfer Protocol buffers the packet and passes a Send_Data.request to SpaceWire-RT. SpaceWire-RT reads the packet data from the packet buffer, segmenting it and passing a Send_Packet.request for each segment to SpaceWire. SpaceWire sends each SpaceWire packet containing a segment of the PTP packet to the required destination node. Once SpaceWire-RT has passed all the SpaceWire packets to SpaceWire for sending, it passes a Send_Complete.confirmation to SpaceWire-PTP which passes it on to SOIS. When the destination SpaceWire node receives a SpaceWire packet it passes it to SpaceWire-RT using the Receive_Data indication primitive. SpaceWire_RT assembles the PTP packet from the segments received in the SpaceWire packets and indicates that a PTP packet has been received to SpaceWire-PTP with the Receive_Data.indication primitive. SpaceWire-PTP passes a Packet_Receive.indication to SOIS to indicate that a packet has been received and the SOIS packet services passes this primitive up to the user application in the destination node.

The sequence diagram of Figure 3-6 showed the sequence of events for a packet being transferred using the Best Effort QoS. The events that occur when the Assured service is used are illustrated in Figure 3-7.

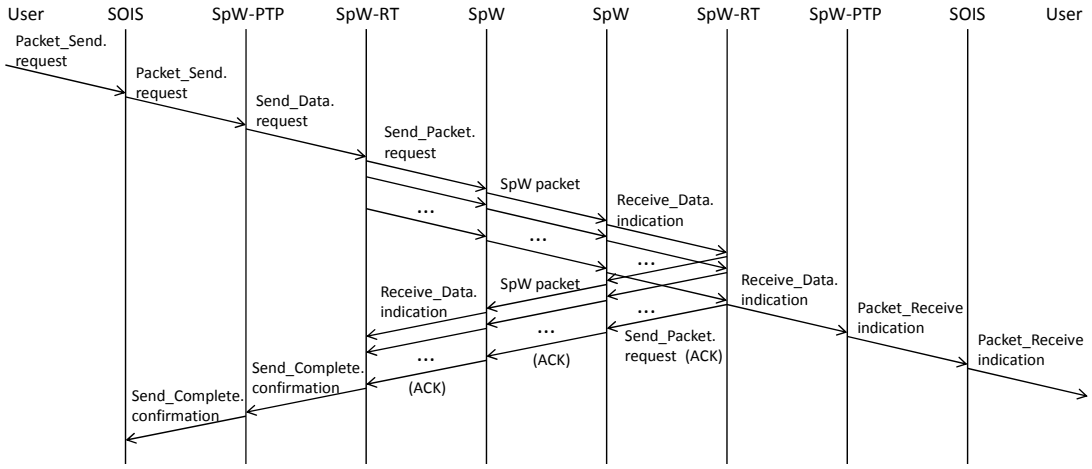

Figure 3-7 Packet Transfer Example – Assured Service

When the SpaceWire packets are received by SpaceWire-RT in the destination node, SpaceWire-RT will generate an acknowledgment for each SpaceWire packet if the assured QoS is being used. The acknowledgement is itself a SpaceWire packet containing the acknowledgement. For each SpaceWire packet received SpaceWire-RT requests SpaceWire to send a SpaceWire packet (containing the ACK). SpaceWire sends these ACK packets back to the source. When the ACK SpaceWire packets are received at the source, SpaceWire sends a Receive_Data.indication containing the ACK to SpW-RT. When the ACKs for all SpaceWire packets containing segments of the PTP packet have been received SpaceWire-RT generates a Send_Complete.confirmation to SpaceWire-PTP which releases its packet buffer and passes the Send_Complete.confirmation on to SOIS, indicating that the packet has been successfully delivered.

3.4.2 Example: Reading from Memory

A sequence diagram illustrating how a SOIS Memory Read operation from a remote node is performed across a SpaceWire network using the SpaceWire-RMAP, SpaceWire-RT and SpaceWire protocols is shown in Figure 3-8.

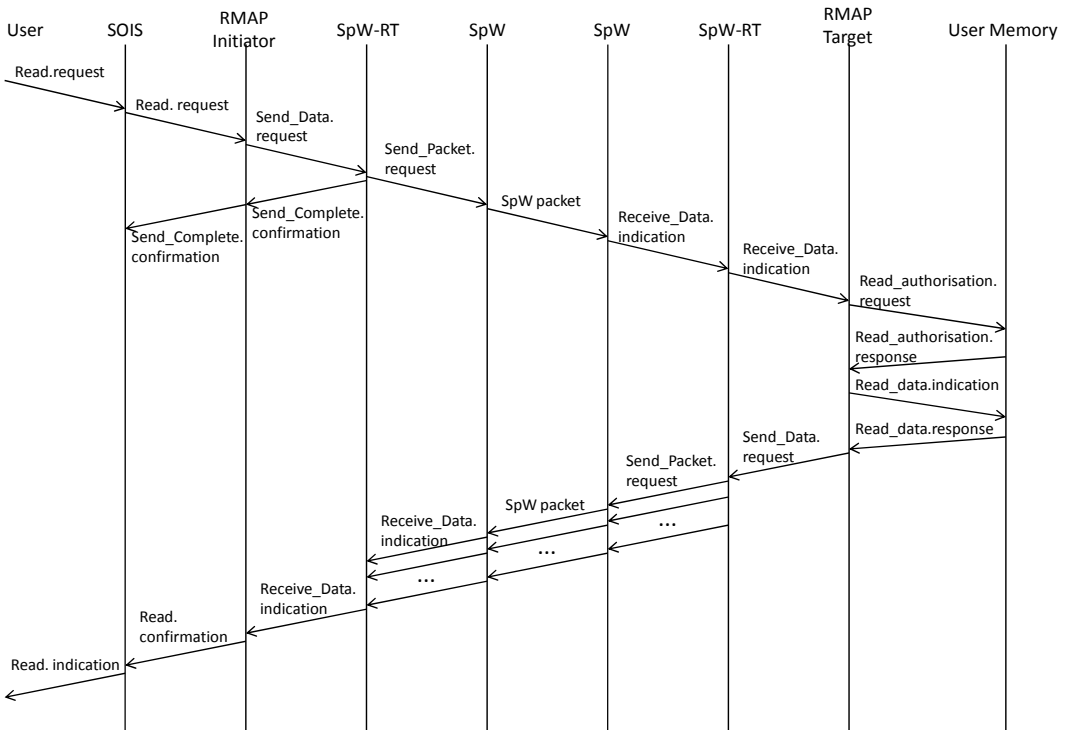

Figure 3-8 Reading from Memory Example – Best Effort Service

The user passes a Read.request to the SOIS Memory Access service which passes the request to RMAP. RMAP forms a read command from the Read.request and passes a Send_Data.request to SpaceWire-RT. Since an RMAP read command fits into a single SpaceWire-RT packet SpaceWire-RT sends a single Send_Packet.request to SpaceWire containing the RMAP read command encapsulated as a SpaceWire-RT packet. SpaceWire sends this packet to the destination. Once SpaceWire-RT has passed the SpaceWire-RT packet to SpaceWire, it passes a Send_Complete.confirmation back to RMAP. RMAP passes the Send_Complete.confirmation on to the SOIS Memory Access Service. When the SpaceWire packet containing the RMAP read command arrives at the destination, it is passed up to SpaceWire-RT (Receive.Data.indication). SpaceWire-RT extracts the RMAP read command and passes it to the RMAP Target. The RMAP Target decodes the read command and asks the User Memory controller to authorise the memory access (Read_Authorisation.request). The User Memory controller responds with a Read.Authorisation.response and the RMAP Target then reads the required data from User Memory (Read_Data.indication and Read.Data.response). The data read is put into an RMAP reply packet and passed from the RMAP Target to SpaceWire-RT (Send_Data.request). SpaceWire-RT segments the RMAP reply packet into one or more SpaceWire-RT packets which are passed (Send_Data.request) to SpaceWire for returning to the source of the RMAP command (Initiator node). SpaceWire transfers the SpaceWire packets to the Initiator node (Initiator node). The SpaceWire

receiver in the Initiator receives these packets and passes them to SpaceWire-RT (Receive_Data.indication). SpaceWire-RT reassembles the RMAP reply packet from the segments received in the SpaceWire packets and passes the complete RMAP reply to the RMAP Initiator (Receive_Data.indication). The RMAP Initiator checks the RMAP reply and passes the data to the SOIS Memory Access service (Read.confirmation). The SOIS Memory Access service then passes this data on the user application.

The sequence diagram of Figure 3-8 showed the sequence of events for a packet being transferred using the Best Effort QoS. The events that occur when the Assured service is used are illustrated in Figure 3-9.

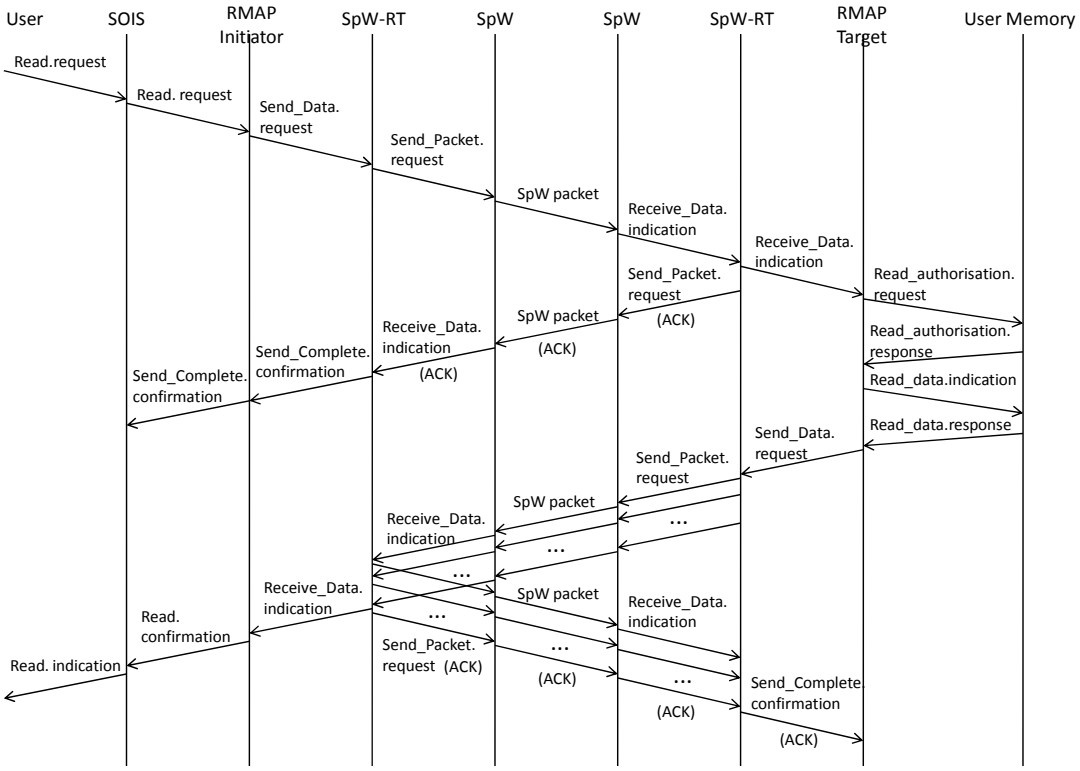

Figure 3-9 Reading from Memory Example – Assured Service

When SpaceWire-RT in the Target receives the SpaceWire packet containing the RMAP command it send an acknowledgement back to the Initiator (Send_Packet.request (ACK)). When the ACK packet arrives at the Initiator the SpaceWire interface passes it up to SpaceWire-RT (Receive_Data.indication (ACK)). SpaceWire-RT decodes the ACK and passes a Send.Complete.confirmation to the RMAP Initiator. The RMAP Initiator passes this on to the SOIS Memory Access service (Send.Complete.confirmation) to affirm that the RMAP command was received without error at the RMAP target node.

When SpaceWire-RT in the Initiator receives the Receive_Data.indications from the SpaceWire interface containing the segments of the RMAP Read Reply, it acknowledges each segment correctly received by asking the SpaceWire interface to send a Send_Packet.request containing an ACK. These ACK packets are sent across the SpaceWire interface to the Target node and passed up to SpaceWire-RT (Receive_Data.indication (ACK)). If ACKs for all segments of the RMAP reply are received, a Send_Complete.confirmation (ACK) is passed to RMAP affirming that the RMAP Read Reply reached the Initiator.

3.5 Asynchronous and Scheduled Systems

There are two types of system that are supported by SpaceWire-RT:

Asynchronous – where the sending of information over the SpaceWire network is asynchronous and priority is used to provide timeliness of delivery. Information in lower number source channels will be sent before information in higher number channels.

Scheduled – where information is sent over the SpaceWire network synchronously with each source channel being assigned one or more time-slots when it is allowed to transmit information. One or more source channels may be assigned to a single time-slot in which case the lower channel numbers allocated to a time-slot have priority over higher channel numbers assigned to that time-slot. Timeliness of delivery is controlled by a schedule table used to specify which source channel can send information in which time-slot. This provides deterministic delivery.

A user application writes into one of the source channel buffers available. This information is then transferred across the SpaceWire network and becomes available in the corresponding destination channel buffer. SpaceWire-RT provides a stream service so that user data written into a source channel buffer appears in and can be read from the corresponding destination channel buffer. This is conceptually the same as a SpaceWire point-to-point link where data written into the source transmit FIFO appears in the destination receive FIFO.

3.6 Quality of Service

SpaceWire-RT provides a quality of service layer for SpaceWire.

3.6.1 SpaceWire-RT QoS Classes

SpaceWire-RT provides five quality of service (QoS) classes:

- Basic
- Best Effort
- Assured

- Resource-Reserved
- Guaranteed

The **Basic QoS** provides a service which does not ensure delivery (i.e. does not provide any redundancy and does not retry in the event of a failure to deliver) and is not timely (i.e. does not deliver information within specified time constraints).

- Makes a single attempt to deliver data to its destination but cannot ensure that it will be delivered successfully.
- Data is provided without errors i.e. a data PDU that arrives with an error is discarded.
- All data PDUs that are received are delivered regardless of their order.
- Duplicate data PDUs are delivered.
- Priority indicates the relative precedence with which data PDUs are handled by the sub-network. Priority is applied across the basic, best-effort and assured service classes where these classes are provided. Higher priority data PDUs get sent before lower priority ones, provided that there is room in the higher priority destination buffers.

The **Best Effort QoS** also provides a service which does not ensure delivery (i.e. does not provide any redundancy and does not retry in the event of a failure to deliver) and is not timely (i.e. does not deliver information within specified time constraints).This service does not deliver duplicate or out of sequence packets.

- Makes a single attempt to deliver data to its destination but cannot ensure that it will be delivered successfully.
- Data is provided without errors, i.e. a data PDU that arrives with an error is discarded.
- The order of data PDUs is preserved (within a priority value). Any out-of-order DPs are discarded.
- Data is provided without duplication, i.e. a data PDU that is a duplicate of a previous packet is discarded.
- Priority indicates the relative precedence with which data PDUs are handled by the sub-network. Priority is applied across the basic, best-effort and assured service classes where these classes are provided. Higher priority data PDUs get sent before lower priority ones, provided that there is room in the higher priority destination buffers.

The **Assured QoS** provides a service which is reliable (i.e. retries in the event of a failure to deliver) but is not timely.

- Ensures delivery of data to its destination.
- Notification is given to sending entity that data was delivered.
- When confirmation of delivery cannot be provided this is indicated to the sending entity.
- Data is provided without errors.
- The order of data PDUs is preserved (within a priority value).
- Data is provided without duplication.
- Priority indicates the relative precedence with which data PDUs are handled by the sub-network. Priority is applied across the basic, best-effort and assured service classes where these classes are provided. Higher priority data PDUs get sent before lower priority ones, provided that there is room in the higher priority destination buffers.

The **Reserved QoS** provides a service which does not ensure delivery (i.e. does not provide any redundancy and does not retry in the event of a failure to deliver), but is timely (i.e. when a packet is delivered it is delivered on time).

- Makes a single attempt to deliver data to its destination but cannot ensure that it will be delivered successfully.
- Data is provided without errors, i.e. a data PDU that arrives with an error is discarded.
- The order of data PDUs is preserved (within a priority value). Any out-of-order DPs are discarded.
- Data is provided without duplication, i.e. a data PDU that is a duplicate of a previous packet is discarded.
- Resources are reserved for transferring the data across the network.

The **Guaranteed QoS** provides a service which is both reliable and timely (i.e. it will retry in the event of a failure to deliver and deliver information on time).

- Ensures delivery of data to its destination.
- Notification is given to sending entity that data was delivered.
- When confirmation of delivery cannot be provided this is indicated to the sending entity.
- Data is provided without errors.
- The order of data PDUs is preserved (within a priority value).
- Data is provided without duplication.

- Resources are reserved for transferring the data across the network.

3.6.2 QoS in an Asynchronous Network

An asynchronous network supports three QoS classes:

- Basic
- Best Effort
- Assured

3.6.3 QoS in a Scheduled Network

A scheduled network supports five QoS classes:

- Basic Allocated or Opportunistic
- Best Effort Allocated or Opportunistic
- Assured Allocated or Opportunistic
- Resource-Reserved
- Guaranteed

The three opportunistic QoS classes do not have network bandwidth specifically scheduled for the data transfer. They make use of any appropriate unused bandwidth available in the network. The resource-reserved and guaranteed QoS classes have network bandwidth specifically allocated to them.

[XXX: The three allocated classes - what is the difference between allocated BE and RR and between Guaranteed and allocated assured? Need to flesh this out]

3.7 CHANNELS

A channel is a set of network resources that connects a source user application in a source node to a destination user application in a destination node. It includes the following:

- Buffer in the source into which data is written to send data across the network (source channel buffer)
- SpaceWire links over which the data PDUs travel

- Buffer in the destination from which data that has been received over the network can be read (destination channel buffer)

A channel connects a source channel buffer (the channel input) to a destination channel buffer (the channel output) via one or more SpaceWire links. Thus a channel is identified by the source logical address (SLA), the destination logical address (DLA) and a channel number. The channel number is used so that there can be more than one channel between a particular source and destination pair.

A channel is unidirectional sending data in one direction only from source to destination, although some control information for the channel does flow in the opposite direction. To provide bi-directional communication two unidirectional channels are required; one in each direction. A node can have one channel going to just one destination node, many channels each going to a different node, many channels all going to the same node, etc. Channels provide virtual unidirectional, point-to-point communications across a SpaceWire network. The entry to a channel in a source node is where data is fed in by a user application. That data will appear at the exit of the channel in the destination node.

A source user application that wants to send information over a channel to a destination user application writes data into the appropriate source channel buffer when there is space in that buffer. Data from this buffer is taken out in chunks with each chunk (referred to as a segment) being put in a separate SpaceWire packet. The SpaceWire packets are sent across the SpaceWire network using the links specified by the channel. When they arrive at the destination node the user information in the SpaceWire packets is extracted and put in the destination channel buffer ready for the destination user application to read. When there is no room in the destination channel buffer the source node is prevented from sending any further packets to that destination channel buffer using a flow-control mechanism.

Up to 256 channels can be implemented between each source and destination pair. It is not necessary for a node to support all of these channels. A node only has to support the number of channels that it needs. Similarly a destination need only accept data from the sources that it expects to receive data from.

Priority is implemented using more than one channel between a source and destination. Information placed in the lower number channel will be transferred before information in a higher number channel.

An example of several channels between three nodes is illustrated in Figure 3-10.

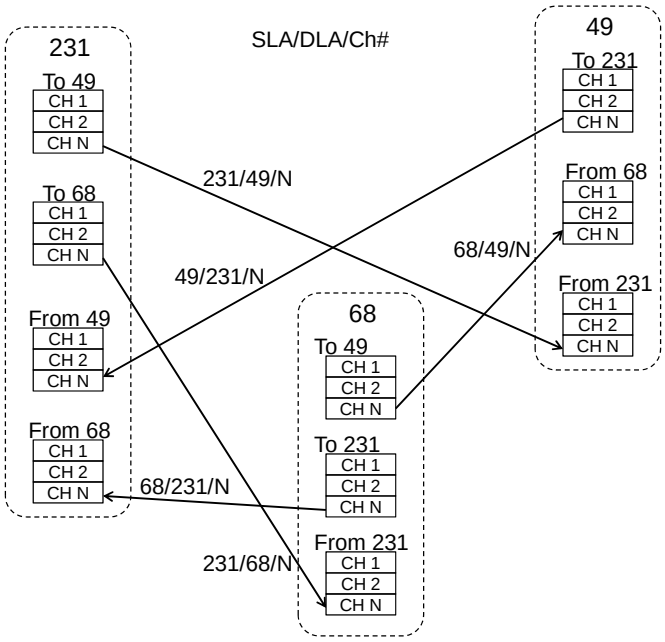

Figure 3-10 SpaceWire-RT Channels

There are three nodes with SpaceWire logical addresses 49, 68 and 231, represented by the dashed rectangles. The source channel buffers in each node are grouped according to the destination for each channel. For example, node 68 has its source channel buffers to node 49 grouped under the heading “To 49”. Similarly, the destination channel buffers in each node are grouped according to the source for each channel. For example, node 49 has its destination channel buffers from node 68 grouped under the heading “From 68”. There are N unidirectional channels available from node 49 to node 68. Only the last one of these is indicated with an arrow from node 68 to node 49 for clarity. Source channel buffer 1 is connected to destination channel buffer 1, 2 to 2 and so on. The arrow between node 68 and 49 is labelled with the source logical address, destination logical address and channel number (SLA/DLA/CH#) which provides a network wide unique reference for that channel.

A pair of channels has been set up between node 49 and 231 using two unidirectional channels (231/49/N and 49/231/N). Similarly another pair of channels has been configured between nodes 68 and 231 (231/68/N and 68/231/N).

3.8 SHARED RESOURCES

Timely delivery of information over a SpaceWire network requires control over the resources in the network. This section considers the resources that need to be managed.

3.8.1.1 SpaceWire Links as Resources

The resources to be managed by the SpaceWire-RT resource reservation mechanism are the SpaceWire links and the source and destination channel buffers. SpaceWire routers need not be managed by the resource reservation mechanism as they are non-blocking switches: provided that there are no conflicts on the SpaceWire links the routers will not block. A SpaceWire router is able to route packets from any input port to any output port provided that the output port is not currently being used to send another packet. It does not matter what the other input and output ports are doing, if the required output port is not currently being used then an input port can route a packet to that output port. If the SpaceWire links providing data to an input port of a router and taking data from an output port are both reserved then the router will be able to handle the transfer of data and so need not be reserved itself: it is the ports of the router that need to be reserved not the switch and since the ports are connected one-to-one to SpaceWire links reserving the links reserves the necessary routing resources.

Source and destination nodes need not be managed by the resource reservation mechanism as they are effectively managed by managing the SpaceWire links to the source or destination. For example, a SpaceWire node can happily receive two packets concurrently, provided that they arrive on separate SpaceWire links. It is, however, important that the source and destination channel buffers are managed.

3.8.1.2 Example Onboard Data-Handling Architecture

To help understand the resource reservation, consider the typical SpaceWire based data handling architecture shown in Figure 3-11.

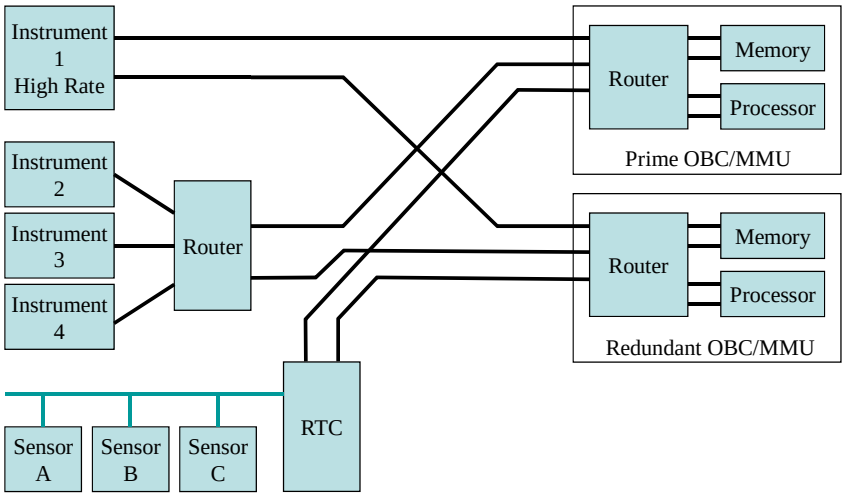

Figure 3-11 Typical SpaceWire based data-handling architecture

This diagram shows both prime and redundant onboard computer / mass memory units (OBC/MMU) and SpaceWire links. For the sake of clarity these redundant units have been removed in Figure 3-12 and logical address have been assigned to nodes e.g. the memory is assigned LA70.

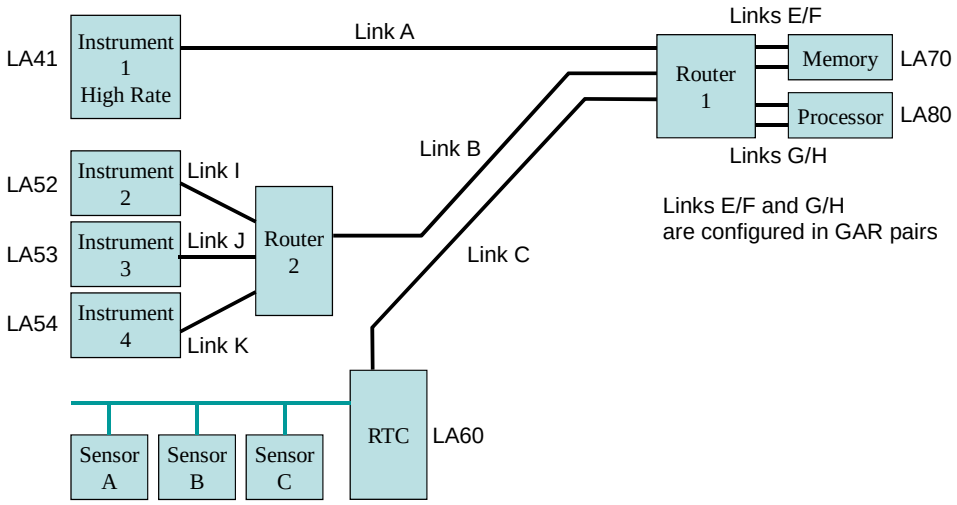

Figure 3-12 Typical architecture with redundancy removed for clarity

Two SpaceWire links are used between Router 1 and the memory to increase the available bandwidth rather than for redundancy. These two links are configured in the Router as a group adaptive routing (GAR) pair. Similarly there are two SpaceWire links configured as a GAR pair to increase bandwidth between Router 1 and the processor.

As an example let's assume the following use case parameters and features:

- SpaceWire links are all running at 200 Mbits/s data signalling rate and, allowing for SpaceWire overheads and some margin, assume that the effective maximum data rate is 100 Mbits/s over each link in both directions.
- Instrument 1 is a high data rate instrument that has to send data to the memory at a rate of up to 100 Mbits/s to the memory unit.
- Instruments 2 sends data at up to 25 Mbits/s to the memory unit.
- Instruments 3 sends data at up to 12.5 Mbits/s to the memory unit.
- Instrument 4 sends data at up to 12.5 Mbits/s to the processor for processing. The processor then sends this data to the memory system at a similar data rate.
- The RTC gathers data from several sensors, packages this data, and passes it to the memory unit. The required data rate is relatively low.

- The memory unit stores data from the instruments including the processed data from instrument 4.
- The processor is responsible for routine control of the instruments and for processing the data from instrument 4. The required data rate is relatively low.

Note that all of these data transfers are uni-directional.

Table 3-1 lists how each of the SpaceWire links are used in the example data handling architecture.

Table 3-1 Utilisation of resources (links)

Link	Left to right / up	Right to left/ down
A	Not shared	Processor commands and ground commands
B	Instruments 2, 3, 4	Processor commands and ground commands
C	RTC	Processor commands and ground commands
D	not defined	
E/F	Instruments 1, 2, 3, 4 and RTC Processor commands and ground commands	Date from memory for down link
G/H	Data from instruments or memory for processing	Processor commands Processed data
I	Not shared	Processor commands and ground commands
J	Not shared	Processor commands and ground commands
K	Not shared	Processor commands and ground commands

The highlighted table entries are those ones where potential conflict can occur because there are several possible units that want to use the particular SpaceWire link. Note that since SpaceWire is a full-duplex, bi-directional data link the two directions of each link are considered separately.

Now consider how the resource reservation can be used to manage the flow of data across the links to avoid potential conflict over the use of the links. Table 3-2 shows example channel allocations for the various types of communication that goes on in the data handling system. Note: uppercase notation is used for links containing information flowing in one direction and lowercase used for information flowing in the opposite direction.

Table 3-2 Channel allocations			
Channel No.	Traffic	Links used L to R / Up	Links used R to L / Down
41/70/1	Instrument 1 to memory	A, E/F	
52/70/1	Instrument 2 to memory	I, B, E/F	
53/70/1	Instrument 3 to memory	J, B, E/F	
54/80/1	Instrument 4 to processor for processing	K, B, G/H	
60/70/1	RTC sensor data to memory	C, E/F	
80/70/1	Processor to memory – processed data	E/F	g/h
80/xx/1	Processor commands to any other unit	E/F	g/h, a, b, c, i, j, k

Please note that channel number 80/xx/1 includes communications from processor to instrument 1 (80/41/1 : g/h, a), processor to instrument 2 (80/52/1 : g/h, b, i), processor to instrument 3 (80/53/1 : g/h, b, j), processor to instrument 4 (80/54/1 : g/h, b, k), processor to RTC (80/60/1 : g/h, c) and processor to memory (80/70/1 : g/h, E/F).

3.8.1.3 Time-Slots

Scheduling requires a means of controlling the traffic on the network. This is done by splitting using equal divisions of time during which a discrete set of network communications can take place. These divisions of time are known as time-slots. Time-slots are distributed in SpaceWire using SpaceWire time-codes. There are 64 unique time-codes so a natural division is to have 64 time slots in a schedule or bandwidth allocation cycle. The 64 time-slots are referred to as an epoch. Time slots are used in the scheduled system for allocating network bandwidth to the network traffic.

The frequency of the time-codes depends upon the system requirements. This is discussed later in 3.9.3.4

3.8.1.4 Scheduled System

Once the channels have been defined and the resources needed by each channel determined it is a relatively straightforward exercise (for the example) to allocate the channels to a time-slot in a scheduled system, taking into account the required maximum data-rates that have to be supported for each unit. An example schedule is illustrated in Figure 3-13.

	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8	…	Slot 63
41/70/1	A, E/F	A, E/F	A, E/F	A, E/F	A, E/F	A, E/F	A, E/F	A, E/F			A, E/F
52/70/1	I, B, E/F				I, B, E/F						
53/70/1		J, B, E/F									
54/70/1			K, B, G/H								
60/60/1				C, E/F							
80/70/1						g/h, E/F					
80/xx/1							E/F, g/h, a,b,c,I,j,k				

Figure 3-13 Slot allocation in scheduled system

The time-slots are listed along the top. In this example eight slots have been filled in which are repeated eight times for the full 64 time-slots. The nine channels are listed along the left hand side. The shaded boxes show when a particular channel is allowed to send data. For example Channel 52/70/1 is permitted to send data in time-slots 0, 4, 8, 12, etc. Inside each box the resources needed for the data transfer are listed. For example Channel 52/70/1 requires links I, B and E/F. Note that E/F means either link E or link F since these two links are considered as a group adaptive routing (GAR) group.

In any one time-slot several communications may be happening concurrently provided that they do not conflict i.e. two or more channels communicating in any one time-slot do not require the same resources. When GAR is being used this constraint alters as the bandwidth available increases with the number of links in a group. In our example there are two links (E and F) in the E/F group. This means that in any one time-slot, up to two channels may be using group E/F. Since the links are full-duplex, bi-directional data flowing in one direction does not impede data flowing in the opposite direction. Hence lower case link resources (one direction) do not conflict with uppercase link resources (other direction).

As may be seen from the schedule table channel 41/70/1 may send data at any time. Channel 52/70/1 has 25% of the link bandwidth (one slot every four) and channel 53/70/1 has 12.5% bandwidth (one slot every eight).

When a particular time-slot comes around then the channels that are scheduled to send data in that time-slot are allowed to send one or more data PDUs provided that they fit within the duration of the time-slot. A time-slot is defined by SpaceWire-RT to be long enough to allow six packets of maximum permitted length to be sent in one time-slot.

3.8.1.5 Resources for Flow Control and Acknowledgements

As well as the main flow of DPs containing data, there will be traffic in the other direction containing acknowledgments and flow control information related to the transfer of the DPs.

In an asynchronous network where there is no control over resource utilisation, acknowledgments and flow control information can be sent without regard to possible delays over the SpaceWire network as timely delivery is not covered by the Basic, Best Effort and Assured QoS supported by an asynchronous network.

For a scheduled network where the resource utilisation is managed using time-slots, it is important that resource is allocated for the acknowledgements and also for any flow control information. To provide this additional resource the time-slots are made longer than needed to transfer the DPs so that there is time for acknowledgement and flow control information to be sent at the start of a time-slot. Time-slots are thus split into three parts:

- Acknowledgement phase
- Flow control phase
- DP transfer phase

During the DP transfer phase DPs are sent according to the schedule. During the acknowledgment phase acknowledgments are sent for the DPs transferred in the transfer phase, along with any flow control information.

To ease implementation the information is actually transferred in time-slots as follows:

1. Time-code received
2. Send acknowledgement for DPs sent in previous time-slot
3. Wait a bit for acknowledgements to propagate across the network
4. Send flow control information
5. Wait a bit for flow control information to propagate across the network
6. Send DPs
7. Complete sending DPs before end of time-slot
8. Wait for next time-code

3.8.1.6 Basic, Best Effort and Assured Traffic over a Scheduled System

Data sent with basic, best effort and assured traffic non-reserved QoS are ad hoc and do not require reservation of network resources (non-reserved) however they still have to be scheduled into time-slots in a scheduled system.

There are three possible ways to send non-reserved data over a scheduled network:

- **Allocated** - time-slot allocated to a source-destination pair
- **Opportunistic** – opportunistic use of otherwise unused time-slots
- **Master** - time-slots that take control of the whole network

One or more of these ways of sending non-reserved data over a scheduled network may be used as required. This is set up by the network manager.

3.8.1.6.1 Allocated

It is possible to use a time-slot to send any non-reserved information between a specific source and destination. In this case any of the channels between the source and destination allocated to the time slot are allowed to send data. The ones that are selected will depend upon the channel QoS, specifically the priority. A channel with high priority which has data ready to send will be able to send its data before a channel with lower priority.

In the allocated scheme multiple source-destination pairs may be allocated to the same time-slot provided that they do not use any common network resources – i.e. that they do not need to send data PDUs over the same SpaceWire links.

3.8.1.6.2 Opportunistic

A time-slot allocated to a channel with resource-reserved or guaranteed QoS is potentially wasted if there is no data to transfer over that channel during the time-slot. This idle time-slot can be used to transfer data over non-reserved QoS channels between the same source and destination pair as the resource-reserved or guaranteed channel to which the time-slot was allocated. Since the same source and destination pair is being used there is no possible network conflict that can arise from the opportunistic use of the idle time-slot. The time-slot will then be used by the resource-reserved channel when it has data to send or by one of the non-reserved channels between the same source and destination pair when there is no data available to be sent by the resource-reserved channel.

3.8.1.6.3 Master

A third form of use of time-slots is the master time-slot. In this case a single node is given access to the whole scheduled network for the duration of the time-slot. It is able to send and receive data from any node provided that the transaction can be completed in the one time slot. The basic QoS is the only level of service provided.

Master time-slots are specifically designed for use during network configuration and management. During a master time-slot a network manager can send network configuration packages to any node

in the system and receive any replies expected. It is up to the network manager to provide any additional QoS characteristics required above the basic QoS.

In a master time-slot there are no acknowledgements (since only the basic level of service is supported) and no flow control support is provided. It is assumed that the destination channel buffer is always ready to receive a configuration command and if it is not the packet is discarded.

3.9 ARCHITECTURE

The SpaceWire-RT protocol includes the following functions:

- User application interface
- Segmentation
- End to end flow control
- Retry
- Error detection
- Redundancy
- Address translation
- PDU encapsulation
- Priority
- Resource reservation
- Network management

Each of these functions is described in the following subsections. The functions that are common to asynchronous and scheduled networks are described first. This is followed by the functions specific to an asynchronous network and then those specific to a scheduled network. Finally network configuration is described.

3.9.1 Common Functions

3.9.1.1 User Application Interface

SpaceWire-RT provides a quality of service layer for a SpaceWire network. Any SpaceWire application is able to run over SpaceWire-RT and benefit from the QoS provided. To achieve this SpaceWire-RT uses the same conceptual model for the interfaces as SpaceWire. Data to be transferred over a SpaceWire network is put into a transmit FIFO at the source and appears in a receive FIFO at the destination, following address information provided in the SpaceWire packet

header, and with packets terminated by an EOP. The interface is a FIFO or stream type interface. SpaceWire-RT uses the same model it provides stream interfaces using source and destination buffers which effectively act as FIFOs.

The user application interface to SpaceWire-RT is via the source and destination channel buffers. The source user application writes information to be transferred across the SpaceWire network into a source channel buffer. This is then readout by SpaceWire-RT, and transferred across the SpaceWire network to the destination channel buffer associated with the source channel buffer. The destination user application can then read the information from the destination channel buffer.

There are four pieces of information that the source user application has to pass to the source channel buffer:

- **Destination** – the logical address of the destination to which the information is to be sent.
- **Channel number** – the number of the source and destination channel buffers to which the data to be transferred is to be written. The channel used will also determine the quality of service provided, since QoS parameters are associated with each channel.
- **Data** – the data or other information that is to be transferred to the destination channel buffer.
- **Separator** – that separates out one complete piece of user application information from the next piece of user information being sent over a channel. For example a complete piece of information may be a CCSDS PUS packet or an RMAP command. SpaceWire-RT will send information provided to a source channel buffer in segments when there is room in the destination channel buffer. Normally it sends segments of information that fill the maximum permitted DP. The last part of a complete piece of user information may not fill a DP, but should be transferred as soon as possible, without waiting for further user information to fill a maximum size DP. The separator is used to signal to SpaceWire-RT that the information in the source channel buffer is to be sent straightaway without waiting for a full DP's worth of data. The separator is equivalent to the EOP in SpaceWire.

The destination channel buffer provides a similar interface to the destination user application with five pieces of information:

- **Indication** – An indication that a new piece of user information has started to arrive on a particular channel. The indication contains the source logical address and channel number which defines the destination channel buffer the information is being received in.
- **Source** – The logical address of the source that the destination user application wants to read data from.

- **Channel number** – The channel number that the destination user application wants to read data from. Together the source and channel number identify the particular destination channel buffer from which data is to be read.
- **Data** – The data or other information that has been received from the related source channel buffer and placed in the destination channel buffer which is being read by the destination user application.
- **Separator** – An indication of when the last of the complete piece of user information from the source channel buffer has been read out of the destination channel buffer by the destination user application.

3.9.1.2 Segmentation Function

User information is passed to SpaceWire-RT for sending across a SpaceWire network. The size of this user information is arbitrary and unknown to SpaceWire-RT. SpaceWire-RT sends information across the SpaceWire network in protocol data units (DPs) each with a size up to a specific maximum DP size. To fit the user information into one or more DPs it has to be split into segments that fit into the available space in the DPs. These segments of user data are the user data segments (UDS) accepted from and delivered to the user application. The maximum size of the UDS is less that the maximum DP size because the DP will also contain other information concerned with delivering the DP to its intended destination.

The segmentation function is responsible for splitting up the user information into user data segments no larger than the maximum user data segment (UDS) size. The maximum UDS size is 256 bytes (data length = 0, means 256 bytes). This allows a single byte to be used to specify the data length in a DP and keeps buffer memory requirements low. DPs with data length less than the maximum UDS size are allowed. A service data unit (SDU) greater than 256 bytes in size will be segmented into one or more DPs each containing 256 bytes of data followed by the last DP containing 256 or fewer bytes of data. The minimum UDS size is 1 byte.

3.9.1.3 Address Translation

SpaceWire can provide up to 223 logical addresses, permitting up to 223 separate nodes. This number is adequate for most foreseen space missions, so for SpaceWire-RT node identification uses the SpaceWire logical address. Path addressing may be used to route a packet to its destination but the node identification is done using the logical address.

A SpaceWire logical address is used to uniquely identify a node attached to the SpaceWire network. A node may be identified by more than one SpaceWire logical address, but there is only one node that has a specific logical address. For example node A can be identified by SpaceWire logical addresses 124 and 125, but logical address 132 cannot be used to identify both node B and node C

(at the same time). Note that is SpaceWire logical addresses are assigned to nodes not to SpaceWire ports (links) attached to a node. For example, if a node has logical address 212 and if this node has three SpaceWire ports, then SpaceWire packets can be routed to any of these three ports using the logical address 212.

The address translation function translates from the SpaceWire logical address to the SpaceWire address that will be used to send the packet across the network. The SpaceWire address can be a path, logical, or regional logical address or an address constructed using any combination of these addressing modes. The type of address used will be dependent upon the redundancy approach being used. Address resolution is used to determine the SpaceWire address bytes that are included in the header of the SpaceWire packet to route it along the required path across the SpaceWire network to its intended destination.

An example address translation scheme using look-up tables is shown in Figure 3-14.

SpaceWire Logical Address	Priority	Prime SpaceWire Address	Redundant SpaceWire Address
120	-	120	120
124	-	1, 6, 5, 2, 124	2, 3, 5, 2, 124
132	low	132	132
132	high	133	133
150	-	1, 132	2, 132

Figure 3-14 Example Address Translation

In Figure 3-14 there are two columns containing SpaceWire addresses: the prime SpaceWire address and the redundant SpaceWire address columns. These are used to support autonomous redundancy switching (see section 3.9.1.5). Normally the prime SpaceWire address is used, but if there is an error on this route through the network the redundant SpaceWire address can be used which will route packets along an alternative route through the network to the destination. Further columns may be added if additional alternative routes through the network are to be provided.

SpaceWire logical address 120 identifies a node which is to be sent information using a SpaceWire logical address. In this case the prime and redundant SpaceWire addresses are the same (both 120). Redundancy switching would have to be done by reconfiguration of the routing tables in the routers in the SpaceWire network. A managed redundancy approach is being used rather than an autonomous one.

SpaceWire logical address 124 identifies a node which is to be sent information using SpaceWire path addresses. There are two different routes through the network identified: a prime and a redundant one.

SpaceWire logical address 132 identifies a node which is to be sent information using a SpaceWire logical address. The priority arbitration scheme of the SpW-10X router device is being used to provide preferential routing of packets destined for this node. SpaceWire logical address 132 is set up in the routers with low priority and address 133 with high priority. Both have the same routing information. When information is to be sent using the high priority route the priority parameter is set high and SpaceWire address 133 is used. A managed approach to redundancy switching is used.

SpaceWire logical address 150 identifies a node which is to be sent information using a SpaceWire logical address. In this case autonomous redundancy switching is to be used. To accommodate this the port that is used to start a packet on either the prime or redundant path is included in the SpaceWire address. Thereafter the routing tables in the routers route the packet through alternative paths through the network to the destination.

When simultaneous retries are used (see section3.9.1.4) then the information is sent in two packets at the same time one using the prime SpaceWire address and the other using the redundant SpaceWire address.

It is possible to have more levels of redundancy than prime and redundant. In fact the Address translation table may be used to translate from spacecraft state and network redundancy information to a path through the SpaceWire network from source to destination. This allows, for example, different paths to be used during spacecraft “safe-mode” than during normal operation. The network management function (section 3.9.3.4) is responsible for configuring and maintaining the address translation tables in all nodes.

For acknowledgements and buffer flow control tokens the flow of information is in the opposite direction to the flow of DPs for a specific channel. The return path for the acknowledgements and buffer flow control tokens is normally the mirror of the sending path i.e. they follows the same path but in the opposite direction.

3.9.1.4 Error detection

Error detection is needed to support the retry functions (defined separately for asynchronous and scheduled networks) and also for error notification for the Basic, Best Effort and Resource Reserved classes of traffic. There are six possible types of error:

- Packet received with header error i.e. the header CRC has detected an error in the header.
- Packet delivered to wrong destination i.e. the destination SpaceWire logical address does not correspond to the SpaceWire address of the node that it has been delivered to.

- Packet received with data error i.e. the data CRC has detected an error in the data field.
- Missing packet or out-of sequence packet detected using sequence numbers i.e. the sequence number of the packet received from a specific source logical address is not one more than the previous packet received from that address. Note that there is a separate sequence count for each channel.
- Duplicated packet received i.e. two packets with the same sequence number are received. Note that since the sequence number is an eight-bit number sequence number and will roll over every 256 packets. Duplicate packet numbers must therefore occur within a certain number of packets, specifically within two times the maximum number of outstanding packets. Sequence number incrementation and checking is modulo 256.
- SpaceWire Error End of Packet Error (EEP) i.e. somewhere on the path from source to destination a SpaceWire link error (parity, disconnect, credit or escape error) has occurred resulting in the packet being terminated prematurely by an EEP.

For the Basic, Best Effort and Resource-Reserved QoS there is no retry in the event of an error. In this case when an error occurs it is simply logged and optionally reported at the receiving node.

For the Assured and Guaranteed QoS errors are logged at the receiver even when they are recovered by the retry mechanism. Persistent errors, that exceed the set number of retries, are logged separately and flagged to the user application.

3.9.1.5 Redundancy Function

The Redundancy Model adopted by SpaceWire-RT is that of alternative paths from a source node to a destination node across a SpaceWire network. SpaceWire-RT supports autonomous switching between alternative paths. These alternative paths may be used in one of three ways:

- Sending data over both paths at the same time, which is referred to as simultaneous retry.
- Sending over the prime path and then if there is a failure using the redundant path.
- Sending over either path and if there is a failure of one path all traffic goes over the remaining path.

The path through the SpaceWire network may be defined using SpaceWire path addressing, logical addressing, regional addressing, or any appropriate combination of these addressing methods.

These three approaches may all be implemented autonomously, with SpaceWire-RT providing automatic redundancy switching in the event of a failure. Alternatively redundancy switching may be centrally managed by either specifying a path that can be used for sending from a source to a destination and changing this path specification in the event of a failure being detected, or by

reconfiguring the routing tables in the SpaceWire network to reroute the traffic avoiding the faulty links.

The parameters that control the redundancy switching are:

- Number of attempts on prime path (Np)
- Number of attempts on redundant path (Nr)
- Number of attempts on other alternative paths when appropriate (Na)
- Autonomous reconfiguration enabled/disabled
- Simultaneous retry on/off

When simultaneous retry is off SpaceWire-RT will first send a DP on the prime path. If this fails it will then retry Np-1 times on the prime path. If there is still no success then it will try up to Nr attempts to send the DP on the redundant path. When appropriate, further alternative paths may be tried. Once all retries have been attempted then the persistent error is reported to the local host system.

The following examples serve to illustrate the use of the redundancy switching parameters.

Example: try once on prime path and report if this fails.

- Number of attempts on prime path = 1
- Number of attempts on redundant path = 0
- Autonomous reconfiguration = disabled
- Simultaneous retry = off

Example: try three times on prime path and report if this fails.

- Number of attempts on prime path = 3
- Number of attempts on redundant path = 0
- Autonomous reconfiguration = disabled
- Simultaneous retry = off

Example: try twice on prime path, twice on the redundant path and report if this fails.

- Number of attempts on prime path = 2
- Number of attempts on redundant path = 2
- Autonomous reconfiguration = enabled
- Simultaneous retry = off

Example: try twice simultaneously on prime and redundant paths.

- Number of attempts on prime path = 2
- Number of attempts on redundant path = 2
- Autonomous reconfiguration = disabled
- Simultaneous retry = on

3.9.1.6 Priority

Priority is provided for both asynchronous and scheduled systems, thus it is used in all five qualities of service. Note that SpaceWire-RT does not rely on the router priority feature implemented in the SpW-10X router.

A separate source channel buffer and destination channel buffer are used for each level of priority to be supported for each destination. A DP will be sent from a source channel buffer with a lower channel number before sending a DP from a source channel buffer with a high channel number. The source channel buffers are allocated to destination and priority levels bearing this is mind. For example if there are four destinations (logical addresses 49, 62, 75 and 112) that a source needs to send information to and it wants to have three priority levels (low, medium, high) for communication to each of these destinations then the example channel allocation scheme shown in Table 3-3 can be used.

Table 3-3 Example Channel Numbering for Priority Arbitration		
Destination	**Channel Number**	**Priority**
49	1	High
49	2	Medium
49	3	Low
62	1	High
62	2	Medium
62	3	Low
75	1	High
75	2	Medium
75	3	Low
112	1	High
112	2	Medium
112	3	Low

Channel 1 is used for high priority traffic to the destination with logical address 49.

The source user application puts information into an appropriate source channel buffer depending on where it wants to send the data (destination) and the priority of the data. For example to send data with low priority from the source to the destination with logical address 75, the data is written into source channel buffer 11. SpaceWire-RT will then transfer this information across the SpaceWire network when there is room in the destination channel buffer and after other higher priority traffic with space in its destination buffers has been transferred.

Note that the retry/no-retry selection is also defined on a channel by channel basis.

Note a priority scheme where the priority for each channel is explicitly declared rather than being the same as the channel number will also be considered. This is easier to manage but more difficult to implement in hardware.

3.9.2 Asynchronous Network Functions

In this section the functions specific to asynchronous networks are described.

3.9.2.1 Retry in an Asynchronous Network

In an asynchronous network the go-back N approach is used. When the source sends a DP, it starts a timer. When the DP arrives at the destination, an acknowledgement is returned to the source. If the source does not receive the acknowledgement before the timer times-out, the DP is assumed not to have arrived at the destination and the source resends the DP. If multiple copies of the same DP arrive at the destination the duplicates are discarded.

The retry mechanism is illustrated in Figure 3-15. Note that for simplicity the example shown is go-back 1 which is a specific case of go-back N.

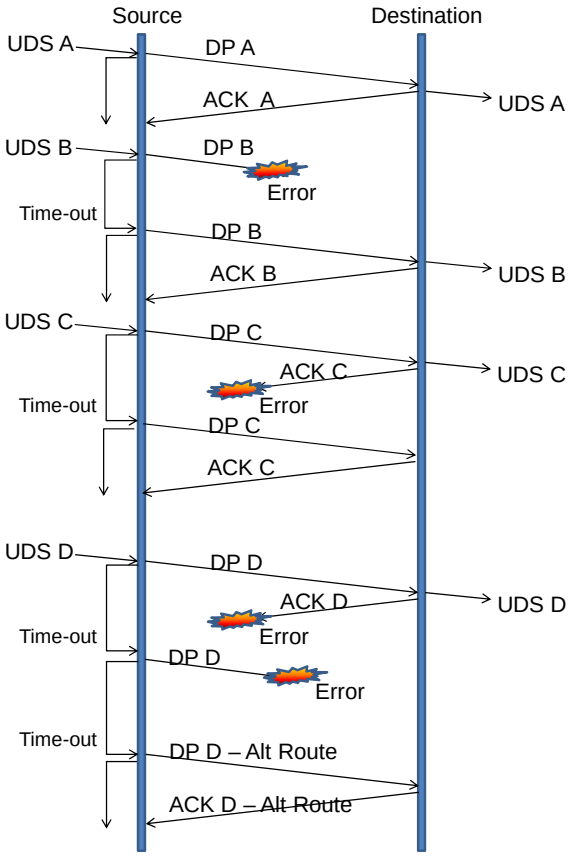

Figure 3-15 Retry Mechanism

UDS A is written into the source channel buffer to be sent to the destination. It is read out of this buffer and packaged into DP A which is sent across the SpaceWire network to the destination. A time-out timer is started when the DP is sent waiting for an ACK. When DP A arrives at the destination node the UDS is extracted from the DP and written into the destination channel buffer. An acknowledgement (ACK A) is sent back to the source to indicate that the DP arrived successfully.

When ACK A arrives back at the source the time-out timer is cancelled and the space in the source channel buffer containing UDS A is freed.

UDS B is written into the source channel buffer. It is subsequently read out of this buffer, packaged in a DP (DP B) and sent to the destination. When it is sent a time-out timer is started waiting for an ACK. On its way to the destination DP B is lost or corrupted. Since it does not arrive at the destination no ACK is returned to the source and the time-out timer expires. When this occurs DP B is resent to the destination and the time-out timer is restarted. This time DP B reaches the destination intact, is written into the destination channel buffer and an acknowledgement (ACK B) returned to the source. When the ACK arrives back at the source the time-out timer is cancelled and the space in the source channel buffer containing UDS B is freed.

UDS C is written into the source channel buffer, then read out, packaged in a DP (DP C) and sent to the destination. When it is sent a time-out timer is started waiting for an ACK. DP C arrives at the destination successfully, is written into the destination channel buffer and an acknowledgement (ACK C) returned to the source. On its way back to the source this ACK is lost or corrupted. Since ACK C does not arrive back at the source the time-out timer expires and DP C is resent to the destination. The time-out timer is restarted when DP is resent. DP C arrives at the destination once more. Since it is a duplicate of a DP that has already been received and placed in the destination channel buffer, it is discarded and another ACK (ACK C) sent back to the source. This ACK arrives at the source intact, the time-out timer is cancelled and the space in the source channel buffer containing UDS C is freed.

UDS D is written into the source channel buffer, then read out, packaged in a DP (DP D) and sent to the destination. When it is sent a time-out timer is started waiting for an ACK. DP D arrives at the destination successfully, is written into the destination channel buffer and an acknowledgement (ACK D) returned to the source. On its way back to the source this ACK is lost because there is permanent failure with a SpaceWire link. Since ACK D does not arrive back at the source the time-out timer expires and DP D is resent to the destination. The time-out timer is restarted when DP is resent. Since there is a permanent failure on a SpaceWire link DP D does not arrive at the destination and no acknowledgement is sent. The time-out timer in the source expires a second time and DP D is resent using an alternative path through the SpaceWire network. When DP D is resent the time-out timer is restarted once more. Along the alternative route there is no fault so DP D arrives at the destination. Since it is a duplicate of a DP that has already been received it is discarded and another acknowledgement (ACK D) returned to the source. Travelling over the alternative route the ACK arrives safely at the source, the time-out timer is cancelled and the space in the source channel buffer containing UDS D is freed.

As well as providing sequential retry where a duplicate DP is sent if the ACK for a DP fails to arrive before the time-out timer expires, SpaceWire-RT provides simultaneous retry. This is illustrated in Figure 3-16.

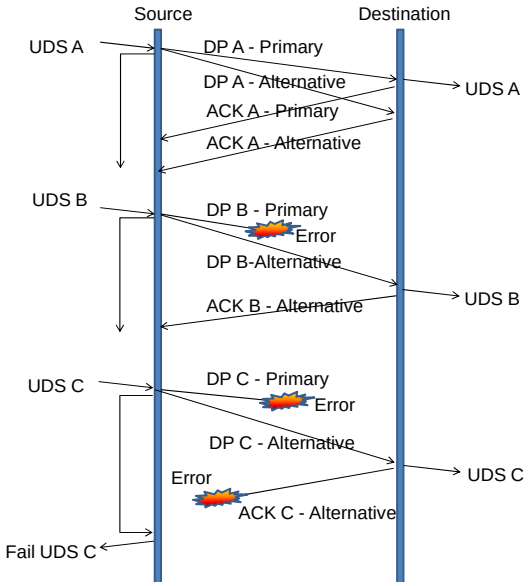

Figure 3-16 Simultaneous Retry

The source user application puts UDS A into the source channel buffer to send to the destination. Assuming that there is space in the destination channel buffer for the UDS it is packaged into a DP and sent to the destination twice using different paths (DP A – primary and DP A – Alternative). When these DPs are sent a single time-out timer is started waiting for an acknowledgement. In Figure 3-16 DP A – Primary arrives at the destination, UDS A is extracted and put in the destination channel buffer and an acknowledgement (ACK A – Primary) returned to the source. Shortly after DP A – Primary starts to arrive, DP A – Alternative arrives. This DP is a duplicate of DP A – Primary so is discarded and an acknowledgement (ACK A – Alterative) is sent back to the source. Assuming ACK A – Primary arrives at the source first it cancels the acknowledgement time-out timer and frees the space used by UDS A in the source channel buffer. When ACK A – Alternative arrives it is ignored since the time-out timer has been cancelled already.

UDS B is placed in the source channel buffer next. It is subsequently sent in two DPs to the destination (DP B – Primary and DP B – Alternative) and a time-out timer is started waiting for an acknowledgement. DP B – Primary is lost or corrupted on the way, but there is no problem on the alternative path so DP B – Alternative arrives at the destination and UDS B is extracted and placed in the destination channel buffer. ACK B – Alternative is returned to the source arriving there without misfortune and causing the time-out timer to be cancelled and the space for UDS B in the source channel buffer to be freed.

UDS C is the next UDS to be placed in the source channel buffer. Again two DPs are sent and the time-out timer started. DP C – Primary is lost or corrupted on its way to the destination. DP C – Alternative arrives safely, UDS C is extracted and put in the destination channel buffer and ACK C –

Alternative sent back to the source. On its way to the source this DP is lost or corrupted. Since no acknowledgement is received at the source the timer-out timer expires and a failure indication is given to the source user application indicating the failure to reliably deliver UDS C to the destination (even though it was actually received in this case). Both primary and alternative paths failed in this example.

It is also possible to do the simultaneous retry over a single path. In this case a short interval should be left between the two packets in case of an error covering the end of one packet and the start of the next. Note: there is still a problem if the EOP of the first packet goes missing. The sending of an empty packet between simultaneous retry packets could be considered to avoid this failure case.

3.9.2.2 Flow Control in an Asynchronous Network

End to end flow control is necessary to make sure that there is room in a buffer at the destination node before a DP is sent. This prevents the SpaceWire packet containing the DP being blocked by a destination that is not ready to receive it and thus being strung out across the SpaceWire network blocking other network traffic. Flow control is achieved by the destination channel buffer sending a buffer flow control token (BFCT) when it has enough room for another maximum length DP. To avoid a problem if a BFCT is lost BFCTs are acknowledged. If a BFCT acknowledgement (BACK) is not received within a certain time-out interval then the BFCT is resent. Each BFCT contains a sequence number which increments each time another BFCT is sent for a specific channel. The sequence number of the BFCT is also used in the BACK so that each BACK is related to a specific BFCT.

The available buffer space is handled in units of the amount of space needed to hold a maximum sized DP because this is the unit of data that is sent. All DPs are the maximum DP size except the last one for a particular SDU.

Operation of the flow control mechanism is illustrated in Figure 3-17 and Figure 3-18.

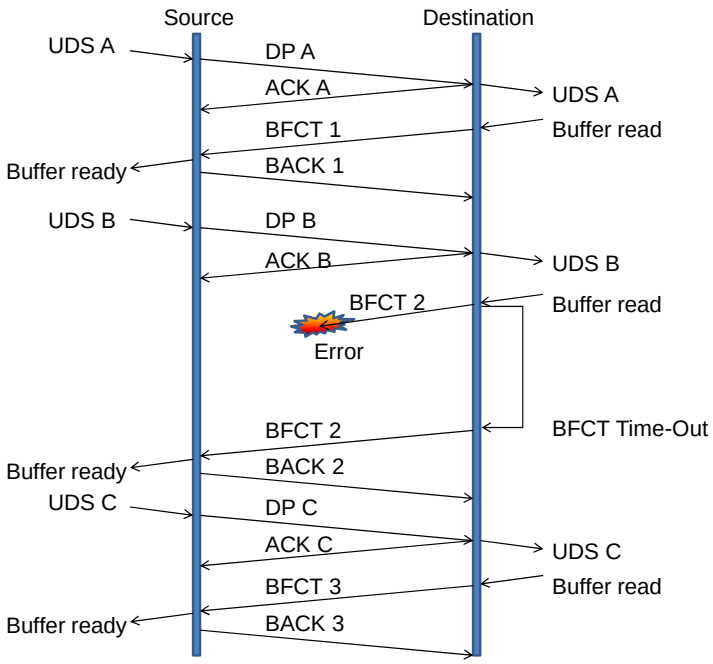

Figure 3-17 Flow Control Mechanism: BFCT Time-Out

In Figure 3-17 a source node is sending data to a destination node across a channel. The source user application passes UDS A to SpaceWire-RT to send. It is packaged into DP A and sent across the SpaceWire network. UDS A is extracted from the DP and put in the appropriate destination channel buffer. Sometime later UDS A is read by the destination user application freeing space in the destination channel buffer. This causes a BFCT to be sent back to the source. This BFCT has sequence number 1 (BFCT 1). When it arrives at the source it signals to the source user application that there is space for another UDS in the destination channel buffer (buffer ready). The source user application can then submit another UDS for sending across the channel when it has more data to send. An acknowledgement to the BFCT (BACK 1) is returned to the destination.

The source user application passes UDS B to SpaceWire-RT to send. It is packaged into DP B and sent across the SpaceWire network. When it arrives at the destination UDS B is extracted from the DP and put into the available space in the destination channel buffer. This UDS is read from the buffer and a BFCT with the next value sequence number (BFCT 2) is returned to the source. Unfortunately this BFCT is lost or is corrupted on its way across the network. Since no BFCT arrives at the source the channel is blocked with the source being unable to send another DP because it does not know that there is space in the destination channel buffer. To overcome this problem when the BFCT is sent the destination starts a time-out time waiting for the BACK. This timer is cancelled when a BACK arrives on the channel with a sequence number equal to or greater than the sequence number of the BFCT. If no BACK arrives before the time-out timer expires then the BFCT is resent. This ensures that

the BFCTs are delivered and that the source can send data to the destination whenever there is room in the destination channel buffer.

The situation that occurs when a BFCT goes missing is illustrated in Figure 3-17. BFCT 2 has been lost or corrupted so no more DPs are received on that channel because the source does not know that there is space available in the destination channel buffer. The BFCT timer times out and BFCT 2 is resent. This time it arrives safely at the source and the fact that there is more space in the destination channel buffer is reported to the source user application (buffer ready). The source can then submit the next UDS for sending (UDS C).

Figure 3-18 shows what happens when a BFCT is lost but a subsequent BFCT is delivered successfully.

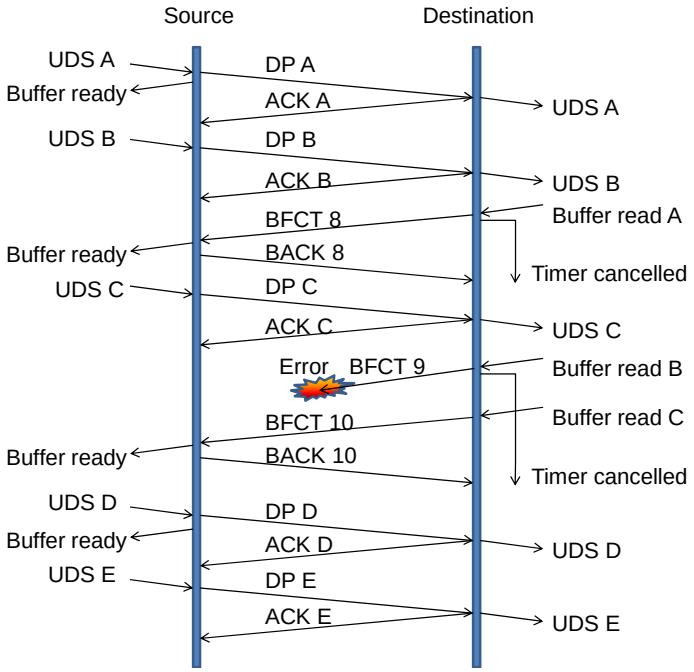

Figure 3-18 Flow Control Mechanism: Cancelling BFCT Time-Out Timers

The source has received two BFCTs already. The source user application passes UDS A to the channel to send. Since the destination channel buffer has room for two more UDSs, the UDS is packaged into DP A and sent across the SpaceWire network. When DP A reaches its destination it is placed in the destination channel buffer and an acknowledgement (ACK A) returned to the source. There is room for one more UDS in the destination channel buffer so the source user application passes UDS B to the channel to send. Packaged into DP B it travels across the SpaceWire network and UDS B is delivered to the destination channel buffer. The destination user application reads UDS A from the destination channel buffer sometime later and a BFCT (BFCT 8) is sent to the source to

indicate there is room for one more DP in the destination channel buffer. A time-out timer is started when the BFCT is sent waiting for the corresponding FCAK. BFCT 8 arrives at the source and BACK 8 is returned to the destination cancelling the corresponding time-out timer when it arrives. UDS C is then transferred across the network and ACK C returned to the source. UDS B is read from the destination channel buffer by the destination user application, freeing space for another UDS in this buffer. BFCT 9 is sent to the source to inform it of the available space however it is corrupted or lost on its way across the SpaceWire network. The time-out timer will detect this eventually, but in the meantime UDS C is read from the destination channel buffer and another BFCT (BFCT 10) sent to the source. BFCT 10 arrives at the source safely and the source returns BACK 10 to the destination. Since the previous BFCT that the source received was BFCT 8 when BFCT 10 arrives, it knows that BFCT 9 must have also been sent and been lost, so it can safely assume that there is now space for two UDSs in the destination channel buffer. When BACK 10 arrives at the destination timers for any outstanding BFCT up to BFCT 10 are cancelled.

The situation that occurs when a BACK goes missing is illustrated in Figure 3-19

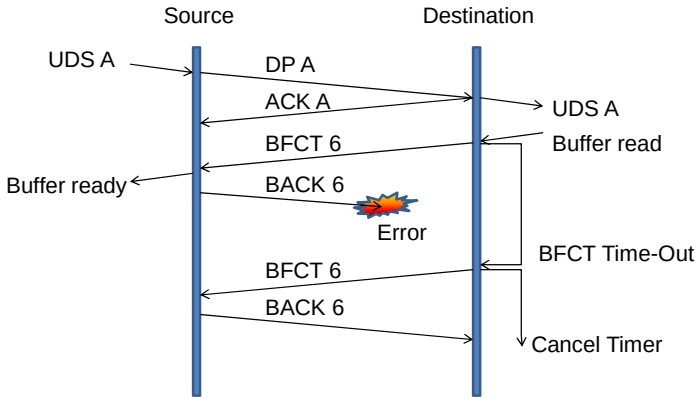

Figure 3-19 Flow Control Mechanism: Missing BACK

UDS A is transferred across the SpaceWire network to the destination channel buffer. An UDS is read out of the destination channel buffer freeing space for another UDS. BFCT 6 is sent to the source to let it know that more space is available and a time-out timer stared waiting for the corresponding BACK. When BFCT 6 arrives at the source BACK 6 is returned to the destination so that it knows that BFCT 6 has been delivered successfully. Unfortunately BACK 6 is lost or corrupted on its way back to the destination. The BFCT time-out timer expires in the destination and BFCT 6 is resent. BFCT 6 arrives once more at the source. Since it is a duplicate it is ignored, but another BACK 6 is sent back to the destination. This time BACK 6 arrives at the destination successfully and the corresponding BFCT timer is cancelled.

If a SpaceWire packet arrives at a destination where there is no room in a buffer for DP it contains, that packet is spilt (discarded) immediately to prevent the SpaceWire network being blocked.

BFCTs use the same redundancy switching mechanism as for DPs.

It is possible that when the BFCT 6 arrives and “buffer ready” is indicted to the source another UDS (UDS 6) is ready for sending straightaway. This situation is illustrated in Figure 3-20.

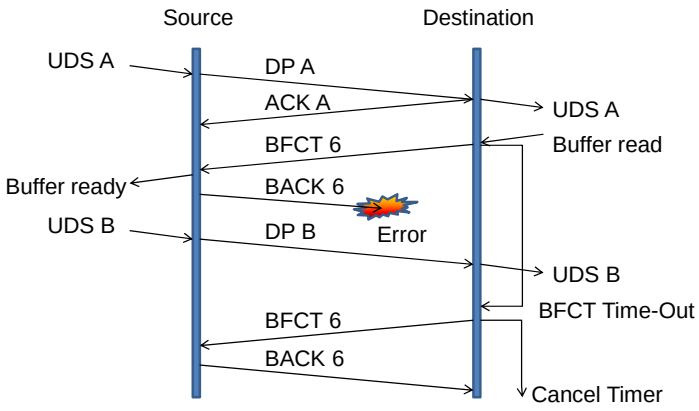

Figure 3-20 Flow Control Mechanism: Missing BACK, UDS Ready

When BFCT 6 arrives at the source BACK 6 is returned to the destination so that it knows that BFCT 6 has been delivered successfully. The fact that the destination buffer is ready to receive more data is indicated to the source. The source buffer has UDS B ready to send to this is transmitted in DP B. BACK 6 is lost or corrupted on its way back to the destination. DP B arrives at the destination, UDS B is extracted and is made available in the destination buffer.

The BFCT time-out timer expires in the destination since BACK 6 was not received and BFCT 6 is resent. BFCT 6 arrives once more at the source. Since it is a duplicate it is ignored, but another BACK 6 is sent back to the destination. This time BACK 6 arrives at the destination successfully and the corresponding BFCT timer is cancelled.

3.9.2.3 Encapsulation

The encapsulation function encapsulates DPs, ACKs, BFCTs and BACKs into SpaceWire packets.

3.9.2.3.1 DP Encapsulation

The DP encapsulation function encapsulates the UDS and associated parameters into a SpaceWire packet. The DP extraction function extracts the UDS from a SpaceWire packet.

The DP encapsulation is illustrated in Figure 3-21.

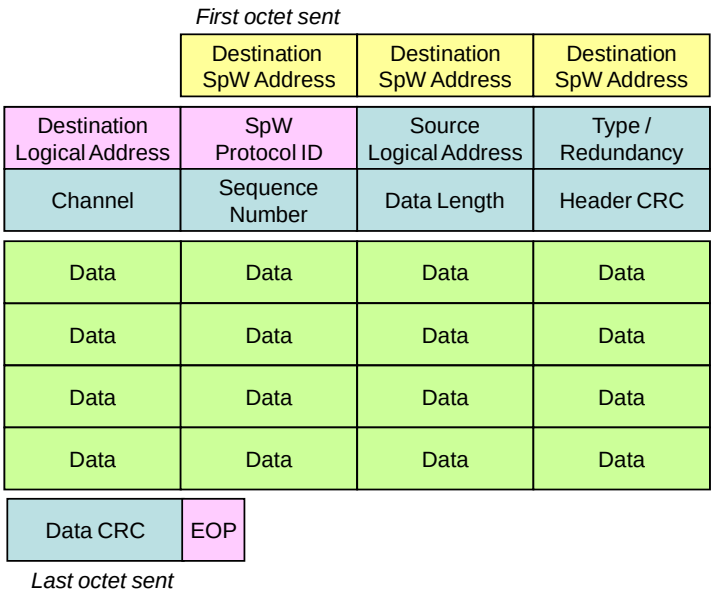

Figure 3-21 DP Encapsulation

The fields of the DP are described below:

The **destination SpaceWire address** is a variable length field that contains the SpaceWire path and/or regional logical address that routes the packet across the SpaceWire network to the required destination.

The **destination logical address** is a one byte field containing the logical address of the destination node.

The **protocol identifier** is a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

The **source logical address** is a one byte field that identifies the SpaceWire node sending the packet by its logical address.

The **type** field is an eight bit field comprising the following sub-fields:

- The **packet type** is a 3-bit field containing the type of packet (DP, ACK, BFCT, BACK, Scheduled ACK (SACK) or Scheduled BFCT (SBFCT)).
- The **redundancy** field is 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network. An ACK or BACK should use the same redundancy path as the corresponding DP or BFCT.
- The **start/end** field contains 2-bits which indicate the position of the encapsulated UDS in the SDU as follows:

- Start/end bits = 10: indicates the DP contains a UDS which is the start of a SDU
- Start/end bits = 01: indicates the DP contains a UDS which is the end of a SDU.
- Start/end bits = 00: indicates the DP contains a UDS which is the middle (neither start nor end) of a SDU.
- Start/end bits = 11: indicates the DP contains a UDS which holds an entire SDU.

- The remaining 1-bit in the type field shall be reserved and set to zero.

The **channel** together with the destination and source logical address identifies the channel being used for communication and the associated source and destination channel buffers.

The **sequence number** is a one byte field containing an 8-bit sequence count used to detect missing DPs and BFCTs. There is a separate sequence count for each channel and for DPs and BFCTs within a channel.

The **data length** is a one byte field that specifies the number of data bytes in the data field. The value 0x00 means a data length of 256 bytes. All other values are directly the data length in bytes.

The **header CRC** is a one byte field containing an 8-bit CRC covering the header of the packet. This uses the same CRC format as the SpaceWire RMAP standard (ECSS-E50-11). The header CRC covers the header from the Destination Logical Address to the byte immediately prior to the header CRC. It does not include the Destination SpaceWire Address as this is deleted during passage through the SpaceWire network. The header CRC is used to check that the header is correct before the packet is processed. If there is an error in the header the entire packet is discarded.

The **data field** is a variable length field containing up to 256 data bytes.

The **data CRC** field contains a 8-bit CRC covering the data field only. This is used to confirm that the data has been delivered without error.

The **end of packet marker** is a SpaceWire control code that indicates the end of the SpaceWire packet and the start of the next one.

3.9.2.3.2 Control Code Encapsulation

The encapsulation of control codes (ACKs, BFCTs, and BACKs) is illustrated in Figure 3-22

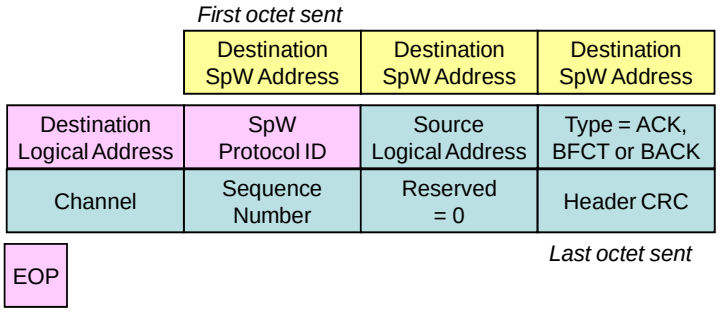

Figure 3-22 Control Code Encapsulation

The control code encapsulation is similar to the DP encapsulation. There is no data field and no data CRC. The date length field in the DP becomes a reserved field set to zero in the control code.

3.9.3 Scheduled Network Functions

In this section the functions specific to scheduled networks are described. The scheduled flow control is necessarily different to asynchronous due to the requirement to control all of the traffic on a scheduled system (including ACKs and BFCTs) and also the need to inform the sender quickly if there is a problem.

Note that the number of channels between a particular source and destination pair is limited to 40 for a scheduled system. This is because of the way that buffer flow control is implemented (see section 3.9.3.3).

3.9.3.1 Scheduling

Scheduling is used specifically to support resource-reserved and guaranteed qualities of service, although basic, best effort and assured QoS can also operate over a scheduled network on an opportunistic basis.

In a scheduled network the network bandwidth is separated using time-division multiplexing into a series of repeating time-slots. A schedule table is used in each source to specify which source channel buffer(s) are allowed to send information during each time-slot. The schedule tables in every source are devised to avoid conflicts on the network. Only one source is allowed to send information at a time, or multiple sources can send information at the same time provided that they do not use any common network resource i.e. send information over the same SpaceWire link.

An example schedule table is shown for a source node with logical address 231 in Table 3-4.

Table 3-4 Example Schedule Table Source node 231	
Time-slot	**Channel**
0	68/32
1	-
2	49/12
3	82/1
...	
62	49/5
63	-

During time-slot 0 channel 32 from node 231 to 68 (channel 231/68/32) is allowed to send a DP. During time-slot 1 no channel is allowed to send a DP (presumably another source is scheduled to send a DP in this time-slot). During time-slot 2 channel 12 from node 231 to 49 (231/49/12) can send a DP.

Scheduling can be combined with priority so that several channels all to the same destination and using the same network resources for communication but with different priority levels are mapped on to time-slot. When the time-slot arrives, the highest priority channel (the one with the lowest channel number) which has data to send and room in its destination buffer, is sent first. An example schedule table with priority is illustrated in Table 3-5.

Table 3-5 Example Schedule Table with Priority	
Time-slot	**Channel**
0	68/4, 68/8, 68/12
1	-
2	49/12
3	82/1, 82/2, 82/3, 82/4
...	
62	49/5
63	-

In time-slot 0 three levels of priority are supported between nodes 231 and 68 using channels 4, 8 and 12. Time-slot 2 between nodes 231 and 49 has only one level of priority. Time-slot 3 between nodes 231 and 82 has four levels of priority provided by channels 1, 2, 3, 4. If a packet of high priority is to be sent to the destination supported by these channels then it is written in source channel buffer 82/1 and will be sent before any DPs in channels 82/2, 82/3 or 82/4, provided that there is room in the destination channel buffer 231/1 in node 82, i.e. as long as there is destination channel buffer credit."

3.9.3.2 Retry in an Scheduled Network

3.9.3.2.1 Nominal Operation of ACK

The nominal operation of ACK in a scheduled network is illustrated in Figure 3-23.

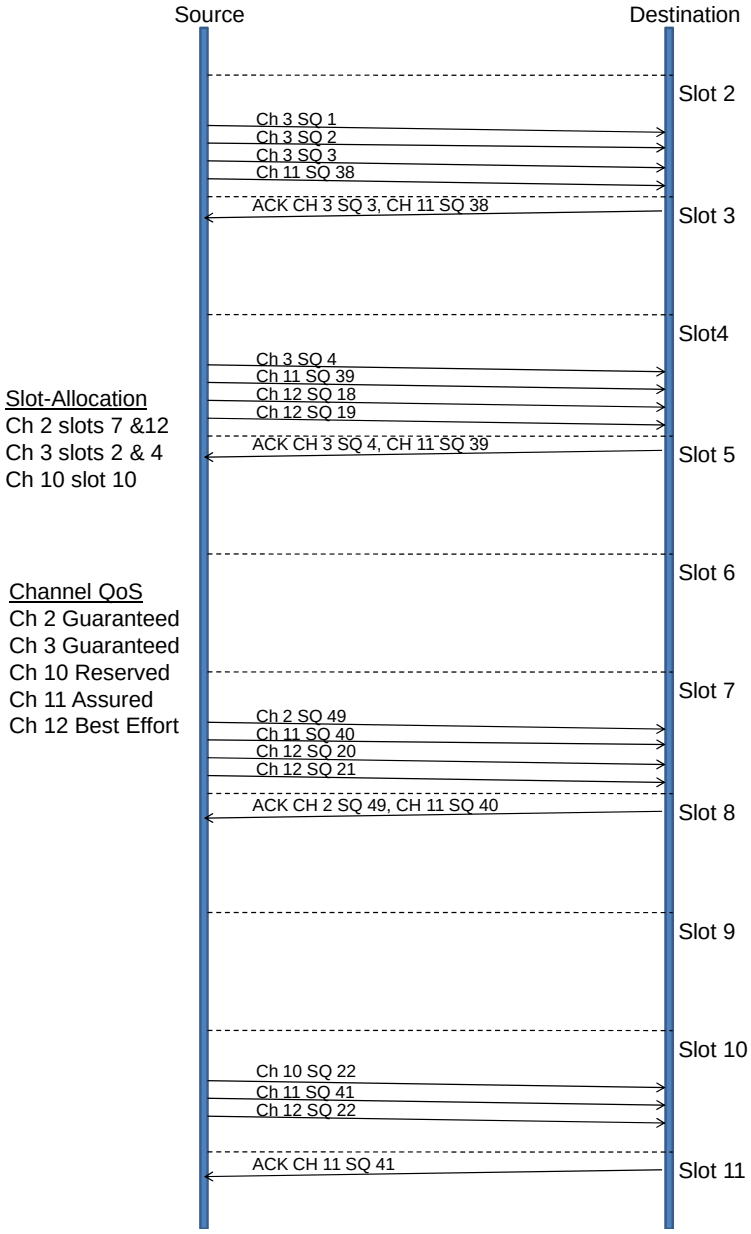

Figure 3-23 Nominal Operation of ACK in Scheduled Network

In this example of Figure 3-24 there are several channels that communicate between a specific source and destination: two guaranteed channels (2 and 3), one resource-reserved channel (10), one assured channel (11) and one best effort channel (12). Channel 2 is scheduled for time-slot 7, channel 3 for time-slots 2 and 4, and channel 10 for time-slot 10. Channels 11 and 12 have ad hoc access to any unused bandwidth in any time-slots allocated for communication between the specific source and destination pair being considered i.e. they can send DPs in time-slots 2, 4, 7 or 10

provided that there is nothing to be sent by any of the guaranteed or resource reserved channels. For brevity the example assumes that the maximum number of DPs that can be sent in a time-slot is four (this is an example only).

In time-slot 2, after an appropriate wait for any acknowledgement traffic, channel 3 sends three DPs with sequence numbers 1, 2 and 3. These are all the DPs available to be sent over channel 3 so the space capacity of this time-slot is used by channel 11 which has a DP ready to send with sequence number 38. Note that only if there is enough time in a time-slot for another complete DP of maximum size to be sent can one be sent.

All of the DPs sent arrive at the destination with no errors. When time slot 3 starts an acknowledgement for channel 3 is sent which indicates that all DPs up to and including the one with sequence number 3 have been successfully received. Similarly all DPs from channel 11up to and including the one with sequence number 38 is acknowledged.

The rest of time-slot 3 is used by some other source/destination pair to transfer information.

Time-slot 4 has been reserved to send any retries for channel 3, but no retires are necessary as all the DPs reached the destination without error. This means that there is some spare capacity available and since another DP is now ready to send over channel 3, the next DP with sequence number 4 is sent. Now there is nothing else to send over channel 3, so one of the ad hoc channels can send some information. Channel 11 has another DP ready to send so it is sent (sequence number 39). Channel 12 has several DPs ready to send so the first two of these are sent (sequence numbers 18 and 19).

All the DPs sent in time-slot 4 arrive without error so at the beginning of time-slot 5 an acknowledgement for channel 3 sequence number 4 and channel 11 sequence number 39 is sent. There is no acknowledgement for channel 12 as this is a best effort channel.

The next time-slot allocated to the specific source/destination pair being considered is time-slot 7. This time-slot is allocated to channel 2 which has a DP ready to send (sequence number 49). There are no more DPs waiting to be sent over channel 2 so the remaining time-slot capacity is used by channels 11 and 12 with channel 11 sending a DP with sequence number 40 and channel 12 sending two DPs with sequence numbers 20 and 21.

At the start of time-slot 8 an acknowledgement is sent for channel 2 sequence number 49 ad channel 11 sequence number 40. No acknowledgement is needed for channel 12 since this is a best effort channel.

The next time-slot that is used by the specific source/destination pair is time-slot 10. This is allocated to channel 10 which is a resource-reserved channel. Channel 10 has a DP ready to send so it is sent in time-slot 10 (sequence number 22). Channel 10 has no more DPs to send so the remaining time-

slot capacity is used by channel 11 to send its next DP with sequence number 41 and then by channel 12 to send its next DP which has sequence number 22.

At the start of slot 11 the DP for channel 11 transferred in time-slot 10 is acknowledged. Since channel 10 is resource-reserved no acknowledgement is needed and again since channel 12 is a best effort channel no acknowledgement is needed there either.

3.9.3.2.2 Lost Data PDUs

The handling of retries in a scheduled system is illustrated in Figure 3-24. Retries are scheduled for a subsequent time-slot. Thus the schedule table has to take account of the timeliness requirements of a possible retry. If no time-slot is allocated for retries for a particular channel then the retry will take place in the next time-slot allocated to that channel. This may be the same numbered time-slot in the next epoch.

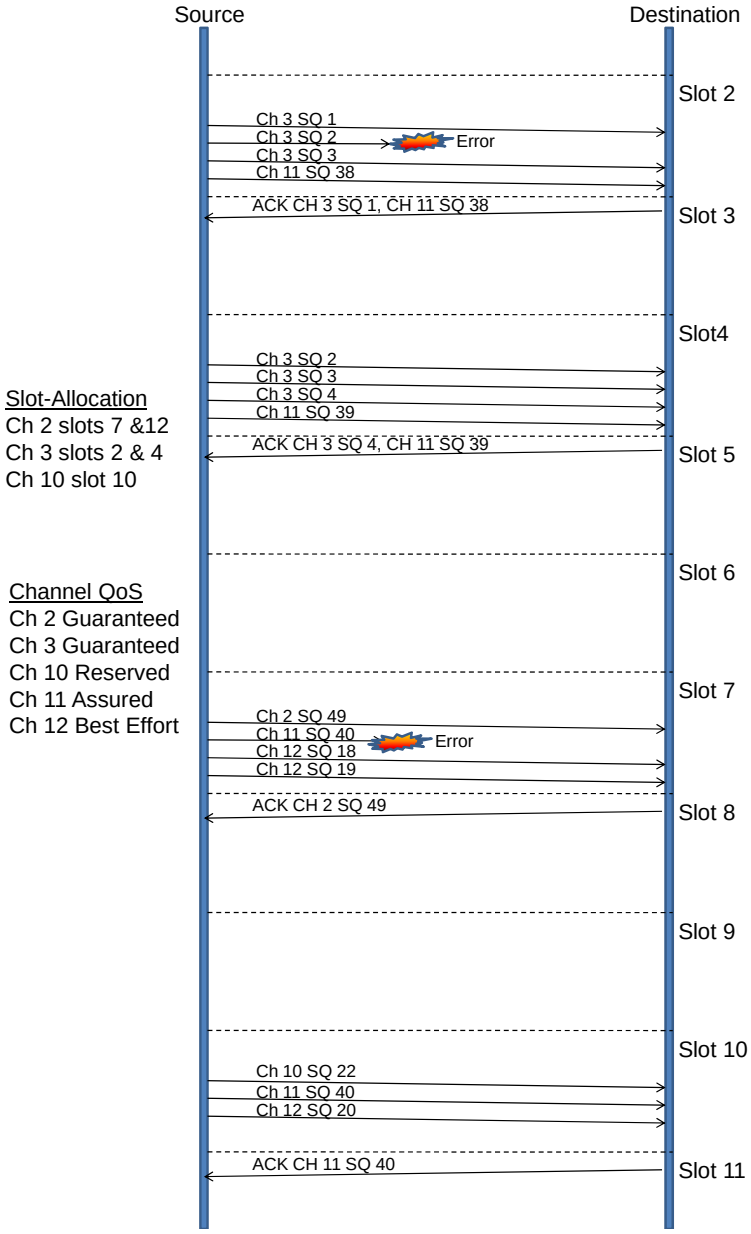

Figure 3-24 ACK in Scheduled Network

Figure 3-24 uses the same scenario as Figure 3-23 except that some errors are introduced.

In time-slot 2, after an appropriate wait for any acknowledgement traffic, channel 3 sends three DPs with sequence numbers 1, 2 and 3. These are all the DPs available to be sent over channel 3 so the space capacity of this time-slot is used by channel 11 which has a DP ready to send with sequence number 38.

One of the DPs sent (channel 3 sequence number 2) is corrupted or lost as it traverses the network. When time slot 3 starts an acknowledgement for channel 3 is sent which indicates that the DP with sequence number 1 has been successfully received along with the DP from channel 11. Since SpaceWire-RT uses a go-back N retry mechanism the DPs with sequence number 2 and 3 have to both be resent even though sequence number 3 was received without error.

The rest of time-slot 3 is used by some other source/destination pair to transfer information.

Time-slot 4 has been reserved to send any retries for channel 3, so the two missing DPs for channel 3 (sequence numbers 2 and 3) are sent. After this there is still capacity available and since another DP is now ready to send over channel 3 the next DP with sequence number 4 is sent. Now there is nothing else to send over channel 3, so one of the ad hoc channels can send some information. Channel 11 has another DP ready to send so it is sent (sequence number 39).

All the DPs sent in time-slot 4 arrive without error so at the beginning of time-slot 5 an acknowledgement for channel 3 sequence number 4 and channel 11 sequence number 39 is sent.

The next time-slot allocated to the specific source/destination pair being considered is time-slot 7. This time-slot is allocated to channel 2 which has a DP ready to send (sequence number 49). There is no more DPs waiting to be sent over channel 2 so the remaining time-slot capacity is used by channels 11 and 12 with channel 11 sending a DP with sequence number 40 and channel 12 sending two DPs with sequence numbers 18 and 19. The DP sent over channel 11 is corrupted.

At the start of time-slot 8 an acknowledgement is sent for channel 2 sequence number 49. No acknowledgement is needed for channel 12 since this is a best effort channel. Since channel 11 sequence number 40 is not acknowledged and since this channel has assured QoS it will be resent at the earliest opportunity.

The next time-slot that is used by the specific source/destination pair is time-slot 10. This is allocated to channel 10 which is a resource-reserved channel. Channel 10 has a DP ready to send so it is sent in time-slot 10 (sequence number 22). Channel 10 has no more DPs to send so the remaining time-slot capacity is used by channel 11 to resend the DP with sequence number 40 and then by channel 12 to send its next DP which has sequence number 20.

At the start of slot 11 the DPs for channels 10 and 11 transferred in time-slot 10 are acknowledged. Since channel 12 is a best effort channel no acknowledgement is needed.

3.9.3.2.3 Lost Acknowledgements

It is possible that an acknowledgement can be lost or corrupted. This situation is illustrated in Figure 3-25.

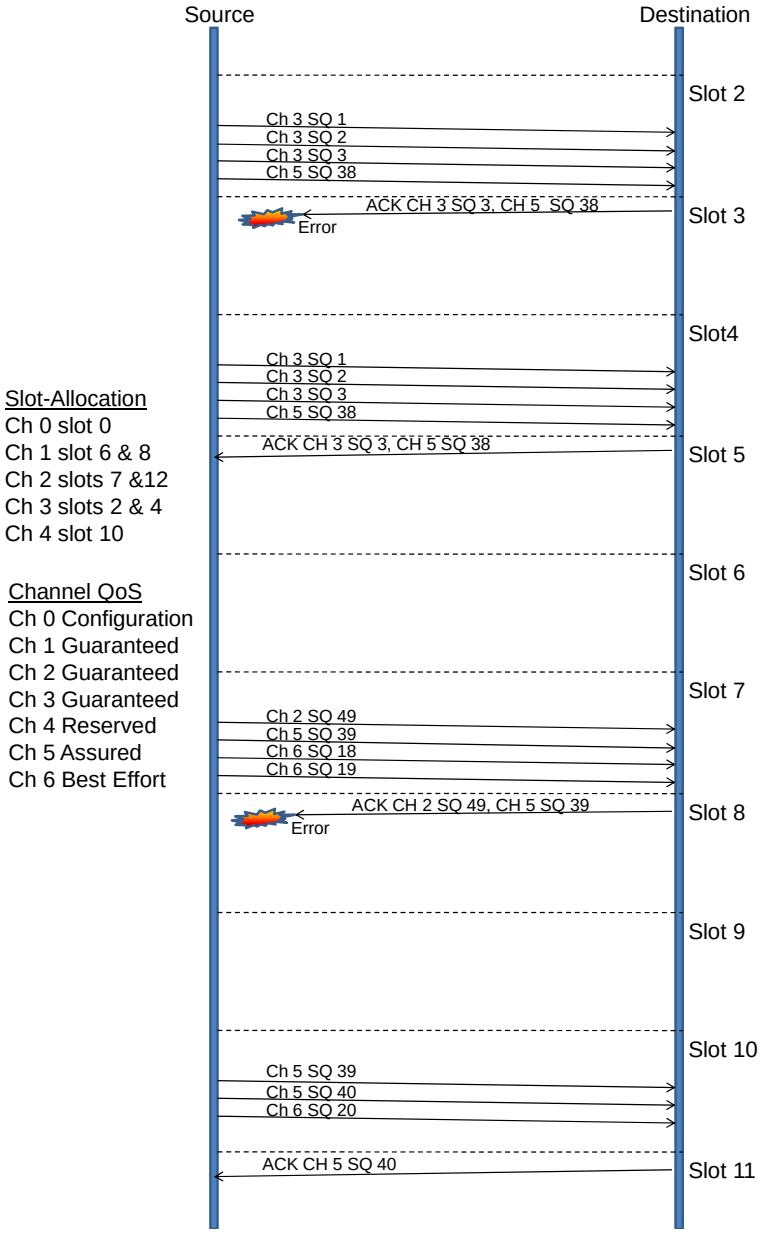

Figure 3-25 ACK Errors in Scheduled Network

At the start of time-slot 3 the DPs sent in the previous time-slot are acknowledged. The acknowledgement is lost or corrupted. Since no acknowledgement is received and since channel 3 has guaranteed QoS and channel 5 is assured, both channel 3 and channel 5 will retry. The guaranteed channel (3) will retry in the next slot scheduled for channel 3 (the retry time-slot) which is time-slot 4. Channel 5 which is assured will retry when the opportunity presents itself i.e. as soon as a time-slot communicating between the particular source and destination pair has some unused

capacity. In the case shown in Figure 3-25, this also happens in time-slot 4, so that in time-slot 4 channel 3 resends three DPs (sequence numbers 1, 2 and 3) and channel 5 resends the DP with sequence number 38.

Later on the acknowledgement for DPs sent in time-slot 7 is also lost or corrupted. In time-slot 10 channel 5 gets the chance to resend the DP with sequence number 39. Channel 2 has to wait until time-slot 12 which is allocated to channel 2 to provide the required retry capability for the guaranteed QoS. The lost acknowledgement did not contain any information about channel 6 since this channel is using best effort QoS and no retries are needed.

3.9.3.3 Flow Control in a Scheduled Network

Flow control in a scheduled system is handled in a completely different manner to that of an asynchronous system. The primary driver for this is that it is difficult to schedule the BFCTs and BACKs as they travel in opposite directions and a BFCT can occur at any time. Since communications between a particular source/destination pair takes place in one or more specific time-slots, flow control can be sent at the end of the time-slot (or the beginning of the next one) for all the channels between the particular source/destination pair. As flow control information is being sent regularly there is no need for BACKs and the related time-out timers. Furthermore 2-bits serve to indicate whether a particular destination channel buffer is able to accept up to three DPs or not. Thus to indicate the flow control status of the maximum possible number of channels between a particular source/destination pair would take 512 bits (64 bytes). Since 2-bits are used a maximum of three DPs can normally be sent in one time-slot.

The flow control information for all the channels between a particular source/destination pair is sent at the beginning of a time-slot. This provides up to date information on the status of the destination channel buffers.

If the flow-control information is lost then transfer of data can use the last received destination channel buffer status information together with knowledge of what DPs have been sent since to determine what buffers definitely have space available. To do this the source counts the DPs sent since the last BFCT received so that it knows when it has run out of credit. The flow control information will be repeated next time the source/destination pair has a time-slot allocated to it, updating the information for the next time-slot.

The flow control mechanism for a scheduled system is illustrated in Figure 3-26.

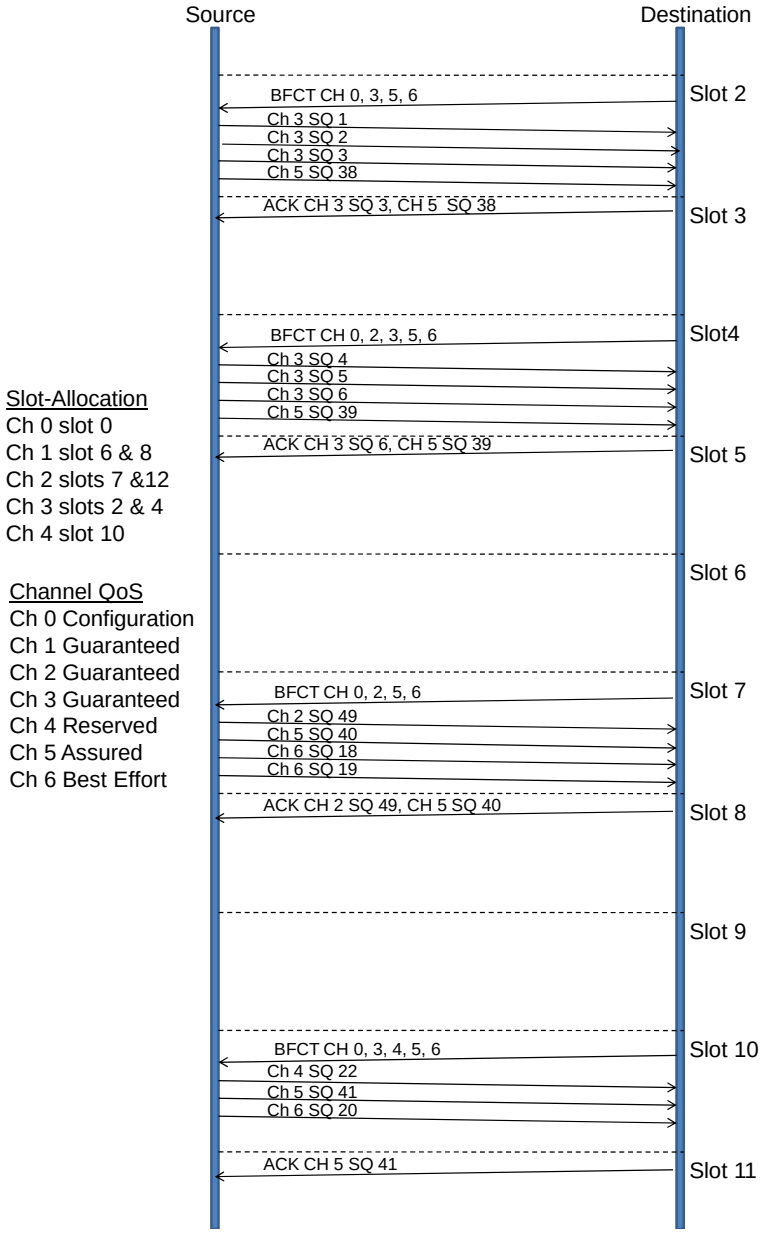

Figure 3-26 Scheduled Flow-Control

In Figure 3-26 it is assumed that all source channel buffers have data to send all the time.

Slots 2, 4, 7 and 10 are allocated to various channels connecting the source/destination pair being considered. The other time-slots are for other source/destination pairs. Note that multiple source/destination pairs can use time-slots in parallel provided that there is no sharing of network resources (i.e. SpaceWire links on the network). It is possible for a source/destination pair to use consecutive time-slots.

Shortly after the start of time-slot 2 the destination sends a BFCT packet to the source to provide it with the latest channel buffer status information. This indicates that there is room for more data in the destination channel buffers for channels 0, 3, 5 and 6. Since slot 3 is allocated to the guaranteed channel 3 this channel will transfer any DPs it has ready to go. It transfers three DPs with sequence numbers 1, 2 and 3, and then has no more DPs to send. The remaining time-slot capacity is used by channel 5 an opportunistic assured channel. The acknowledgements for the channel 3 and channel 5 activity is sent from the destination to the source at the start of time-slot 3.

In time-slot 4 the BFCT packet indicates channels 0, 2, 3, 5 and 6 all have room for data. Slot 3 is allocated to the guaranteed channel 3. It sends three more DPs with sequence numbers 4, 5 and 6, and then has no more data to send. Channel 5 has another DP ready to send so this is sent using the spare capacity of the time-slot. At the start of time-slot 5 the acknowledgements for channels 3 and 5 are sent.

In time-slot 7 the BFCT indicates that channels 0, 2, 5 and 6 may send data. Channel 2 is a guaranteed channel allocated to time-slot 7 so it sends DP with sequence number 49. This is the only packet that channel 2 has to send. The remaining capacity of the time-slot is used by the opportunistic channels 5 and 6. At the start of time-slot 8 the acknowledgements for channels 2 and 5 are sent. There is no acknowledgement for channel 6 as it has opportunistic best effort QoS.

It is possible that the BFCT is lost or corrupted. This situation is illustrated in Figure 3-27.

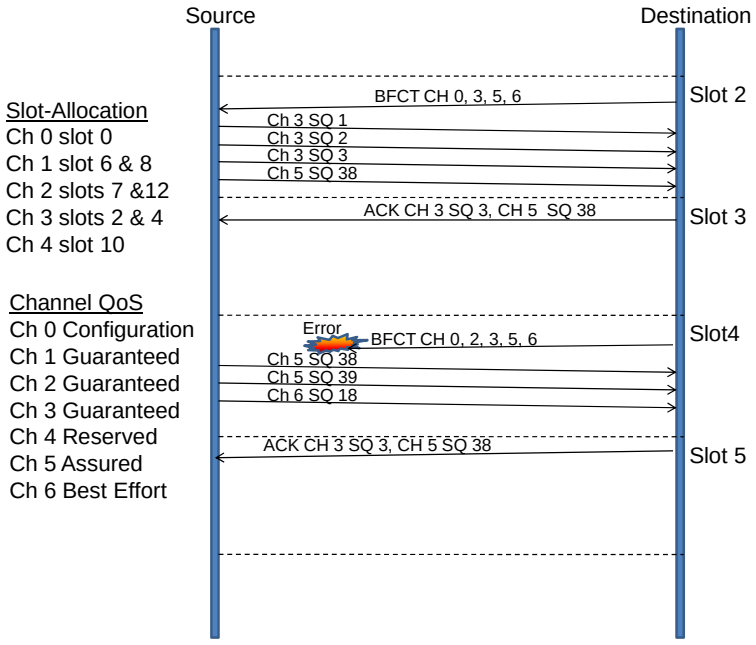

Figure 3-27 Error in Scheduled Flow-Control

In time-slot 2 the BFCT indicates that channels 0, 3, 5 and 6 have available destination channel buffer space, say space for three DPs in each. Channel 3 is thus able to send three DPs in slot 2 and channel 5 one DP using the spare time-slot capacity.

At the start of time-slot 4 the BFCT is lost or corrupted. The source then has to rely on the previous received BFCT and the DPs sent since then along with their acknowledgements. Channel 3 which should send data in time-slot 4 had room for three DPs at the start of time-slot 2, but has sent three DPs since then all of which were acknowledged. Therefore no further DPs can be sent across channel 3. Channel 5 has room for a further two DPs so it sends another two (sequence numbers 38 and 39). Channel 6 uses the remaining time-slot capacity to send a single DP.

It is worth noting that if one or more of the PDs sent over channel 3 in time-slot 2 were not acknowledged then they could still be resent in time-slot 4 even if no BFCT is received at the start of time-slot 4.

3.9.3.4 Time-Slot Timing

The timing of a time-slot in a scheduled network is illustrated in Figure 3-28.

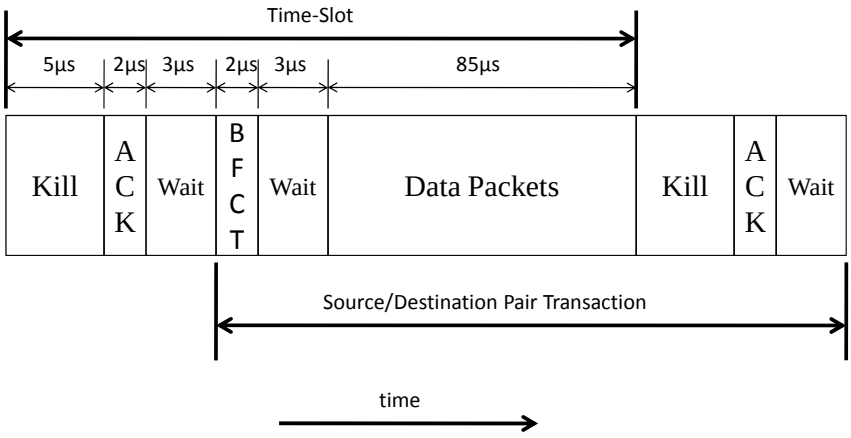

Figure 3-28 Time-Slot Timing

To understand the time-slot timing, consider first the source/destination pair transaction shown in Figure 3-28. This starts with the destination sending the BFCT to the source informing the source about the current status of the destination channel buffers related to the source/destination pair. To allow the BFCTs to propagate through the network unhindered by other packets a gap (wait) follows the BFCT where no transmissions take place. After the wait interval data PDUs are sent from the source to the destination over one or more channels. There is a limit to the number of DPs that can be sent so that they do not extend beyond the end of the Data PDUs interval. At the end of the Data PDUs interval the next time-code arrives. When this happens any packets still being sent are terminated abruptly and a “kill” interval allows time for all traffic on the network to die out.

The killing of packets only happens on the sending side. Packets being received are not killed. Note that if any packets are still being sent when the time-code is distributed, the scheduling has failed and an error is reported. The “killing” of packets still being sent when the time-code arrives is done to prevent fault propagation.

Following the “kill” interval the acknowledgements are sent in the ACK interval for the relevant DPs sent in the previous time-slot. ACK is followed by another wait period to allow the ACKs to propagate across the network. Following this wait interval the next BFCT is sent.

The actual time-slot covers the period from the “kill” interval to the end of the data PDU transfer. This is so that the arrival of the next time-code can terminate the sending of any data PDUs that are still being sent when the time-code arrives. The “kill”, ACK, wait, BFCT and wait intervals are then reasonably deterministic in length, so that no other traffic is flowing when the Data PDU interval starts.

Typical timing when the SpaceWire link speeds are all set to 200 Mbps are shown in Figure 3-28. This timing results in a time-slot of 100 µs and an epoch of 6.4 ms. Six maximum size DPs can be sent in the 85 µs Data PDU interval. Note that the DP length is 256 maximum and is not a user configurable parameter. There is a maximum of 6 DPs in a time-slot.

3.9.3.5 Scheduled Encapsulation

The encapsulation function encapsulates DPs, ACKs, BFCTs and BACKs into SpaceWire packets.

3.9.3.5.1 DP Encapsulation

The DP encapsulation function encapsulates the UDS and associated parameters into a SpaceWire packet. The DP extraction function extracts the UDS from a SpaceWire packet.

The DP encapsulation is illustrated in Figure 3-29 and is the same as that for an asynchronous network.

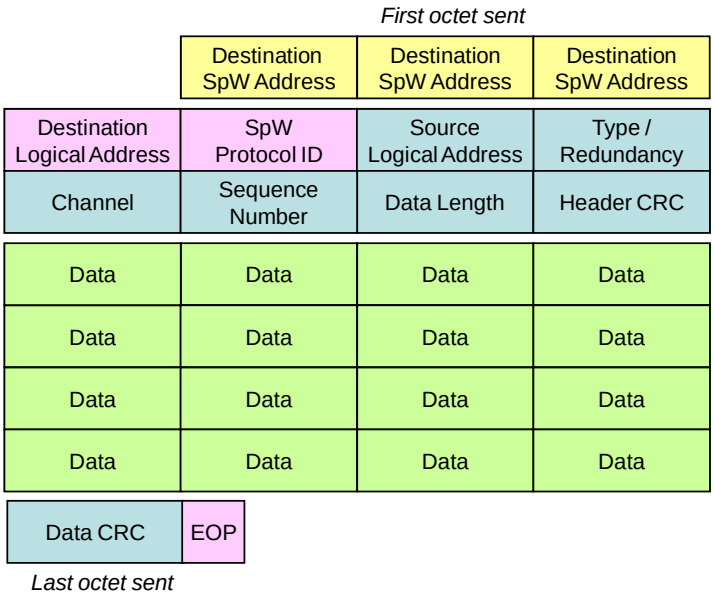

Figure 3-29 DP Encapsulation

The fields of the DP are described below:

The **destination SpaceWire address** is a variable length field that contains the SpaceWire path and/or regional logical address that routes the packet across the SpaceWire network to the required destination.

The **destination logical address** is a one byte field containing the logical address of the destination node.

The **protocol identifier** is a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

The **source logical address** is a one byte field that identifies the SpaceWire node sending the packet by its logical address.

The **type** field is an eight bit field comprising the following sub-fields:

- The **packet type** is a 3-bit field containing the type of packet (DP).
- The **redundancy** field is a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network. An ACK or BFCT should use the same redundancy path as the corresponding DP.
- The **start/end** field contains 2-bits which indicate the position of the encapsulated UDS in the SDU as follows:
    - Start/end bits = 10: indicates the DP contains a UDS which is the start of a SDU

- Start/end bits = 01: indicates the DP contains a UDS which is the end of a SDU.
- Start/end bits = 00: indicates the DP contains a UDS which is the middle (neither start nor end) of a SDU.
- Start/end bits = 11: indicates the DP contains a UDS which holds an entire SDU.

- The remaining 1-bit in the type field shall be reserved and set to zero.

The **channel number** together with the destination and source logical addresses identifies the channel being used for communication and the associated source and destination channel buffers.

The **sequence number** is a one byte field containing an 8-bit sequence count used to detect missing DPs and BFCTs. There is a separate sequence count for each channel.

The **data length** is a one byte field that specifies the number of data bytes in the data field. A data length of 0 means there are 256 bytes in the data field.

The **header CRC** is a one byte field containing an 8-bit CRC covering the header of the packet. This uses the same CRC format as the SpaceWire RMAP standard (ECSS-E50-11). The header CRC covers the header from the Destination Logical Address to the byte immediately prior to the header CRC. It does not include the Destination SpaceWire Address as this is deleted during passage through the SpaceWire network. The header CRC is used to check that the header is correct before the packet is processed. If there is an error in the header the entire packet is discarded.

The **data field** is a variable length field containing up to 256 data bytes.

The **data CRC** field contains a 8-bit CRC covering the data field only. This is used to confirm that the data has been delivered without error.

The **end of packet marker** is a SpaceWire control code that indicates the end of the SpaceWire packet and the start of the next one.

3.9.3.5.2 Control Code Encapsulation

The encapsulation of acknowledgements and BFCTs are illustrated in Figure 3-30 and Figure 3-31 respectively.

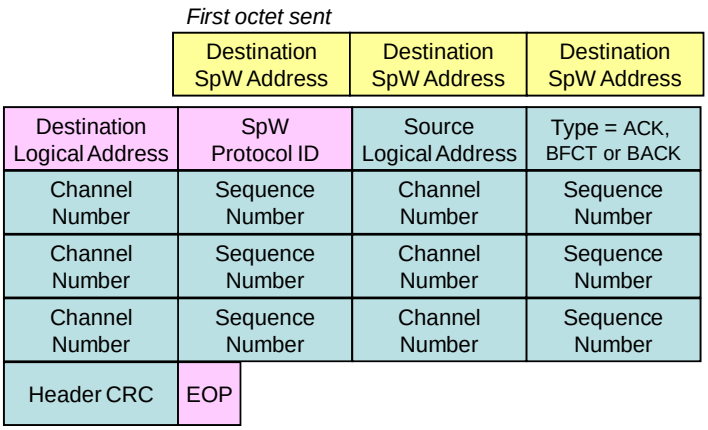

Figure 3-30 Scheduled ACK Encapsulation

The scheduled ACK is similar to other PDUs except that it contains a string of channel numbers and sequence numbers in pairs. Each channel number holds the number of the channel being acknowledged. The following sequence number is the sequence number of the last DP being acknowledged in the corresponding channel. There is one channel/sequence number pair for each channel being acknowledged. Only guaranteed and assured PDs are acknowledged.

The fields of the schedule ACK are described below:

The **destination SpaceWire address** is a variable length field that contains the SpaceWire path and/or regional logical address that routes the packet across the SpaceWire network to the required destination. Since this is an acknowledgement the destination is the channel input node.

The **destination logical address** is a one byte field containing the logical address of the channel input node.

The **protocol identifier** is a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

The **type** field is an eight bit field comprising the following sub-fields:

- The **packet type** is a 3-bit field containing the type of packet (scheduled ACK).
- The **redundancy** field is 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network. An ACK or BFCT should use the same redundancy path as the corresponding DP.
- The other 3-bits in the type field are reserved and are set to zero.

The **source logical address** is a one byte field that identifies the SpaceWire node sending the packet by its logical address. This is the logical address of the channel output node.

Each **channel number** is the number of a channel being acknowledged.

Each **sequence number** is a one byte field containing an 8-bit sequence count used to detect missing DPs. There is a separate sequence count for each channel.

The **header CRC** is a one byte field containing an 8-bit CRC covering the header of the packet. This uses the same CRC format as the SpaceWire RMAP standard (ECSS-E50-11). The header CRC covers the header from the Destination Logical Address to the byte immediately prior to the header CRC. It does not include the Destination SpaceWire Address as this is deleted during passage through the SpaceWire network. The header CRC is used to check that the header is correct before the packet is processed. If there is an error in the header the entire packet is discarded.

The **end of packet marker** is a SpaceWire control code that indicates the end of the SpaceWire packet and the start of the next one.

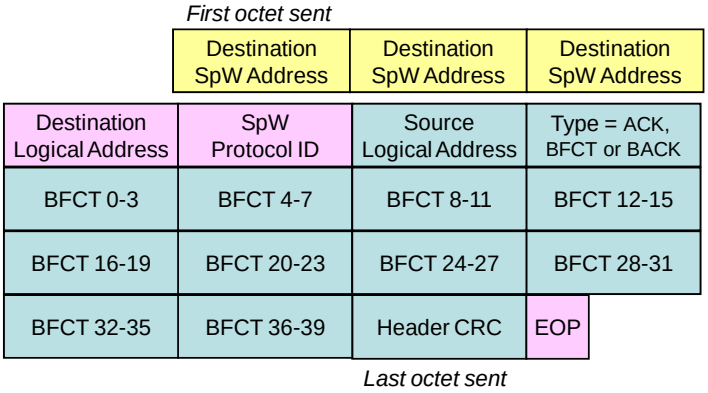

Figure 3-31 Scheduled BFCT Encapsulation

The scheduled BFCT is similar to other PDUs except that it contains a string of 2-bit codes representing the status of all the destination channel buffers in the destination node (output node). These 2-bit codes represent space for zero, one, two or three UDS in the corresponding destination channel buffer. The 2-bit codes are in channel number order starting with channel zero. All channels that are implemented in a destination are included in the scheduled BFCT information.

The fields of the scheduled BFCT are described below:

The **destination SpaceWire address** is a variable length field that contains the SpaceWire path and/or regional logical address that routes the packet across the SpaceWire network to the required destination. Since this is a BFCT the destination is the channel input node.

The **destination logical address** is a one byte field containing the logical address of the channel input node.

The **protocol identifier** is a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

The **source logical address** is a one byte field that identifies the SpaceWire node sending the packet by its logical address. This is the logical address of the channel output node.

The **type** field is an eight bit field comprising the following sub-fields:

- The **packet type** is a 3-bit field containing the type of packet (scheduled BFCT).
- The **redundancy** field is a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network. A BFCT should use the same redundancy path as the corresponding DPs.
- The other 3-bits in the type field are reserved and are set to zero.

Each **BFCT** field contains four 2-bit BFCT codes indicating the destination channel buffer status of four destination channel buffers. The first BFCT field contains information about channel 0-3, the next BFCT field relates to channels 4-7 and so on. Only the channels that are implemented in a node need to be included.

The **2-bit BFCT code** is 0 when there is no room in the destination channel buffer. Values of 1, 2 and 3 indicate room for 1, 2 or 3 maximum sized DPs, respectively.

The **header CRC** is a one byte field containing an 8-bit CRC covering the header of the packet. This uses the same CRC format as the SpaceWire RMAP standard (ECSS-E50-11). The header CRC covers the header from the Destination Logical Address to the byte immediately prior to the header CRC. It does not include the Destination SpaceWire Address as this is deleted during passage through the SpaceWire network. The header CRC is used to check that the header is correct before the packet is processed. If there is an error in the header the entire packet is discarded.

The **end of packet marker** is a SpaceWire control code that indicates the end of the SpaceWire packet and the start of the next one.

3.9.4 Network Management

This section has yet to be completed.

3.9.4.1 Channel Configuration

3.9.4.1.1 Open Channel

3.9.4.1.2 Close Channel

4 ASYNCHRONOUS NETWORK SPECIFICATION

4.1 USER APPLICATION INTERFACE

4.1.1 Source User Interface

a) The source user interface shall consist of one or more source channel buffers.

b) Each source channel buffer shall be associated with a destination channel buffer in a destination node.

c) There shall be a one to one association between a source channel buffer and a destination channel buffer.

d) Source channel buffers shall be grouped into destination groups i.e. according to the destination node that they are associated with.

e) A maximum of 256 source channel buffers shall be provided in a destination group.

f) Fewer than 256 source channel buffers may be implemented in a destination group.

g) Source channel buffers shall be numbered consecutively from zero.

h) Source channel buffer zero is reserved for configuration. It is mandatory to implement this channel.

i) All other source channel buffers except zero are available for user application.

j) The prime and zero or more alternative routes to the destination node shall be associated with the source channel buffer. Normally there is one prime route for each destination group and one route for the redundant and any other alternative route to the destination.

k) The following QoS parameters shall be associated with the source channel buffer:

   i. Type of service (Basic, Best Effort, Assured)

   ii. Priority

   iii. Retry prime attempts

   iv. Retry redundant attempts

   v. Prime path

   vi. Redundant path

l) The source channel buffer shall indicate to the source user application interface when it has room for more information to send to the destination channel buffer.

m) The source user application shall write data to be sent to the destination channel buffer into the source channel buffer associated with that destination channel buffer.

n) When a complete user application data unit has been written into a source channel buffer the source user application shall indicate to the source channel buffer that the end of a user application data unit has been reached. This is so that the contents can be transferred as soon as possible without waiting for more data to be put into the source channel buffer.

4.1.2 Destination User Interface

a) The destination user interface shall consist of one or more destination channel buffers.

b) Each destination channel buffer shall be associated with a source channel buffer in a source node.

c) There shall be a one to one association between a destination channel buffer and a source channel buffer.

d) Destination channel buffers shall be grouped into source groups i.e. according to the source node that they are associated with.

e) A maximum of 256 destination channel buffers shall be provided in a source group.

f) Fewer than 256 destination channel buffers may be implemented in a source group.

g) Destination channel buffers shall be numbered consecutively from zero.

h) Destination channel buffer zero is reserved for configuration.

i) All other destination channel buffers except zero are available for user application.

o) The prime and any alternative routes back to the source node for acknowledgements and BFCTs shall be associated with the destination channel buffer. Normally there is one prime route for each source group and one route for the redundant and any other alternative route to the source.

p) The following QoS parameters shall be associated with the destination channel buffer:

   i. Type of service (Basic, Best Effort, Assured)

   ii. Priority

   iii. Retry prime attempts

   iv. Retry redundant attempts

   v. Prime return path

   vi. Alternative return path

j) The QoS parameters shall match those of the corresponding source channel buffer.

k) The destination channel buffer shall indicate to the destination user application when it has data available.

l) The destination user application shall read data from the destination channel buffer.

m) Data should be read out in segments of 256 bytes except for the last segment in a SDU.

n) The destination channel buffer shall indicate to the destination user application when a complete user application data unit has been read out of the destination channel buffer.

4.2 SEGMENTATION

a) Data in a source channel buffer shall be split up into user data segments (UDS) no larger than the maximum UDS size of 256 bytes.

b) Each UDS shall be encapsulated in a DP for transferring across the SpaceWire network.

c) An UDS shall be put into a DP and sent across the network whenever there is room in the destination buffer for a complete maximum size DP, and when there is either

   i. 256 or more bytes of user data in the source channel buffer

   ii. Less than 256 bytes of user data followed by an end of user data character.

4.3 END TO END FLOW CONTROL

a) A source channel buffer shall only request to send data to a destination channel buffer when there is room for the maximum size UDS (256 bytes) in the destination channel buffer.

b) The destination channel buffer shall signal that space is available for another maximum size UDS by sending a buffer flow control token to the source channel buffer associated with the destination channel buffer.

c) The buffer flow control token shall contain the logical address of the destination node that contains the destination channel buffer, the channel number of the destination channel buffer, and a BFCT sequence number.

d) The BFCT sequence number shall increment each time enough space for a further maximum size UDS becomes available in the destination channel buffer

e) When the BFCT is incremented a BFCT shall be sent to the source node containing the source channel buffer that is sending data to the destination channel buffer.

f) The BFCT sequence number shall be an 8-bit count.

g) The BFCT sequence number shall wrap round to 0 when the maximum value (255) is reached.

h) When a BFCT is sent a time-out timer shall be started waiting on the arrival of an acknowledgement to the BFCT (BACK).

i) When a BFCT arrives at the source node it shall be passed to the source channel buffer identified by the destination logical address and channel number contained in the BFCT.

j) The source channel buffer shall keep a note of the sequence number of the last BFCT it received.

k) The source channel buffer shall keep a note of the amount of available space in the destination channel buffer.

l) If the sequence number of the BFCT received is greater or equal (modulo 256) to the sequence number of the last BFCT received then the source channel buffer

   i. Shall update its record of the amount of available space in the destination channel buffer accordingly,

   ii. Shall send an acknowledgement to the BFCT (BACK) containing the sequence number of the BFCT received.

m) If the sequence number of the BFCT received is less than the sequence number of the last BFCT received then the source channel buffer shall ignore the BFCT as it is a duplicate of a previously sent BFCT.

n) When a BACK is received at a node it shall be passed to the appropriate destination channel buffer as determined by the channel number.

o) The destination channel shall keep a record of the sequence number of the last BACK received.

p) If the sequence number of the BACK is greater than the sequence number of the last BACK received,

   i. The timers for all BFCTs with sequence numbers from one more than the last BACK received to the BACK just received shall be cancelled,

   ii. The record of the sequence number of the last BACK received shall be updated to the sequence number of the BACK just received.

q) If a BFCT time-out timer expires,

   i. A BFCT shall be sent with the sequence number of the highest (modulo 256) sequence number DP received in sequence,

ii. The time-out timer shall be restarted.

r) If a timer-out timer for a BFCT expire more that a maximum allowed number of times the local host system shall be informed of the error.

4.4 RETRY

4.4.1 Types of error:

a) The SpaceWire-RT receiver shall detect the following types of error:

i. Packet received with header error
ii. Packet delivered to wrong destination or with wrong SpaceWire Protocol ID
iii. Packet received with data error
iv. Missing or out of sequence packet
v. Duplicate packet
vi. SpaceWire Error End of Packet Error (EEP)

4.4.2 CRC generation and checking:

a) The header CRC shall be an 8-bit CRC which is the same as used for RMAP.

b) The header CRC shall cover all bytes in the header from the destination logical address to and including the byte immediately prior to the header CRC.

c) The data CRC shall be an 8-bit CRC which is the same as used for RMAP.

d) The data CRC shall cover all data bytes in the data field.

4.4.3 Sending a DP

a) Each source channel buffer shall have a last sent sequence number variable which shall hold the sequence number of the last DP sent from that source channel buffer.

b) Each source channel buffer shall have a last acknowledged sequence number variable which shall hold the sequence number of the last DP that has been acknowledged.

c) After power up or reset the last sent and last acknowledged sequence number variables shall be set to zero so that the sequence number of the first DP sent from each channel shall be one.

d) When the source channel buffer has an UDS ready to send and there is room for it in the destination channel buffer, it shall request for it to be sent it with the next sequence number

for that channel i.e. the sequence number will be one more (modulo 256) than that of the last sequence number variable for that channel.

e) When a DP is sent, a timeout timer shall be started waiting for an acknowledgement from that packet.

f) When a DP is sent, the last sent sequence number shall be incremented so that it contains the sequence number of the last DP sent.

g) The space in the source channel buffer for the UDS sent shall not be freed until an acknowledgement has been received for the corresponding DP.

h) The source channel buffer shall send no more than a maximum number of outstanding (unacknowledged) DPs before receiving acknowledgements for them.

i) Once this limit is reached the source shall not send any more packets until one or more acknowledgements have been received.

j) The maximum number of outstanding packets shall be a network management parameter.

4.4.4 Receiving a DP

a) When a DP is received without error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP:

   i. The UDS shall be extracted from the DP and placed in the destination channel buffer.

   ii. An acknowledgement shall be sent to the source of the DP containing:

      a. The logical address of the source that sent the DP being acknowledged,

      b. The logical address of the destination that received the DP and is sending the acknowledgement,

      c. The channel number from the DP, i.e. the channel number in the source,

      d. The sequence number from the DP.

   iii. The last received sequence number variable in the destination channel buffer shall be updated to the sequence number of the DP just received.

4.4.5 Receiving a DP when the Destination Channel Buffer is Full

a) When a DP is received without error and there is no room in the destination channel buffer specified by the source logical address and channel number:

   i. The DP shall be discarded.

   ii. The error shall be logged as a flow control error.

4.4.6 Receiving a Packet Containing Errors

a) When a packet is received with a header error indicated by the header CRC, or by an EOP or EEP being received before the header CRC is received:

   i. The packet shall be discarded,

   ii. The error shall be logged as a header error.

b) When a packet is received with a correct header CRC but the destination logical address does not match a logical address accepted by the destination node:

   i. The packet shall be discarded,

   ii. The error shall be logged as an invalid destination error.

c) When a packet is received with a correct header CRC, but the source logical address is not one that the destination node accepts data from:

   i. The packet shall be discarded,

   ii. The error shall be logged as an invalid source error.

d) When a packet is received with a correct header CRC, but the combination of the source logical address and the channel number does not identify a destination channel buffer:

   i. The packet shall be discarded,

   ii. The error shall be logged as an invalid channel error.

e) When a packet is received with a correct header CRC, but the sequence number is not equal to or one more than the value of the last received sequence number:

   i. The packet shall be discarded,

   ii. The error shall be logged as an out of sequence error.

f) When a packet is received with a correct header CRC, but the sequence number is equal to the value of the last received sequence number:

   i. The packet shall be discarded,

ii. The occurrence of a duplicate packet shall be logged.

g) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but a data CRC is detected after the UDS has been written to the destination channel buffer:

i. The DP shall be discarded,

ii. The UDS shall be removed from the destination channel buffer,

iii. The error shall be logged as a data CRC error.

h) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but an EOP occurs before the data CRC:

i. The DP shall be discarded,

ii. Any data written to the destination channel buffer from this DP shall be removed,

iii. The error shall be logged as an early EOP error.

i) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but an EEP occurs before or immediately after the data CRC:

i. The DP shall be discarded,

ii. Any data written to the destination channel buffer from this DP shall be removed,

iii. The error shall be logged as an EEP error.

j) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but the data field is larger than specified in the data length field:

i. The DP shall be discarded,

ii. Any data written to the destination channel buffer from this DP shall be removed,

iii. The error shall be logged as a data length error.

4.4.7 Receiving an Acknowledgement:

a) When an acknowledgement (ACK) is received it shall be passed to the appropriate source channel buffer as determined by the channel number.

b) The destination logical address in the acknowledgement shall be checked against the destination for the source channel buffer.

c) If the destination logical address in the acknowledgement does not match the destination for the source channel buffer,

   i. The acknowledgement shall be discarded

   ii. The error shall be logged as an ACK destination error.

d) If the sequence number in the acknowledgement is greater than the last acknowledged sequence number and less than or equal to the sequence number of the last DP sent,

   i. Each UDS with a sequence number greater than the last acknowledged sequence number and less than or equal to the sequence number of the acknowledgement:

      i. Shall have its time-out timer cancelled

      ii. Shall have the space in the source channel buffer for the UDS freed

   ii. The last acknowledged sequence number variable shall be set to the sequence number of the acknowledgement.

e) If the sequence number in the acknowledgement is less than or equal to the last acknowledged sequence number and less than or equal to the sequence number of the last DP sent,

   i. The acknowledgement shall be discarded,

   ii. The occurrence of a duplicate acknowledgement shall be logged.

f) If the sequence number in the acknowledgement is greater than the sequence number of the last DP sent,

   i. The acknowledgement shall be discarded,

   ii. An invalid acknowledgement sequence number shall be recorded.

4.4.8 Acknowledgement Time-Out:

a) When an acknowledgement time-out timer expires,

i. All acknowledgement time-out timers that are still running waiting for acknowledgements for DPs sent from the source channel buffer shall be cancelled.

ii. The unacknowledged DPs for the channel shall be resent.

iii. Acknowledgement time-out timers shall be started for all DPs resent.

iv. A retry count variable for the channel shall be incremented.

4.4.9 Retry Count:

a) The retry count shall determine over which path the resent packets are sent when automatic redundancy switching is enabled.

b) The retry count shall be used to flag to the network management system when failure of a path through a network has been detected.

c) If the retry count for a channel in the source ever equals the Retry Prime Limit managed parameter, then it is assumed that the prime path through the network from the source to the destination has failed. This shall be logged and flagged as a prime retry failure at the source.

d) If automatic redundancy switching is enabled and the retry count equals the Retry Prime Limit, the source shall start to send and resend all packets to the destination over the redundant route to the destination.

e) If automatic redundancy switching is not enabled and the retry count equals the Retry Prime Limit, then the source shall cease sending and resending all packets over the channel that has failed to deliver packets.

NOTE: It is then up to network management to switch to a redundant path or logical address and to re-enable packet transmission to the destination.

f) If in the source the retry count for a channel ever exceeds the Retry Redundant Limit managed parameter then it is assumed that the redundant path through the network from the source to the destination has failed:

a. This shall be logged and flagged as a redundant retry failure at the source

b. The source channel buffer shall then cease sending any packets until told that it may resume sending to the failed destination by network management.

4.5 REDUNDANCY

4.5.1 Redundancy Management

a) Automatic redundancy switching shall be enabled by network management parameter.

b) In the source node the address translation table shall determine the prime and redundant for each channel.

c) There may be more than two levels of redundancy (e.g. primary, secondary and tertiary paths) up to a maximum of four levels which shall be implemented by extending the address translation table accordingly.

d) A parameter (active path) shall determine which is the active path (or logical address) to use for each destination.

e) The active path parameter shall be set to prime on system reset or power up.

f) It shall be possible for network management to set the active path parameter at any time.

g) If automatic redundancy switching is enabled then the active path parameter may be altered from prime to redundant when the retry count equals the Retry Prime Limit.

h) A parameter (enable sending) for each channel shall be used to enable or disable the sending of data over the corresponding channel.

i) It shall be possible for network management to set the enable sending path parameter at any time.

j) When automatic redundancy switching is not enabled then if the retry count reaches the Retry Prime Limit for a particular channel then the enable sending parameter for the channel shall be set to disabled to stop further sending of data over that channel, until subsequent intervention by the network management system.

k) If the retry count reaches the Retry Redundant Limit for a particular destination then the enable sending parameter for the destination shall be set to disabled to stop further sending of data to that destination, until subsequent intervention by the network management system.

4.5.2 Retry and Redundancy for Different Types of Service

In this section the retry and redundancy facilities provided for each type of service are summarised.

4.5.2.1 Basic

a) For the Basic service no retries shall be permitted.

b) Redundancy switching for the Basic service shall performed by network management setting the active path management parameter in each source channel buffer and destination channel buffer.

4.5.2.2 Best Effort

a) For the Best Effort service no retries shall be permitted.

b) Redundancy switching for the Best Effort service shall performed by network management setting the active path management parameter in each source channel buffer and destination channel buffer.

4.5.2.3 Assured

a) For the Assured service retries shall be permitted.

b) Redundancy switching for the Assured service shall be performed either automatically or by network management.

4.6 ADDRESS TRANSLATION

a) SpaceWire logical addresses shall be used to identify nodes on a SpaceWire-RT network.

b) The maximum number of nodes permitted shall be 223 which is the maximum number of unreserved SpaceWire logical addresses.

c) A node may be identified by more than one SpaceWire logical address.

d) The address translation function shall translate from the SpaceWire logical address of a node to a SpaceWire address that is used to send a SpaceWire packet across the network.

e) The SpaceWire address may be a SpaceWire path address, logical address, regional logical address or an address constructed using any combination of these addressing modes.

f) An address translation table shall translate from the identity of the node that a SpaceWire packet is to be sent to (i.e. its logical address) and its priority level to one or more SpaceWire addresses.

g) There shall be SpaceWire addresses in the address translation table corresponding to the primary paths through the SpaceWire network to the intended destination nodes.

h) There shall be SpaceWire addresses in the address translation table corresponding to each set of alternative paths through the SpaceWire network to the intended destination nodes.

4.7 ENCAPSULATION

4.7.1 Data PDU

a) The DP encapsulation function shall encapsulate a UDS and associated parameters into a SpaceWire packet.

Note: The DP encapsulation is illustrated in Figure 4-1.

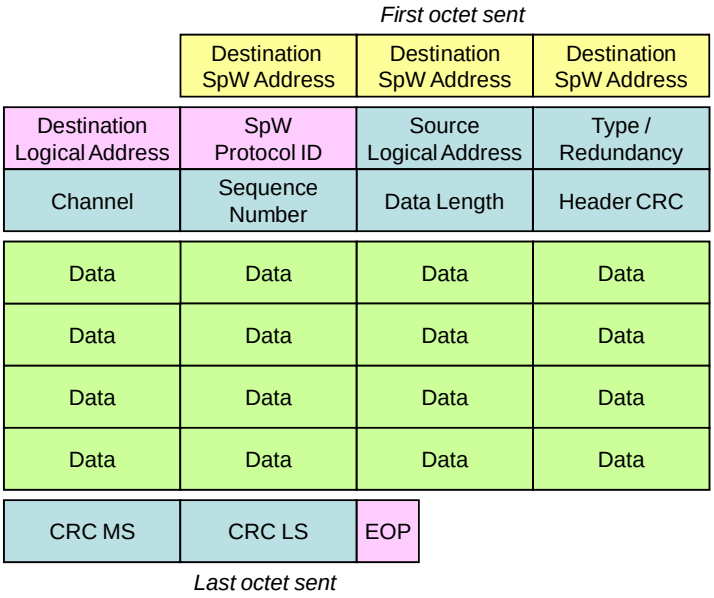

Figure 4-1 DP Encapsulation

b) The DP extraction function shall extract a UDS from a SpaceWire packet.

c) The destination SpaceWire address shall be a variable length field that contains the SpaceWire path and/or regional logical address that routes the packet across the SpaceWire network to the required destination (output node).

d) The destination logical address shall be a one byte field containing the logical address of the destination node (output node).

e) The protocol identifier shall be a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

f) The source logical address shall be a one byte field that identifies the SpaceWire node sending the packet (input node) by its logical address.

g) The source logical address shall also identify the group of destination channel buffers associated with the source node (input node).

h) The type field shall be an eight bit field comprising the following sub-fields:

   i. The packet type shall be a 3-bit field containing the type of packet (DP, ACK, BFCT, BACK, Scheduled ACK (SACK) or Scheduled BFCT (SBFCT)).

   ii. The redundancy field shall be a two bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network.

iii. An ACK or BACK should use the same redundancy index as the corresponding DP or BFCT.

iv. The start/end field contains 2-bits which indicate the position of the encapsulated UDS in the SDU as follows:

i. Start/end bits = 10: indicates the DP contains a UDS which is the start of a SDU

ii. Start/end bits = 01: indicates the DP contains a UDS which is the end of a SDU.

iii. Start/end bits = 00: indicates the DP contains a UDS which is the middle (neither start nor end) of a SDU.

iv. Start/end bits = 11: indicates the DP contains a UDS which holds an entire SDU.

v. The remaining 1-bit in the type field shall be reserved and set to zero.

i) The channel field shall (together with the destination and source logical addresses) identify the destination channel buffer.

j) The sequence number shall be a one byte field containing an 8-bit sequence count used to detect missing DPs and BFCTs. There is a separate sequence count for each channel and for DPs and BFCTs within a channel.

k) The data length shall be a one byte field that specifies the number of data bytes in the data field.

l) The header CRC shall be a one byte field containing an 8-bit CRC covering the header of the packet.

m) The header CRC shall use the same CRC format as the SpaceWire RMAP standard (ECSS-E50-11).

n) The header CRC shall covers the header from the Destination Logical Address to the byte immediately prior to the header CRC.

o) The header CRC shall not cover the Destination SpaceWire Address as this is deleted during passage through the SpaceWire network.

p) The header CRC shall be used to check that the header is correct before the packet is processed.

q) The data field shall be a variable length field containing up to 256 data bytes.

r) The data field shall contain the UDS.

s) The data CRC field shall contain an 8-bit CRC covering the data field only.

t) The data CRC shall use the same CRC format as the SpaceWire RMAP standard (ECSS-E50-11).

u) The data CRC shall be used to confirm that the data has been delivered without error.

v) The end of packet marker shall be a SpaceWire control code that indicates the end of the SpaceWire packet and the start of the next one.

4.7.2 ACK

The encapsulation of an acknowledgement (ACK) control code is illustrated in Figure 4-2.

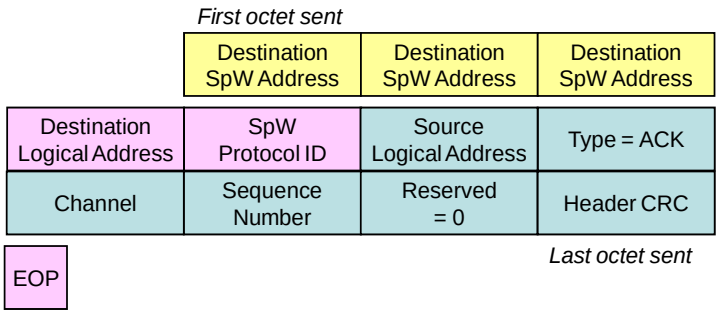

Figure 4-2 ACK Encapsulation

a) The destination SpaceWire address shall be a variable length field that contains the SpaceWire path and/or regional logical address that routes the acknowledgement across the SpaceWire network back to the source (input node) of the DP that is being acknowledged.

b) The destination logical address shall be a one byte field containing the logical address of the source of the DP being acknowledged i.e. the logical address of the input node.

c) The destination logical address shall be copied from the source logical address field of the DP being acknowledged.

d) The protocol identifier shall be a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

e) The source logical address shall be a one byte field that identifies the SpaceWire node sending the acknowledgement by its logical address.

f) The source logical address shall be copied from the destination field of the DP that is being acknowledged.

g) The type field shall be an eight bit field comprising the following sub-fields:

i. The packet type shall be a 3-bit field containing the type of packet i.e. an ACK.

ii. The redundancy field shall be a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network.

iii. The other 3-bits in the type field shall be reserved and set to zero.

h) The channel field shall (together with the destination and source logical addresses) identify the source channel buffer in the input node that sent the DP being acknowledged.

i) The sequence number shall be a one byte field containing the sequence number copied from the DP being acknowledged.

j) The reserved field shall be a one byte field that is set to zero.

k) The header CRC shall be a one byte field containing an 8-bit CRC covering the header of the packet.

4.7.3 BFCT

The encapsulation of BFCT control codes is illustrated in Figure 4-3.

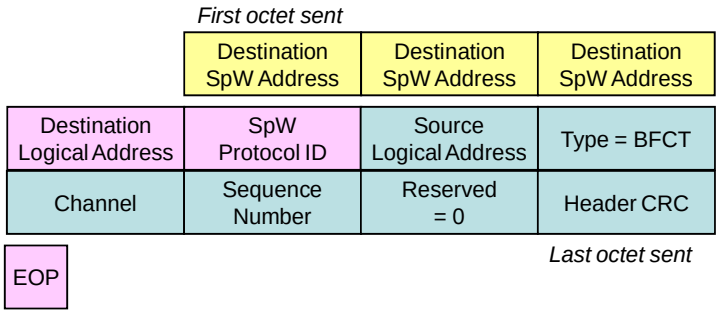

Figure 4-3 BFCT Encapsulation

a) The destination SpaceWire address shall be a variable length field that contains the SpaceWire path and/or regional logical address that routes the BFCT across the SpaceWire network to the source (input node) of DPs for the destination channel buffer that is sending the BFCT.

b) The destination logical address shall be a one byte field containing the logical address of the source (input node) of DPs for the destination channel buffer that is sending the BFCT.

c) The protocol identifier shall be a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

d) The source logical address shall be a one byte field that identifies the SpaceWire node sending the BFCT by its logical address.

e) The type field shall be an eight bit field comprising the following sub-fields:

   i. The packet type shall be a 3-bit field containing the type of packet i.e. a BFCT.

   ii. The redundancy field shall be a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network.

   iii. The other 3-bits in the type field shall be reserved and set to zero.

f) The channel field shall (together with the destination and source logical addresses) identify the source channel buffer in the input node to which the BFCT is being sent.

g) The BFCT sequence number shall be a one byte field containing an 8-bit sequence count used to detect missing BFCTs.

h) The reserved field shall be a one byte field that is set to zero.

i) The header CRC shall be a one byte field containing an 8-bit CRC covering the header of the packet.

4.7.4 BACK

The encapsulation of BACK control codes is illustrated in Figure 4-4.

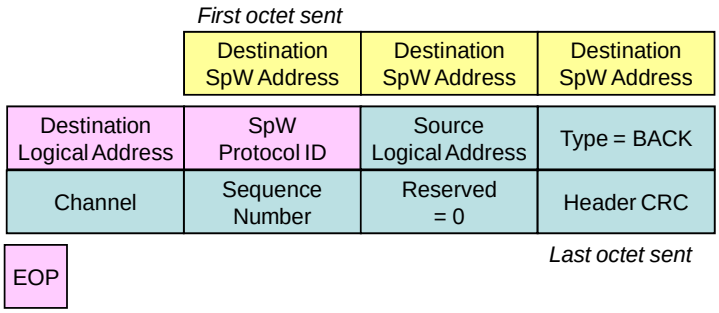

Figure 4-4 BACK Encapsulation

a) The destination SpaceWire address shall be a variable length field that contains the SpaceWire path and/or regional logical address that routes the packet across the SpaceWire network to the destination node (output node) containing the destination channel buffer that sent the BFCT being acknowledged.

b) The destination logical address shall be a one byte field containing the logical address of the destination node (output node) containing the destination channel buffer that sent the BFCT being acknowledged.

c) The destination logical address shall be a copy of the destination logical address in the BFCT that is being acknowledged.

d) The protocol identifier shall be a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

e) The source logical address shall be a one byte field that identifies the SpaceWire node (input node) sending the BACK by its logical address.

f) The source logical address shall be a copy of the source logical address in the BFCT that is being acknowledged.

g) The type field shall be an eight bit field comprising the following sub-fields:

   i. The packet type shall be a 3-bit field containing the type of packet i.e. a BACK.

   ii. The redundancy field shall be a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network.

   iii. The other 3-bits in the type field shall be reserved and set to zero.

h) The channel field shall (together with the destination and source logical addresses) identify the destination channel buffer in the output node that sent the BFCT being acknowledged.

i) The BFCT sequence number shall be a one byte field containing the BFCT sequence number copied from the BFCT that is being acknowledged.

j) The reserved field shall be a one byte field that is set to zero.

k) The header CRC shall be a one byte field containing an 8-bit CRC covering the header of the packet.

4.8 PRIORITY

a) A separate channel shall be used for each level of priority to be supported for each type of QoS and for each source/destination pair.

b) Priority shall be associated with the channel number.

c) A DP shall be sent from a source channel buffer with a lower channel number before sending a DP from a source channel buffer with a high channel number.

5 SCHEDULED NETWORK SPECIFICATION

5.1 USER APPLICATION INTERFACE

5.1.1 Source User Interface

a) The source user interface for a scheduled network shall be identical to that for an asynchronous network (see sub-clause 4.1.1).

5.1.2 Destination User Interface

a) The source user interface for a scheduled network shall be identical to that for an asynchronous network (see sub-clause 4.1.2).

5.2 SEGMENTATION

a) Segmentation for a scheduled network shall be identical to that for an asynchronous network (see sub-clause 4.2)

5.3 END TO END FLOW CONTROL

a) A source channel buffer shall only request to send data to a destination channel buffer when there is room for the maximum size UDS (256 bytes) in the destination channel buffer.

b) Each destination channel buffer shall signal the space available in the buffer by sending a scheduled buffer flow control token (SBFCT) near the start of each time-slot assigned to the source/destination pair for communication.

c) The scheduled buffer flow control token shall contain buffer status information on every destination channel buffer that connects to the present source node (input node).

d) The buffer status information shall be encoded in 2-bits as follows:

   i. 00 = no room in the destination channel buffer

   ii. 01 = room for one maximum sized UDS in the destination channel buffer

   iii. 10 = room for two maximum sized UDS in the destination channel buffer

   iv. 11 = room for three maximum sized UDS in the destination channel buffer

e) The buffer status information for the channel 0 shall be first followed by that for channel 1 and then sequentially all the other channels between the source and destination pair.

f) The scheduled buffer flow control token shall contain the logical address of the output node sending the token.

g) When a SBFCT arrives at the input node the destination channel buffer information it contains shall be passed to the set of source channel buffers identified by the source logical address of the SBFCT.

h) Each source channel buffer shall keep a note of the space available in the corresponding destination channel buffer.

5.4 RETRY

5.4.1 Types of error:

a) The SpaceWire-RT receiver in a scheduled network shall detect the following types of error:

i. Packet received with header error
ii. Packet delivered to wrong destination or with wrong SpaceWire Protocol ID
iii. Packet received with data error
iv. Missing packet
v. Duplicate packet
vi. SpaceWire Error End of Packet Error (EEP)

5.4.2 CRC generation and checking:

a) The header CRC used in a scheduled network shall be identical to that used in an asynchronous network (see sub-clause 4.4.2)

b) The data CRC used in a scheduled network shall be identical to that used in an asynchronous network (see sub-clause 4.4.2)

5.4.3 Sending a DP

a) Each source channel buffer shall have a last sent sequence number variable which shall hold the sequence number of the last DP sent from that source channel buffer.

b) Each source channel buffer shall have a last acknowledged sequence number variable which shall hold the sequence number of the last DP that has been acknowledged.

c) After power up or reset the last sent and last acknowledged sequence number variables shall be set to zero so that the sequence number of the first DP sent from each channel shall be one.

d) When the source channel buffer has a UDS ready to send and there is room for it in the destination channel buffer, it shall request for it to be sent it with the next sequence number

for that channel i.e. the sequence number will be one more (modulo 256) than that of the last sequence number variable for that channel.

e) Whether the destination channel buffer has any room shall be determined from the last received SBFCT and knowledge of any DPs sent over the channel since then.

f) When a DP is sent, the last sent sequence number shall be incremented so that it contains the sequence number of the last DP sent.

g) The space in the source channel buffer for the UDS sent shall not be freed until an acknowledgement has been received for the corresponding DP.

h) The source channel buffer shall send no more than a maximum number of outstanding (unacknowledged) DPs before receiving acknowledgements for them.

i) Once this limit is reached the source shall not send any more packets until one or more acknowledgements have been received.

j) The maximum number of outstanding packets shall be a management parameter.

5.4.4 Receiving a DP

a) When a DP is received without error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP:

   i. The UDS shall be extracted from the DP and placed in the destination channel buffer.

   ii. An acknowledgement shall be sent to the source of the DP containing:

      a. The logical address of the source (input node) that sent the DP being acknowledged,

      b. The logical address of the destination (output node) that received the DP and is sending the acknowledgement,

      c. The channel number from the DP,

      d. The sequence number from the DP.

   iii. The last received sequence number variable in the destination channel buffer shall be updated to the sequence number of the DP just received.

5.4.5 Receiving a DP when the Destination Channel Buffer is Full

a) When a DP is received without error and there is no room in the destination channel buffer specified by the source logical address and channel number:

   i. The DP shall be discarded.

   ii. The error shall be logged as a flow control error.

5.4.6 Receiving a Packet Containing Errors

a) When a packet is received with a header error indicated by the header CRC, or by an EOP or EEP being received before the header CRC is received:

   i. The packet shall be discarded,

   ii. The error shall be logged as a header error.

b) When a packet is received with a correct header CRC but the destination logical address does not match a logical address accepted by the destination node:

   i. The packet shall be discarded,

   ii. The error shall be logged as an invalid destination error.

c) When a packet is received with a correct header CRC, but the source logical address is not one that the destination node accepts data from:

   i. The packet shall be discarded,

   ii. The error shall be logged as an invalid source error.

d) When a packet is received with a correct header CRC, but the combination of the source logical address and the channel number does not identify a destination channel buffer:

   i. The packet shall be discarded,

   ii. The error shall be logged as an invalid channel error.

e) When a packet is received with a correct header CRC, but the sequence number is not equal to or one more than the value of the last received sequence number:

   i. The packet shall be discarded,

   ii. The error shall be logged as an out of sequence error.

f) When a packet is received with a correct header CRC, but the sequence number is equal to the value of the last received sequence number:

   i. The packet shall be discarded,

ii. The occurrence of a duplicate packet shall be logged.

g) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but a data CRC is detected after the UDS has been written to the destination channel buffer:

i. The DP shall be discarded,

ii. The UDS shall be removed from the destination channel buffer,

iii. The error shall be logged as a data CRC error.

h) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but an EOP occurs before the data CRC:

i. The DP shall be discarded,

ii. Any data written to the destination channel buffer from this DP shall be removed,

iii. The error shall be logged as an early EOP error.

i) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but an EEP occurs before or immediately after the data CRC:

i. The DP shall be discarded,

ii. Any data written to the destination channel buffer from this DP shall be removed,

iii. The error shall be logged as an EEP error.

j) When a DP is received without a header CRC error and its sequence number is one more than the last received sequence number for the destination channel buffer and there is room in the destination channel buffer specified by the source logical address and channel number in the DP but the data field is larger than specified in the data length field:

i. The DP shall be discarded,

ii. Any data written to the destination channel buffer from this DP shall be removed,

iii. The error shall be logged as a data length error.

5.4.7 Receiving an Acknowledgement:

a) A scheduled acknowledgement (SACK) shall contain the channel numbers and associated sequence numbers for every packet sent in the last time-slot that is using assured or guaranteed QoS.

b) A SACK shall be sent shortly after all the DPs being transferred in one time-slot have been received.

c) There shall be one SACK covering all DPs transferred in one time-slot.

d) The destination logical address in the acknowledgement shall be checked against the destination for the source channel buffer.

e) If the source logical address (output node) in the acknowledgement does not match the destination for the source channel buffer (output node),

   i. The acknowledgement shall be discarded

   ii. The error shall be logged as a SACK destination error.

f) When the SACK is received the individual acknowledgements for each channel covered by the SACK shall be extracted and passed to the appropriate source channel buffer as determined by the channel number.

g) If the sequence number in an acknowledgement for a channel is greater than the last acknowledged sequence number for that channel and less than or equal to the sequence number of the last PD sent,

   i. Each UDS with a sequence number greater than the last acknowledged sequence number and less than or equal to the sequence number of the acknowledgement:

      i. Shall have the space in the source channel buffer for the UDS freed

   ii. The last acknowledged sequence number variable for the channel shall be set to the sequence number of the acknowledgement.

h) If the sequence number in the acknowledgement for a channel is less than or equal to the last acknowledged sequence number for that channel and less than or equal to the sequence number of the last PD sent,

   i. The acknowledgement shall be discarded,

   ii. The occurrence of a duplicate acknowledgement shall be logged.

i) If the sequence number in the acknowledgement for a channel is greater than the sequence number of the last PD sent,

i. The acknowledgement shall be discarded,

ii. An invalid acknowledgement sequence number shall be recorded.

5.4.8 Acknowledgement Time-Out:

a) If the SACK is not received within a predetermined interval of time (TBA SpaceWire bit periods) after receiving a time-code, a retry count variable for the channel or group of channels shall be incremented.

5.4.9 Retry Count:

a) The retry count shall determine over which path the resent packets are sent when automatic redundancy switching is enabled.

b) The retry count shall be used to flag to the network management system when failure of a path through a network has been detected.

c) If in the source the retry count for a channel ever equals the Retry Prime Limit managed parameter, then it is assumed that the prime path through the network from the source to the destination has failed. This shall be logged and flagged as a prime retry failure at the source.

d) If automatic redundancy switching is enabled and the retry count equals the Retry Prime Limit, the source shall start to send and resend all packets to the destination over the redundant route to the destination.

e) If automatic redundancy switching is not enabled and the retry count equals the Retry Prime Limit, then the source shall cease sending and resending all packets over the channel that has failed to deliver packets.

NOTE: It is then up to network management to switch to a redundant path or logical address and to re-enable packet transmission to the destination.

f) If in the source the retry count for a channel ever exceeds the Retry Redundant Limit managed parameter then it is assumed that the redundant path through the network from the source to the destination has failed:

a. This shall be logged and flagged as a redundant retry failure at the source

b. The source channel buffer shall then cease sending any packets until told that it may resume sending to the failed destination by network management.

g) The Retry Redundant Limit shall not be set to a value less than or equal to the Retry Prime Limit.

5.5 REDUNDANCY

5.5.1 Redundancy Management

a) Automatic redundancy switching shall be enabled by management parameter.

b) In the source node the address translation table shall determine the prime and redundant for each channel.

c) There may be more than two levels of redundancy (e.g. primary, secondary and tertiary paths) up to a maximum of four levels which shall be implemented by extending the address translation table accordingly.

d) A parameter (active path) shall determine which is the active path (or logical address) to use for each destination.

e) The active path parameter shall be set to prime on system reset or power up.

f) It shall be possible for network management to set the active path parameter at any time.

g) If automatic redundancy switching is enabled then the active path parameter may be altered from prime to redundant when the retry count equals the Retry Prime Limit.

h) A parameter (enable sending) for each channel shall be used to enable or disable the sending of data over the corresponding channel.

i) It shall be possible for network management to set the enable sending path parameter at any time.

j) When automatic redundancy switching is not enabled then if the retry count reaches the Retry Prime Limit for a particular channel then the enable sending parameter for the channel shall be set to disabled to stop further sending of data over that channel, until subsequent intervention by the network management system.

k) If the retry count reaches the Retry Redundant Limit for a particular destination then the enable sending parameter for the destination shall be set to disabled to stop further sending of data to that destination, until subsequent intervention by the network management system.

5.5.2 Retry and Redundancy for Different Types of Service

In this section the retry and redundancy facilities provided for each type of service are summarised.

5.5.2.1 Best Effort

a) For the Best Effort service no retries shall be permitted.

b) Redundancy switching for the Best Effort service shall be performed by network management setting the “active path” and “enable sending” management parameters in source for each destination that the source sends data to.

5.5.2.2 Assured

a) For the Assured service retries shall be permitted.

b) Redundancy switching for the Assured service shall be performed either automatically or by network management.

5.5.2.3 Reserved

a) For the Reserved service no retries shall be permitted.

b) Redundancy switching for the Reserved service shall be performed by network management setting the “active path” and “enable sending” management parameter in source for each destination that the source sends data to.

5.5.2.4 Guaranteed

a) For the Guaranteed service retries shall be sent in another time-slot.

b) Redundancy switching for the Guaranteed service shall be performed either automatically or by network management.

5.6 ADDRESS TRANSLATION

a) The address translation used in a scheduled network shall be identical to that used in an asynchronous network (see sub-clause 4.6)

5.7 ENCAPSULATION

5.7.1 Data PDU

a) The DP encapsulation function shall encapsulate a UDS and associated parameters into a SpaceWire packet.

Note: The DP encapsulation is illustrated in Figure 5-1.

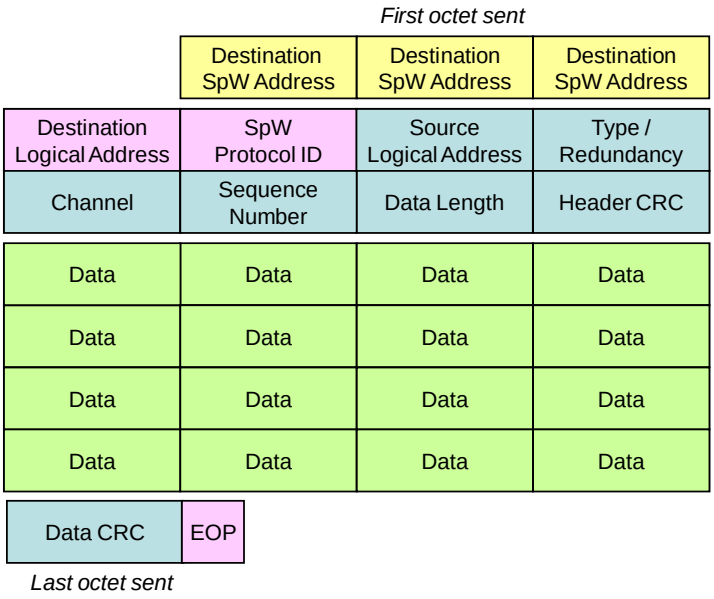

Figure 5-1 DP Encapsulation

b) The DP encapsulation used in a scheduled network shall be identical to that used in an asynchronous network (see sub-clause 4.7.1).

5.7.2 Scheduled ACK

The encapsulation of the Scheduled ACK (SACK) control code is illustrated in Figure 5-2.

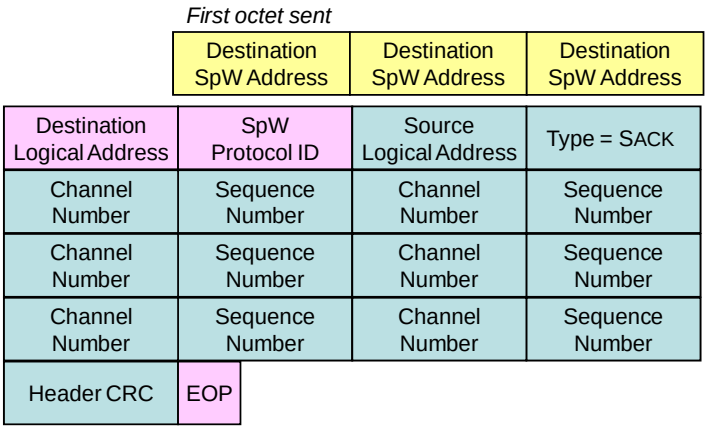

Figure 5-2 Scheduled ACK Encapsulation

a) The destination SpaceWire address shall be a variable length field that contains the SpaceWire path and/or regional logical address that routes the acknowledgement across the SpaceWire network back to the source (input node) of the DP that is being acknowledged.

b) The destination logical address shall be a one byte field containing the logical address of the source (input node) of the DP being acknowledged.

c) The destination logical address shall be copied from the source logical address field of the DP being acknowledged.

d) The protocol identifier shall be a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

e) The source logical address shall be a one byte field that identifies the SpaceWire node sending the acknowledgement by its logical address.

f) The source logical address shall be copied from the destination field of the DP that is being acknowledged.

g) The type field shall be an eight bit field comprising the following sub-fields:

   i. The packet type shall be a 3-bit field containing the type of packet i.e. a SACK.

   ii. The redundancy field shall be a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network.

   iii. The other 3-bits in the type field shall be reserved and set to zero.

h) The channel fields shall (together with the destination and source logical addresses) identify source channel buffers in the input node that sent the DPs being acknowledged.

i) There shall be a channel field for each channel being acknowledged.

j) Each channel field shall have a sequence field associated with it.

k) The sequence field shall be a one byte field containing the most recent in-sequence sequence number received on the associated channel.

l) The header CRC shall be a one byte field containing an 8-bit CRC covering the header of the packet.

5.7.3 SBFCT

The encapsulation of the Scheduled BFCT (SBFCT) control code is illustrated in Figure 5-3.

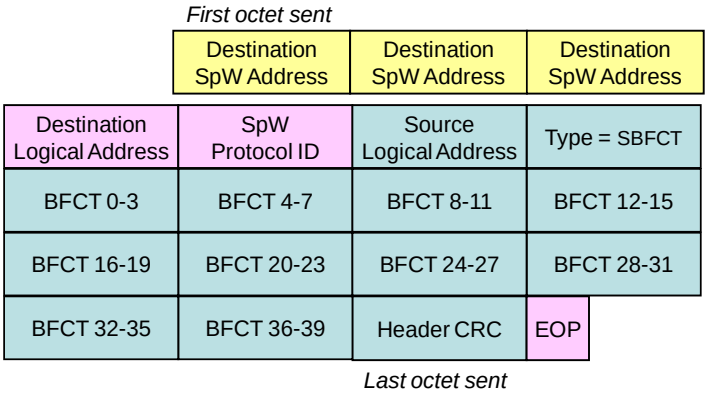

Figure 5-3 Scheduled BFCT Encapsulation

a) The destination SpaceWire address shall be a variable length field that contains the SpaceWire path and/or regional logical address that routes the SBFCT across the SpaceWire network to the source (input node) of DPs for the destination channel buffer that is sending the SBFCT.

b) The destination logical address shall be a one byte field containing the logical address of the source (input node) of DPs for the destination channel buffer that is sending the SBFCT.

c) The protocol identifier shall be a one byte field containing the SpaceWire-RT protocol identifier value (0x03).

d) The source logical address shall be a one byte field that identifies the SpaceWire node sending the SBFCT by its logical address.

e) The type field shall be an eight bit field comprising the following sub-fields:

   i. The packet type shall be a 3-bit field containing the type of packet i.e. a SBFCT.

   ii. The redundancy field shall be a 2-bit field that identifies which path (prime, redundant, other) the DP or control code is taking through the SpaceWire network.

   iii. The other 3-bits in the type field shall be reserved and set to zero.

f) Each BFCT field shall contain four 2-bit BFCT codes which indicate the destination channel buffer status of four destination channel buffers.

g) The first BFCT field shall contain information about channel 0-3, the next BFCT field information about channels 4-7 and so on.

h) Only the BFCT fields for channels that are implemented shall be included.

i) The 2-bit BFCT code shall be:

i. 0 when there is no room in the destination channel buffer.

ii. 1 when there is room for 1 maximum sized DP in the destination channel buffer.

iii. 2 when there is room for 2 maximum sized DP in the destination channel buffer.

iv. 3 when there is room for 3 maximum sized DP in the destination channel buffer.

j) The header CRC shall be a one byte field containing an 8-bit CRC covering the header of the packet.

5.8 PRIORITY

a) A separate channel shall be used for each level of priority to be supported for each destination.

b) Priority shall be associated with the channel number.

c) A DP shall be sent from a source channel buffer with a lower channel number before sending a DP from a source channel buffer with a high channel number.

d) The order of access to the media shall be dependent on two things: priority of data PDUs that are available to send and availability of space in the corresponding destination buffer.

e) The packet send next shall be the packet with highest priority that is both ready to send and for which there is space in the corresponding destination buffer.

5.9 RESOURCE RESERVATION

5.9.1 Time-Slots

a) Time-slot boundaries shall be distributed using SpaceWire time-codes.

b) There shall be 64 time-slots each identified by the time value (0-63) in the SpaceWire time-code.

c) Each time-slot starts on receipt of a time-code and ends on the receipt of the next time-code.

d) Time-codes shall be sent out at a uniform rate determined by a managed parameter, the time-code rate.

e) An Epoch shall be defined as being 64 time-slots in length starting with time-slot 0 and ending with time-slot 63.

f) A time-slot shall contain several time intervals as illustrated in Figure 5-4

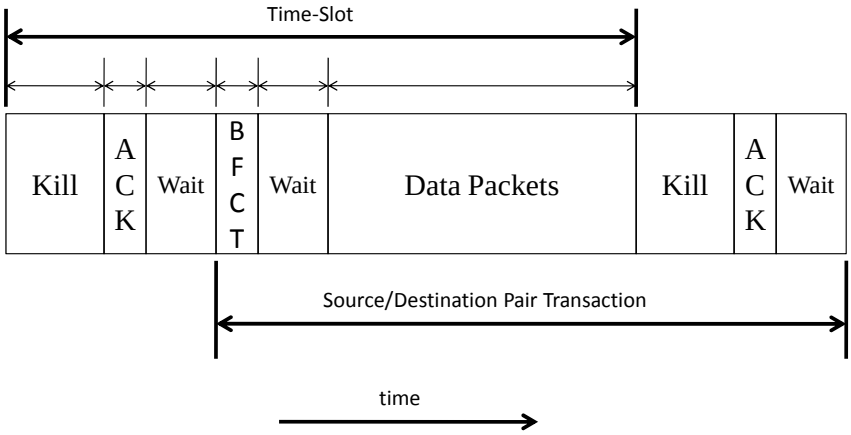

Figure 5-4 Time-Slot Timing

g) The time-slot shall begin with the “kill” interval during which any data PDUs still being sent are terminated immediately with an EEP.

h) The ACK interval shall follow the “kill” interval.

i) During the ACK interval the scheduled acknowledgements shall be sent for the data PDUs transferred between the source/destination pair serviced in the previous time-slot.

j) A “wait” interval shall follow the ACK interval during which time any scheduled ACKs can propagate across the SpaceWire network.

k) After the “wait” interval following the ACK interval, the scheduled BFCT interval shall start during which time the scheduled BFCTs are sent from the destination back to the source.

l) A “wait” interval shall follow the BFCT interval during which time any scheduled BFCTs can propagate across the SpaceWire network.

m) After the “wait” interval following the BFCT interval, the data PDU interval shall start during which time data PDUs may be sent from the source to the destination.

n) Multiple source/destination pairs may operate in the same time-slot provided that they do not use anywhere the same SpaceWire link.

5.9.2 Resources and Channels

a) A channel shall comprise a source channel buffer in the input node, a destination channel buffer in the output node and a router or path through the SpaceWire network from the source to the destination.

b) The channel shall be uniquely identified by the logical address of the source, the logical address of the destination and a channel number.

c) The channel number shall be an 8-bit value that identifies the source channel buffer in the source and the destination channel buffer in the destination for a particular channel.

d) The source channel buffer and destination channel buffer for a particular channel shall both have the same channel number.

5.9.3 Scheduled System

a) In a scheduled system the control of network traffic shall be done using time-slots.

b) In a time-slot a number of source/destination pairs shall be allowed to communicate concurrently provided that there is no conflicting use of resources i.e. communication between the different source/destination pairs must not use the same SpaceWire link. Note that this includes SpaceWire links in the middle of a SpaceWire network.

c) Channels with resource reserved or guaranteed QoS shall be assigned to time-slots in such a way that there is no conflicting use of resources.

d) The assignment of channels to time-slots shall be referred to as the schedule table.

e) When a specific time-slot starts, any source with a channel assigned to that time-slot may send a packet.

f) For the guaranteed service retries shall be sent in any time-slot scheduled to handle the same source/destination pair as the original packet provided that this time-slot is not needed for transfer of any other resource-reserved or guaranteed channel.

g) Specific time-slots shall be allocated for and guaranteed channel retries.

h) Retry shall be a go back N for a scheduled system where N is the maximum number of outstanding DPs for any channel (i.e. N = 3).

i) Time-slots that are scheduled for use by resource-reserved or guaranteed channels may be used by opportunistic channels when the scheduled channels have no more data to send.

5.9.4 Opportunistic Traffic

a) Basic, best effort and assured traffic in a synchronous system shall access network bandwidth on an opportunistic basis.

b) Basic, best effort and assured channels in a synchronous system shall be known collectively as opportunistic channels.

c) If the scheduled user (resource-reserved or guaranteed) of a time-slot allocated for data transfer between a particular source/destination pair does not have any more data to send and there are no outstanding retries to be sent between the particular source/destination pair, the unused capacity of the time-slot shall be allocated to an opportunistic channel.

d) The opportunistic channel between the particular source/destination pair being serviced by a specific time-slot that has data waiting to be sent, has space in the corresponding destination channel buffer, and that has the highest level of priority, shall be allowed to send one or more DPs in that time-slot, until the capacity of the time-slot has been used up.

e) Opportunistic channels shall only send DPs when there is space in the corresponding destination channel buffer.

f) An opportunistic assured channel that fails to successfully deliver a DP on the first attempt shall retry according to the current retry settings for that channel, in the next available time-slot between the relevant source/destination pair where there is spare capacity.

g) Assured retries shall have higher priority than other opportunistic traffic.

h) Many opportunistic communications may operate in parallel with other opportunistic or scheduled communications between different source/destination pairs provided that there is no two channels use the same SpaceWire link anywhere in the network.

5.9.5 Master Communication

a) Master communication shall take place in one or more time-slots scheduled for mastering.

b) During a mastering communication only one node shall have control of the entire SpaceWire network.

c) The master node shall be able to send commands and receive replies from any other node in the network, provided that the entire transaction can take place within the SBFCT, wait and data transfer interval of the time-slot.

d) At the end of the master time-slot any node still sending information shall terminate its data transfer prematurely with an EEP.

e) Several time-slots may be allocated to the same or different master nodes if required.

6 NETWORK CONFIGURATION PARAMETERS

This section has to be completed.

a) The network manager shall be responsible for network configuration.

b) The SpaceWire links shall be configured to AutoStart.

c) The routing tables in the SpaceWire router shall be configured by network management to provide the required paths for SpaceWire logical addresses.

d) Configuration of SpaceWire routers shall be performed using the SpaceWire RMAP protocol (ECSS-E50-11).

e) All SpaceWire links in the network shall be set to operate at the same data rate.

f) Automatic power-down on link silence may be enabled for SpaceWire links in the network to save power when the traffic over the link is expected to be sporadic.

7 DOCUMENT CHANGES

The changes made this document are listed in this section.

Table 7-1: Changes to Document (v1.1 to v2.0)	
Section/Reference	**Change**
Entire document	Major changes throughout document