SpaceWire-RT

Steve Parkes, Albert Ferrer-Florit
Space Technology Centre,
University of Dundee
Contents: Concepts

- SpaceWire protocol stack
- SOIS and the motivation for RT
- Key features of RT
- SpaceWire-RT functions
- Asynchronous SpaceWire-RT
- Scheduled SpaceWire-RT
- SpaceWire-T
SpaceWire Protocol Stack

- User Application
- CCSDS PTP
- SpaceWire
- RMAP
- SpW PnP
SpaceWire RMAP

- Remote Memory Access Protocol
- RMAP service runs over SpaceWire
- Means of reading and writing to memory
  - Of a remote node
  - Over SpaceWire network
- Supports
  - Device configuration, control & monitoring
  - Transfer of data from instrument to memory etc
SpW CCSDS Packet Transfer Protocol

- Transfers CCSDS Space Packets across a SpaceWire network
- Defines a common format for putting a CCSDS Space Packet into a SpaceWire packet
SpaceWire-PnP

- SpaceWire Plug and Play
- Mechanisms for:
  - Consistent configuration of nodes and routers
  - Discovering presence of nodes on a network
- Designed to use RMAP protocol
  - To reduce additional hardware needed
Why is SpaceWire successful?

- **Driving need**
  - For high-speed data links/networks on board spacecraft

- **Simple, flexible solution**
  - Easy to build a network architecture that suits a specific application

- **Standard**
  - Replacing proprietary solutions

- **Radiation tolerant components available**
  - Driven by ESA initially

- **Test and development equipment available**

- **It works**
Why is SpaceWire successful?

Instrument 1
High Rate

Instrument 2

Instrument 3

Instrument 4

Router

Sensor A

Sensor B

Sensor C

Bridge

Router

Prime PDHU

Mass Memory

Processor

Router

Redundant PDHU

Mass Memory

Processor
Centralised payload data-handling unit

- Most payload data-handling architectures have a centralised data handling unit
- Makes managing network resources easy
- Managing data transfers does not require any support from the network
- Can all be done under control of central payload data-handling unit
CCSDS SOIS
and
Motivation for SpaceWire-RT
CCSDS SOIS and motivation for SpW-RT

- **CCSDS SOIS**
  - Separates software applications from network
    - Application can then run over different networks
  - Aims to
    - Simplify software development
    - Enable reuse of software components
    - Integrate into broader set of CCSDS protocols
  - SOIS outlines set of services that network has to provide
CCSDS SOIS Serive Interface

Packet Service | Memory Access Service | Device Discover

PnP

SOIS Services

SpW-PnP

User memory control

QoS

SpW

SpaceWire
SOIS Services

- **Memory Access Service**
  - Reads and writes to "user" memory
  - Requires management of "user" memory
  - Various QoS requirements
  - "User" memory considered part of sub-network

- **Packet Delivery Service**
  - Delivers packets from source to destination
  - Requires management of packet buffers
  - Various QoS requirements

- **Device Discovery Service**
  - Discovers (new) devices on the network

- **Synchronisation Service**

- **Test Service**
QoS Requirements from CCSDS SOIS

- **Best Effort**
  - Single attempt to deliver
  - In order delivery

- **Assured**
  - Retry in event of failure to deliver

- **Resource Reserved**
  - Single attempt
  - Over reserved network resource

- **Guaranteed**
  - Retry in event of failure to deliver
  - Reserved network resource
Key Benefits of SpaceWire-RT
Key Benefits of SpW-RT

- Provides range of quality of service
  - Supporting different user requirements
- Helps to prevent network congestion
- Confirms delivery of data
- Ensures data delivery
- Delivers data within specific time bounds
- Works with existing SpaceWire devices and standards
Channels

- **Channel is:**
  - Set of network resources
  - Connects SpW-RT user in source
  - To SpW-RT user in destination
  - Uni-directional

- **Channel specified by**
  - Source / destination / channel number
  - E.g. 231 / 82 / 3
Asynchronous and Scheduled Systems

- Two types of system supported:
  - Asynchronous
    - Sending information is asynchronous
    - Best Effort and Assured QoS only
  - Scheduled
    - Network bandwidth split up using time-slots
    - Each source channel assigned one or more time-slots
      - When it is allowed to send data
    - Provides deterministic delivery
    - Support all SOIS QoS classes
SpaceWire-RT
Architecture and Functions
Architecture

- **User interface**
  - Interface to users of SpaceWire-RT

- **Segmentation**
  - Chops SDUs into chunks to send in Data PDUs (DPs)
  - Ensures that a large SDU does not hog the SpW network

- **Address translation**
  - SpW logical addresses used to identify nodes
  - Translates from logical address to path or logical address
  - Includes prime/redundant path addresses
Architecture

- **Retry**
  - Resends DPs that are lost or arrive with errors
  - Uses acknowledgement to confirm receipt

- **Redundancy**
  - Alternative paths through SpaceWire network

- **End to end flow control**
  - Check destination buffer ready before sending packet
  - Ensures that DPs accepted immediately by destination to prevent blocking
Architecture

- **Resource reservation**
  - **Asynchronous network:**
    - No reservation of resources
  - **Scheduled network:**
    - Time-codes sent periodically
    - Divide time into time-slots
    - One source can send in any one time-slot
      - Avoids conflicting use of network resources
    - Or several sources can send if they do not use the same network resources – i.e. links
Address Translation

- Nodes identified using SpaceWire logical addresses.
- Up to 223 logical addresses
  - Sufficient for most spacecraft applications
- Routing can be done with path and/or logical addressing
- Node identification done with logical addresses
## Address Translation

<table>
<thead>
<tr>
<th>SpaceWire Logical Address</th>
<th>Prime SpaceWire Address</th>
<th>Redundant SpaceWire Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
</tbody>
</table>
### Address Translation

<table>
<thead>
<tr>
<th>SpaceWire Logical Address</th>
<th>Prime SpaceWire Address</th>
<th>Redundant SpaceWire Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>124</td>
<td>1, 6, 5, 2, 124</td>
<td>2, 3, 5, 2, 124</td>
</tr>
<tr>
<td>Prime</td>
<td>Redundant</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td>1, 6, 5, 2, 124</td>
<td>2, 3, 5, 2, 124</td>
<td></td>
</tr>
</tbody>
</table>

Node 56

Router

Node 124

Node 132, 133

Router

Router

Router

Router

Router

Router

Router

Node 120

Prime 1, 6, 5, 2, 124

Redundant 2, 3, 5, 2, 124
## Address Translation

<table>
<thead>
<tr>
<th>SpaceWire Logical Address</th>
<th>Prime SpaceWire Address</th>
<th>Redundant SpaceWire Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td>124</td>
<td>1, 6, 5, 2, 124</td>
<td>2, 3, 5, 2, 124</td>
</tr>
<tr>
<td>150</td>
<td>1, 150</td>
<td>2, 150</td>
</tr>
</tbody>
</table>
Address translation

- Address tables for reply etc
  - Accessed via SLA, DLA, Ch#
  - Held in each node
  - May require updating if network changes

- Multiple network configuration regimes may be incorporated in the table
  - To allow rapid re-organisation of channel paths
  - Depending on network state
Redundancy

- Redundancy model:
  - Alternative paths from source node to destination node

- Managed and autonomous redundancy switching

- Redundancy supported in several ways:
  - Hot redundant
    - Send over both paths simultaneously
  - Warm redundant
    - Send over prime path
    - If failure send over redundant path
  - Cold redundant
    - Send over prime path
    - If failure power up redundant path and send over it
Hot Redundant
Warm Redundant
Cold Redundant
Redundancy Parameters

- Simultaneous retry on/off
- Number of attempts on prime path
- Autonomous reconfiguration enabled/disabled
- Number of attempts on redundant path
- Number of attempts on other alternative paths when appropriate
Example

- Try once on prime path & report failure
  - Simultaneous retry = off
  - Number attempts on prime path = 1
  - Autonomous reconfiguration = disabled
  - Number attempts on redundant path = 0
Example

- Try three times on prime path & report failure
  - Simultaneous retry = off
  - Number attempts on prime path = 3
  - Autonomous reconfiguration = disabled
  - Number attempts on redundant path = 0
Example

- Try twice on prime path, twice on redundant path & report failure
  - Simultaneous retry = off
  - Number attempts on prime path = 2
  - Autonomous reconfiguration = enabled
  - Number attempts on redundant path = 2
Example

- Try twice simultaneously on prime and on redundant paths & report failure
  - Simultaneous retry = on
  - Number attempts on prime path = 2
  - Autonomous reconfiguration = disabled
  - Number attempts on redundant path = 2
Asynchronous SpaceWire-RT

Retry
Flow-Control
Retry

- Retry function necessary for reliability
  - In conjunction with redundancy
- Resends any segment
  - that goes missing
  - or that arrives in error
- Means that applications do not have to worry about providing a retry mechanism
- Delivery is ensured
- Simplifies application development
- Efficient implementation
- Flexible
Retry: Normal Operation

Source

UDS A

Start Timer
Cancel Timer

DP A

ACK A

Destination

UDS A
End to End Flow Control

- Why do we need flow control?
  - SpaceWire uses worm hole routing
  - A blockage at a destination
  - Can cause disruption through network

- Two options
  - Throw away packets if no room in destination buffer
    - Wastes system bandwidth
    - Hinders timeliness
  - Use flow control
Flow-Control: Normal Operation

- Source:
  - UDS A
  - DP A
  - ACK A
  - BFCT 1
  - BACK 1
  - Buffer ready

- Destination:
  - UDS A
  - Buffer read
Scheduled SpaceWire-RT

Scheduled Network,
Retry,
Flow-Control
Scheduled Network

- Network bandwidth divided using time-slots
- Schedule table assigns communication to time-slots
- Avoids conflict
- Ensures deterministic delivery
Example Scheduling

Channel 41/70/1

Instrument 1
High Rate

Instrument 2

Instrument 3

Instrument 4

Channel 52/70/1

Router

Sensor A
Sensor B
Sensor C

Bridge

Router

LA 41
LA 52
LA 53
LA 54

LA 70
LA 80

Mass Memory
Processor
<table>
<thead>
<tr>
<th>Slot 0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>41/70/1</td>
<td>A, E/F</td>
</tr>
<tr>
<td>52/70/1</td>
<td>I, B, E/F</td>
</tr>
<tr>
<td>53/70/1</td>
<td></td>
</tr>
<tr>
<td>54/70/1</td>
<td></td>
</tr>
<tr>
<td>60/60/1</td>
<td></td>
</tr>
<tr>
<td>80/70/1</td>
<td></td>
</tr>
<tr>
<td>80/xx/1</td>
<td></td>
</tr>
</tbody>
</table>
Example Allocating Time-Slots

Channel 41/70/1
Link A

Channel 52/70/1
Link B

Links E/F
LA 70

Router

Mass Memory

Processor

LA 80

LA 41
Instrument 1
High Rate

LA 52
Instrument 2

LA 53
Instrument 3

LA 54
Instrument 4

Sensor A
Sensor B
Sensor C

Bridge
LA 60
## Allocating Channels to Time-Slots

<table>
<thead>
<tr>
<th></th>
<th>Slot 0</th>
<th>Slot 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>41/70/1</td>
<td>A, E/F</td>
<td>A, E/F</td>
</tr>
<tr>
<td>52/70/1</td>
<td>I, B, E/F</td>
<td></td>
</tr>
<tr>
<td>53/70/1</td>
<td></td>
<td>J, B, E/F</td>
</tr>
<tr>
<td>54/70/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60/60/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80/70/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80/xx/1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example Allocating Time-Slots

Channel 41/70/1
Link A

Instrument 1
High Rate

Instrument 2

Instrument 3

Instrument 4

Channel 53/70/1
Link B

LA 41

LA 52

LA 53

LA 54

Router

Mass Memory

Processor

Links E/F

LA 70

LA 80

Sensor A

Sensor B

Sensor C

Bridge

LA 60
## Allocating Channels to Time-Slots

<table>
<thead>
<tr>
<th></th>
<th>Slot 0</th>
<th>Slot 1</th>
<th>Slot 2</th>
<th>Slot 3</th>
<th>Slot 4</th>
<th>Slot 5</th>
<th>Slot 6</th>
<th>Slot 7</th>
<th>Slot 8</th>
<th>…</th>
<th>Slot 63</th>
</tr>
</thead>
<tbody>
<tr>
<td>52/70/1</td>
<td>I, B, E/F</td>
<td></td>
<td></td>
<td>I, B, E/F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53/70/1</td>
<td></td>
<td>J, B, E/F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54/70/1</td>
<td></td>
<td>K, B, G/H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60/60/1</td>
<td></td>
<td></td>
<td>C, E/F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80/70/1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>g/h, E/F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80/xx/1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E/F, g/h, a,b,c, l,j,k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Resources for Flow-Control and ACKs

- **Flow control & ACKs**
  - Travel in opposite direction to data
  - May conflict with other data PDUs

- **Resources have to be allocated for**
  - Flow control information
  - Data
  - Acknowledgements
Sending and Receiving Data

- **1. Buffer flow control**
  - Is there room in the destination buffer?
  - If there is no room in the destination
    - avoid sending data PDU or it will block the network

- **2. Send the data**
  - Send one or more data PDUs

- **3. Confirm that data arrived**
  - Did the data PDUs arrive ok?
Resources for Flow-Control and ACKs

- **Time-slots split into three parts:**
  - Flow control phase
    - Which channels for this time-slot have room in destination channel buffer?
  - Data PDU transfer phase
    - Send data PDUs
    - For channels with room in destination channel buffer
  - Acknowledgment phase
    - Send acknowledgement of receipt of data PDUs

- **This is the logical ordering**
Transferring data from node A to node B

Source/Destination Pair Transaction

Node A to B

Node B to A

BP
FP
DP
DP
DP
DP

BP
FP
DP
DP
DP
DP

ACK

BP
FP

time
Time-Slot and Transaction

Node A to B

Node B to A

Time-Code

Time-Slot
Resources for Flow-Control and ACKs

Timings are examples for hardware implementation. Demonstration software implementation takes about two to three times as long.
Resources for Flow-Control and ACKs

- This is the actual ordering
  - Receive time-code
  - Everyone stops sending
  - Wait long enough for network to become silent
  - Send acknowledgments for previous time-slot
  - Wait for ACKs to propagate across network
  - Send Buffer Flow Control Tokens (BFCTs)
  - To indicate room in destination buffers
  - Wait for BFCTs to propagate across network
  - Send Data PDUs
Scheduled Implementation

- When time-code received
  - Stop sending any more DPs
  - Wait
    - For network to become silent
  - Send ACKs for any DPs just received
  - Wait
    - For ACKs to propagate across network
  - Send Buffer Flow Control Tokens (BFCTs)
    - To indicate room in destination buffers
  - Wait
    - For BFCTs to propagate across network
  - Send any Data PDUs
node A to node B and node B to node A
Fault detection

- Can use bi-directional transfer capability to check for failures
- I.e. At start of transaction
  - Expect to receive BFCT from other end of source/destination pair
- At end of transaction
  - Expect to receive ACK from other end of source/destination pair
- One node is checking operation of other node
- Extend to all node checking that they are only receiving from devices they are permitted to receive from
SpaceWire-T
SpaceWire-T

- SpaceWire-RT without the R
- i.e. No reliability support
  - No retries
  - No redundancy
- Acknowledgement of data delivery is provided
- Mechanisms used are same as SpaceWire-RT
SpaceWire-T

- Over Asynchronous network provides:
  - Best Effort QoS
  - ACK for Best Effort QoS
    - So that the end user application is informed when something fails to be delivered

- Over Scheduled network provides:
  - Determinism
  - Resource Reserved QoS
  - With ACK
SpaceWire-T Architecture

TX User I/F

Segment

Channel Mux

Schedule Priority

Channel De-Mux

De-Segment

SpaceWire CODEC

RX User I/F

PDU Mux

Encapsulate

BFCT

ACK

PDU De-Mux

Decapsulate

Address Translate