SPACEWIRE-D: DETERMINISTIC DATA DELIVERY WITH SPACEWIRE

Session: SpaceWire Standardisation

Long Paper

Steve Parkes, Albert Ferrer,

Space Technology Centre, School of Computing, University of Dundee, Dundee, UK

E-mail: sparkes@computing.dundee.ac.uk

Stuart Mills, Alex Mason

STAR-Dundee Ltd, c/o School of Computing, University of Dundee, Dundee, UK

Email: stuart@star-dundee.com, alex@star-dundee.com

ABSTRACT

How can data be delivered deterministically over an existing SpaceWire network using existing SpaceWire components i.e. with no modification to the SpaceWire interface or router hardware? The approach explored in this paper is the sharing of system bandwidth using time-division multiplexing. The paper explores deterministic delivery over SpaceWire networks, describes how this can be achieved with current SpaceWire nodes and routers, provides corresponding theoretical performance estimates, and reports on a practical demonstration of SpaceWire-D.

1 INTRODUCTION

SpaceWire provides a versatile network architecture which is ideal for many space applications [1]. It has been widely used for payload data-handling on more than 30 space missions. SpaceWire is ideal for data-handling applications but does not address avionics and other applications where responsiveness, robustness, determinism and durability are essential requirements. There is a need for a spacecraft avionics network technology which combines the key features of SpaceWire with the quality of service requirements of real-time avionics applications. One critical requirement for avionics applications is deterministic delivery of information.

2 THE PROBLEM WITH DETERMINISTIC DATA DELIVERY OVER SPACEWIRE

Deterministic data delivery is the delivery of data within predetermined time-constraints: not too early and not too late. The essential consideration is a priori knowledge of when data will be delivered and the level of uncertainty in that knowledge. SpaceWire-D (where D stands for determinism) is a protocol that provides deterministic delivery over a SpaceWire network [2]. SpaceWire-D delivers data within predetermined time constraints.

SpaceWire is an asynchronous network which uses wormhole routing. The leading byte(s) on a SpaceWire packet determines the route through the network using path or

logical addressing. When the start of a packet arrives at a router it is switched to the required output port straightaway, provided that the required output port is not already being used to transfer another packet. Storing and forwarding of packets is not used by SpaceWire routers. This reduces the amount of buffer memory required in the routing switches. A disadvantage arises when the output port is already being used to transfer another packet and the packet has to wait for the output port to become free. The packet will be left strung out across the network from the blockage back to the source of the packet. It will prevent any other packet being transferred across the links that the packet is occupying.

This gives rise to the main difficulty in providing deterministic data delivery over SpaceWire: access to the SpaceWire network has to be controlled to avoid conflicting use of network resources. For example, two packets that need to travel down the same link at the same time will result in one packet having to wait while the other is transferred, leading to possible temporary blockage of the network resources that are being used by the waiting packet. The time of packet delivery thus depends on other packets flowing through the network. To ensure deterministic data delivery, all the traffic flowing through the network must be closely controlled.

A related problem occurs when a destination is not ready to receive and process a SpaceWire packet. If the destination node is not ready the packet will be strung out across the network blocking the links that it is resting on and temporarily preventing them from being used to transfer any other packets. So, not only is there a need to control packets going onto the network, but there is a need to ensure that once a packet arrives at its destination it is dealt with and removed from the network quickly.

Many components and sub-systems have already been designed using SpaceWire and ideally any deterministic data delivery mechanism for SpaceWire should be compatible with existing SpaceWire components. In any case it ought to be fully compatible with the SpaceWire standard.

3 SCHEDULING SPACEWIRE TRAFFIC

To prevent conflicting use of network resources the traffic has to be scheduled to avoid those possible conflicts. This requires a schedule table that defines which nodes can send packets at a particular time and some means of synchronising the nodes on the network so they can all follow the schedule. SpaceWire time-codes provide a mechanism for broadcasting time or synchronisation information across a SpaceWire network. The schedule is constructed from a series of periodic time-slots. The start of each time-slot is indicated by the arrival of a time-code so that all nodes are kept in synchronisation (see Figure 1).

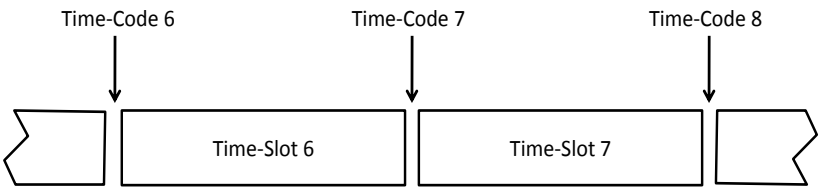

Figure 1 Time-Slots

4 WHAT SHOULD BE SCHEDULED?

Consider a system that schedules the sending of SpaceWire packets. A command may be sent in a SpaceWire packet to another node requesting that it returns some data. There is an asymmetry in the size of the packet used to send the command which is short and the reply containing the data which can be relatively long. Time-slots of equal period would not be very efficient in this case: a short time-slot is required followed by a longer one. If the command is sending data to the destination node and an acknowledgement is required then a long time-slot is required followed by a shorter one.

To mitigate the problem the complete transaction could be scheduled instead of the sending of individual SpaceWire packets. A transaction would then cover both sending the command and receiving the reply. The Remote Memory Access Protocol (RMAP) provides a transaction layer protocol which is able to read or write to memory in a remote SpaceWire node [3]. Designed for configuring, controlling and collecting data from instruments and other sub-systems it is being used in SpaceWire based systems. Components supporting RMAP include the ESA/UoD SpW-10X router [4] and ESA/Saab Remote Terminal Controller (RTC) device [5]. Missions using RMAP include Bepicolombo, ExoMars and MMS [6].

5 SPACEWIRE-D

To make maximum use of existing SpaceWire technology and devices, RMAP is used as the basic communication mechanism for SpaceWire-D and time-codes are used as the synchronisation mechanism.

Operation of the SpaceWire-D network is governed by a schedule that defines which node is allowed to initiate an RMAP transaction at any particular time. An example simple schedule table is illustrated in Figure 2.

Time-slot	0	1	2	3	...	63
Initiator allowed to initiate an RMAP transaction (logical address).	41	56	43	41		98

Figure 2 Example Simple Schedule Table

Each time-slot is long enough to allow one complete RMAP transaction to take place, allowing an RMAP initiator to read or write information to a remote RMAP target device. Since RMAP can transfer large amounts of data in a single transaction, which would take a long time, it is necessary to restrict the maximum amount of data that can be transferred in a single RMAP operation, so that the RMAP transaction does not take longer than one time-slot (but see section 6.3 later).

Each RMAP initiator has a copy of the schedule table. When a valid time-code arrives signalling the start of the next time-slot, each initiator checks whether it is allowed to start an RMAP transaction in that time-slot. If it has permission the initiator will send an RMAP command and wait for the RMAP reply. The RMAP command will travel across the SpaceWire network to the target node without any delay caused by network blockage, assuming that the schedule has been specified correctly.

When the RMAP command arrives at the target device, it is processed according to the RMAP standard and data read from or written to memory in the target device. It is important that this happens fairly quickly without the RMAP command being held up on the network. The RMAP reply containing data for a read command or an acknowledgement for a write command is then sent back across the network to the initiator.

Once the RMAP transaction is complete the initiator waits for the start of the next time-slot and then determines if it is allowed to send another RMAP command in the next time-slot.

6 SCHEDULE TYPES

There are three types of schedule defined in the SpaceWire-D specification: simple, concurrent and multi-slot schedules.

6.1 SIMPLE SCHEDULE

The simple schedule has been illustrated in Figure 2. It gives one specific initiator full control of the network for one or more specified time-slots. This means that when that initiator is permitted to send an RMAP transaction it may do so to ANY target node on the network. The RMAP transaction must start and finish in the same time-slot.

6.2 CONCURRENT SCHEDULE

The concurrent schedule makes more efficient use of network bandwidth by allowing more than one initiator to initiate RMAP transactions in a time-slot. Additional network performance is gained at the expense of a more complex schedule. This gives rise to the possibility that two initiators might attempt to use the same network resources (SpaceWire links) at the same time. The schedule table has to be constructed to prevent this. More than one initiator may initiate RMAP transactions in the same time-slot provided that the paths from each of the initiators to their targets do not use any of the same SpaceWire links in the network.

A typical application is on board data handling, where a mass memory unit is reading data from each instrument and writing data to a telemetry system, while a control processor is controlling instruments and monitoring housekeeping information. An example schedule table is illustrated in Figure 3.

Time-slot	0	1	2	3	...	63
Control Processor Targets	41, 43, 44, 45,	42, 43	42, 43	40, 41, 43, 44,		40
Mass Memory Targets	40	41	41	42		49

Figure 3 Example Concurrent Schedule Table

In this example the control processor can initiate RMAP transactions with one of several target devices listed in the schedule table for each time-slot. The mass memory device is gathering information from one target device in each time-slot. The targets that the control process can communicate with are carefully chosen to avoid any network resource conflicts with the transactions that the mass memory device is initiating in each time-slot.

6.3 MULTI-SLOT SCHEDULE

The multi-slot schedule builds on the concurrent schedule to improve network efficiency further. Where a large amount of data has to be transferred between two nodes, the RMAP transaction to accomplish this is permitted to occupy more than one adjacent time-slot. This allows more data to be transferred in the one RMAP transaction. The schedule has to ensure that no conflict of network resources occurs over the duration of this extended RMAP transaction. Additional network performance is one again achieved at the expense of a more complex schedule.

The schedule table has to ensure that when an RMAP transaction duration has more than one time-slot, it does not use the same network resources (SpaceWire links) as any other transactions occurring during any of those time-slots. An example schedule table is illustrated in Figure 4.

Time-slot	0	1	2	3	...	63
Control Processor Targets	41	42	43			40
Mass Memory Targets	40	41		42		49

Figure 4 Example Multi-Slot Schedule Table

In this example the mass memory initiates a long RMAP transaction with target 41 in time-slot 1. This transaction is expected to complete within two time-slots.

7 INITIATOR AND TARGET CONSTRAINTS

The time for the complete RMAP transaction to take place must be within the limits of the time-slot, or one transaction will have an impact on subsequent transactions. To achieve this, certain constraints are placed on the initiator and the target nodes. SpaceWire-D initiator and target implementations are fully compatible with the SpaceWire and RMAP standard. The constraints listed place some restrictions on the implementation covering the amount of data that can be transferred in a single RMAP transaction and the speed of response of the initiator and target devices.

7.1 INITIATOR CONSTRAINTS

The following constraints apply to the RMAP Initiator:

- The maximum amount of data that can be read in an RMAP read command or written in an RMAP write command is 256 bytes (TBC). This limits the size of the RMAP write command or RMAP read reply so that it does not exceed the duration of the time-slot. There is a trade-off between amount of data transferred in a single RMAP transaction and performance. Sending more data is more efficient giving an improved overall data rate, but the time-slot period then has to be longer making the delivery of data less timely.

- The maximum amount of data may be longer than 256 bytes when multi-slot scheduling is being used.

- The time taken from the receipt of a time-code to starting to send out an RMAP command from an initiator must be less than 5 μs (TBC). Note if this is difficult to achieve with a specific implementation of an initiator, operating

a local clock synchronised to time-codes might help with achieving this requirement.

7.2 TARGET CONSTRAINTS

The following constraints apply to the RMAP targets:

- The time taken from receipt of the complete RMAP command header in a target node to the authorisation or rejection of that RMAP command must be less than 5 μs (TBC).
- The latency in transferring data from SpaceWire interface to memory must be less than 5 μs (TBC).
- The time taken from completion of writing data to memory to starting to send the RMAP command must be less than 5 μs (TBC).

These constraints are readily met by the SpW-10X and RTC devices.

8 PERFORMANCE

In this section the anticipated performance of SpaceWire-D is considered. The operation of an initiator and target performing a write operation during a time-slot is illustrated in Figure 5.

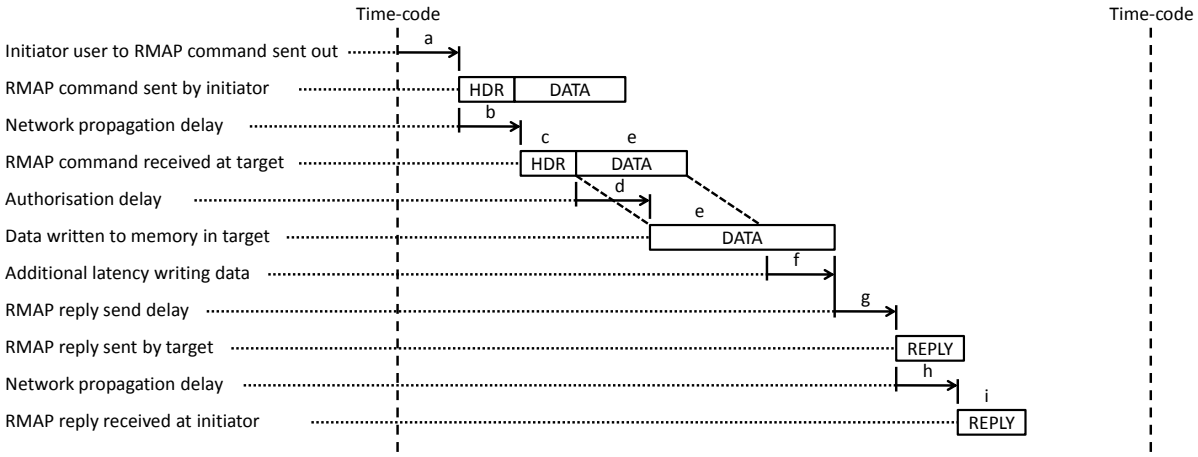

Figure 5 Performance of RMAP Write

The receipt of a time-code starts off the activities indicated in Figure 5. The following paragraphs explain each time interval labelled in Figure 5.

a) Interval from receipt of time-code to the RMAP command starting to be sent by the initiator. This interval includes: the time to receive, decode and respond to the time-code; the time to check the schedule table; the time to start to send out the RMAP command (assuming that the command has already been prepared ready for sending). This interval is entirely dependent upon the initiator implementation.

b) Interval for the SpaceWire packet containing the RMAP command to propagate across the SpaceWire network from initiator to target. This will mainly depend upon the number of routers between the initiator and the furthest target node. Assuming a time delay per router of 0.6 μs, the total propagation delay will be 0.6R μs where R is the number of routers in the longest path used between an initiator and a target.

c) Interval for sending the RMAP header, including any path address bytes. The size of the RMAP header including the SpW Target Address and Reply Address is H=R+16+P bytes, where R is the number of routers in the path from initiator to target and P is a the closest multiple of 4 which is greater than or equal to R. Thus if there are four router R=4 and P=4, so H=24 bytes. The time to send this header depends upon the SpaceWire data rate, S Mbits/s, and is 10H/S μs. For example with H=24 and S = 200 Mbits/s, T_c = 1.2 μs.

d) Interval for authorising the RMAP command once the header has been received. This is dependent upon the target implementation.

e) Interval to send the data and data CRC. This is given by 10(D+1)/S, where D is the number of data bytes. For D = 256 (the maximum amount of data permitted in a SpW-D RMAP write command or read reply) the time to send the data and data CRC is T_e = 12.85 μs when S = 200 Mbits/s.

f) Assuming that the target is able to write data to memory as fast as the SpaceWire network can deliver it, this interval covers any additional latency in the transfer of data from the SpaceWire interface to memory. It is dependent upon the implementation of the target node.

g) Interval from the completion of writing data to memory in the target to starting to send the RMAP reply. This interval is dependent upon the implementation of the target node.

h) Interval for the SpaceWire packet containing the RMAP reply to propagate across the SpaceWire network from target to initiator. This will mainly depend upon the number of routers between the initiator and the furthest target node. Assuming a time delay per router of 0.6 μs, the total propagation delay will be 0.6R where R is the number of routers in the longest path used between an initiator and a target. This interval is the same as (b).

i) Interval for sending the RMAP reply, including any path address bytes. The size of the RMAP reply including the Reply Address is E=R+8 bytes, where R is the number of routers in the path from target to initiator. Thus if there are four router R=4, E=12 bytes. The time to send this header depends upon the SpaceWire data rate, S Mbits/s, and is 10E/S μs. For example with E=12 and S = 200 Mbits/s, T_i = 0.6 μs.

The total time for the complete transaction is:

$$T_{total} = T_a + T_b + T_c + T_d + T_e + T_f + T_g + T_h + T_i$$

The corresponding performance of a SpaceWire-D network using simple scheduling is illustrated in Figure 6. The RMAP read operation has similar performance. The following assumptions were made:

- The link data rate is 200 Mbits/s,
- A simple schedule is used with one initiator initiating transactions at any time,
- The traffic comprised both payload data transfers (256 bytes per RMAP transaction) and command and control information (4 bytes per RMAP transaction) with an average of 132 bytes per RMAP transaction.

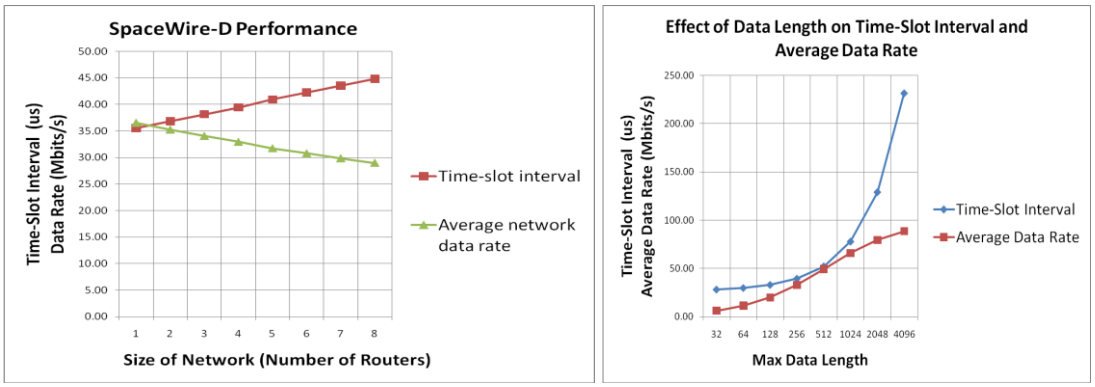

Figure 6 SpaceWire-D Performance

For a network with several layers of routing and links running at 200 Mbits/s, the overall data rate of the simple schedule system is around 30 Mbits/s (real data rate).

Initial SpaceWire-D prototyping done at Dundee to confirms this performance [7].

The performance of a system using concurrent scheduling depends on the number of concurrent initiators and the extent to which they can be scheduled to avoid use of common SpaceWire links. The maximum performance is N times that of the simple schedule, where N is the number of concurrent initiators.

The multi-slot schedule can substantially improve performance when there are large amounts of data to be transferred.

9 DEMONSTRATION SYSTEM

An implementation of SpaceWire-D has been developed by STAR-Dundee and the University of Dundee. This demonstration system used STAR-Dundee SpaceWire interface and routing devices. It also tested operation with an RTC device [7]. A screen shot of the demonstration system is shown in Figure 7.

Figure 7 Screen Shot from SpaceWire-D Demonstration System

10 CONCLUSIONS

SpaceWire-D provides a means of providing deterministic data delivery over SpaceWire. It builds on RMAP and is fully compliant to the existing SpaceWire standard. Furthermore existing components can be used in many cases without modification provided they meet some straightforward timing constraints. A demonstration system has been developed which showed SpaceWire-D operating as expected with multiple initiators and different types of target device.

11 REFERENCES

1. ECSS, "SpaceWire - Links, nodes, routers and networks", ECSS-E-ST-50-12C, July 2008, available from http://www.ecss.nl.

2. S. Parkes and A. Ferrer-Florit, "SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks", ESA Contract No. 220774-07-NL/LvH, University of Dundee, April 2010, available from http://spacewire.esa.int/WG/SpaceWire/.

3. ECSS, "SpaceWire - Remote memory access protocol" ECSS-E-ST-50-52C, 5 February 2010, available from http://www.ecss.nl.

4. Atmel, "AT7910E SpW-10X SpaceWire Router", available from http://www.atmel.com/dyn/products/devices.asp?family_id=641.

5. ESA/Saab, "SpaceWire Remote Terminal Controller", http://spacewire.esa.int/content/Devices/RTC.php.

6. D. Roberts and S. Parkes, "SpaceWire Missions and Applications", International SpaceWire Conference, St Petersburg, Russia, June 2010.

7. A. Ferrer and S. Parkes, "SpaceWire-D Prototyping", International SpaceWire Conference, St Petersburg, Russia, June 2010.