LOW-LATENCY PACKET DELIVERY IN SPACEWIRE NETWORKS

Session: Networks and Protocols

Long Paper

Dr Barry M Cook, C Paul H Walker

4Links Limited

E-mail: barry@4Links.co.uk paul@4Links.co.uk

ABSTRACT

This paper quantifies typical latency requirements and describes a simple technique that uses virtualization and priorities with dynamic, on-demand segmentation, to provide deterministic, low latency delivery of packets whilst allowing high utilisation of the network. Segmentation is low-level and invisible to the user (and to the API). Packets, of any size, will be delivered to the destination node as a contiguous whole, without interleaving and with the contents in strict order. The technique offers the ability to carry data with real-time and low-latency requirements, such as command and control (including unscheduled events), at the same time as, and completely fire-walled from, high-bandwidth data such as that from experiments and instruments.

1 INTRODUCTION

This paper brings together with SpaceWire the themes of Virtualization and Time Triggering. Virtualization has proved to be a secure way to share resources on a computer, such as virtual servers. Time Triggering has become a popular means of providing deterministic (but often very high) latency over a bus.

SpaceWire, as defined in [1], offers very low latency for real-time control loops and for housekeeping accesses, provided that such accesses are not contending with large data transfers, or with blockage in the network.

We describe an enhancement to SpaceWire that provides Virtual SpaceWire Networks [2, 3] and thus brings the benefits of protecting users from each other and of isolating faults. Virtual SpaceWire Networks (VSNs) retain the exceptionally low latency of current SpaceWire for deterministic real-time traffic. Meanwhile, the low priority and bulk data transfer can use all the bandwidth that is not taken by the real-time traffic.

We use priority within the Virtual SpaceWire Networks to ensure that latencies and control loop times can be guaranteed. Each Virtual Network can have its own individual priority, or several Virtual Networks can share the same priority.

In this paper, we acknowledge that many missions have control loops and housekeeping accesses that repeat at 1 second, 100ms, 10ms or shorter periods. We simply group the accesses in each of these sets of periods into separate Virtual SpaceWire Networks.

2 AN EXAMPLE NETWORK AND CALCULATION OF AVAILABLE THROUGHPUT

Table 1 below shows an example set of traffic such as would be used on a satellite and Figure 1 shows a subset of the activity of that traffic on several Virtual SpaceWire Networks sharing a single physical SpaceWire link.

The table and figure include traffic for a couple of 5kHz control loops which need to access data and process it within 200µs. We give this highest priority (priority 1). Priorities 2 to 5 are used for slower control loops or for regular housekeeping updates, at frequencies from 1kHz down to 1Hz (priorities 3 to 5 are omitted from the figure). The lowest priority (6) is used for bulk data, such as image data from cameras.

3 ASSUMPTIONS

Any calculation of performance or guarantees needs to be based on assumptions. We'll make our assumptions explicit and then describe, in general terms, the way the numbers in the table have been calculated and what performance can be guaranteed.

1. We assume a worst-case that all the traffic is shared on a single Virtual SpaceWire Network link. In practice, performance scales with additional links.

2. We assume that the accesses are RMAP (Remote Memory Access Protocol) Read requests and responses. RMAP has higher overheads than some other protocols used on SpaceWire, so our use of RMAP here gives conservative results, and this analysis is in no way confined to use of the RMAP protocol.

3. We assume a very worst case that all the RMAP initiators, with all the different frequencies, start sending their requests at the same time and so will be queued.

Frequency	Number of requests in period	Response Payload, Bytes
5kHz	2	20
1kHz	10	50
100Hz	25	200
10Hz	50	200
1Hz	100	200

Table 1: Set of real-time traffic used as an example

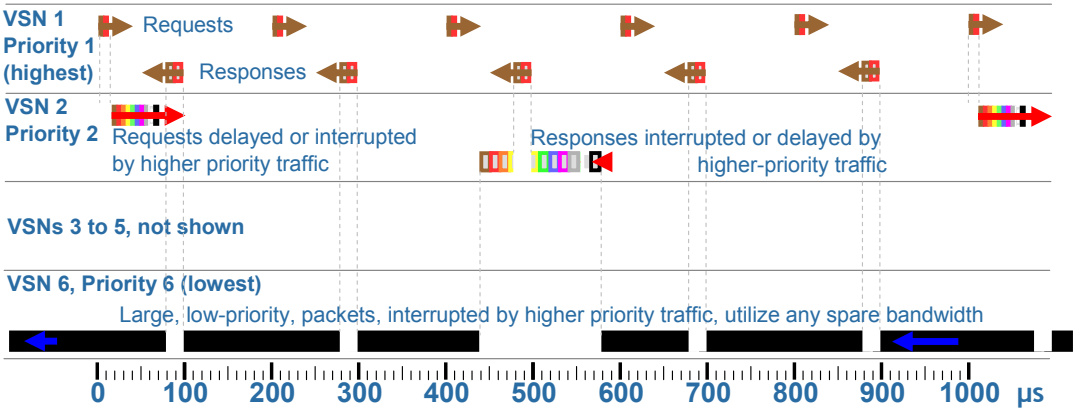

Figure 1: An example of activity on a Virtual SpaceWire Network link over time

4. We assume that a node transmitting an RMAP request has adequate buffer space to receive the full response that it has requested without stalling. And we assume that a node transmitting a response is able to send the complete response without stalling.

5. We assume that all the nodes are current SpaceWire standard nodes, so that a node can only send requests or responses to one Virtual Network. (There can be benefits in having nodes that support multiple Virtual Networks, which will be described later).

6. We assume a link speed of 50Mbits/s. While this amply meets the requirements, even with the worst case assumptions used here, faster link speeds could be used to carry more data or to give even faster responses.

With these assumptions, we can now look at an algorithm for calculating whether the latencies and processing times meet the requirements for completion within the relevant Period at a particular priority.

4 ALGORITHM FOR CALCULATING REAL-TIME AND THROUGHPUT PERFORMANCE

1. For each VSN, add up the **Network delay for requests on this VSN in this period** from the start of the first request being transmitted to the end of the last request reaching its target. These delays include:
    - Delay in the Initiator in transmitting the first packet
    - The transmission time for the total number of Bytes, in all the requests for this VSN, at the link speed of the SpaceWire link
    - An overhead on the transmission time, to allow for flow control characters, an occasional Time Code, and for the possible overhead of switching between Virtual Networks
    - The transmission time for any Nulls that the Initiator inserts into a request (some designs may be unable to send contiguous packets)
    - Total cable delay (although this is probably negligible)
    - Total Routing-Switch latency

2. For each VSN, determine the **Longest target latency** of the various targets on this Virtual SpaceWire Network. This is the time from the end of the request packet to the start of the response packet. The latency should be determined from the manufacturer's data sheet and confirmed by characterization with test equipment [4]

3. For each VSN, add up the **Network delay for responses on this VSN in this period** between start of the first response being transmitted to the end of the last response reaching its initiator. These delays are similar to the network delays for requests and include:
    - Delay in the Target transmitting the first packet
    - The transmission time for the total number of Bytes, in all the responses, at the link speed of the SpaceWire link; note that, while most of the requests are the same length as each other (or very similar) the responses may vary in length depending on the nodes being accessed.

- An overhead on the transmission time, to allow for flow control characters, an occasional Time Code, and for the possible overhead of switching between Virtual Networks
- The transmission time for any Nulls that the Target inserts into a transmitted request (there should not be any but some designs may not be able to send contiguous packets)
- Total cable delay (although this is probably negligible)
- Total Routing Switch latency

4. For each VSN, determine the **Longest processing time** of the various controllers/initiators on this Virtual SpaceWire Network. The processing time is the time from the end of the response packet arriving at the initiator to the end of any action it needs to take as a result of the response.

5. For each VSN, add up the following:
   - the **Network delays for requests on all (higher or equal)-priority VSNs in this Period**. Note that if there is a single VSN at the highest priority, this sum will be zero. Note also that the Network delays must account for *all* the delays at equal or higher priority during the Period of this VSN.
   - the **Network delay for requests on this VSN in this Period**
   - the **Longest target latency**
   - the **Network delays for responses on all (higher or equal)-priority VSNs in this Period**. Note that if there is a single VSN at the highest priority, this sum will be zero. Note also that the Network delays must account for *all* the delays at equal or higher priority during the Period of this VSN.
   - the **Network delay for responses on this VSN in this Period**
   - the **Longest processing time** [1]

   and if this total is less than the Period, then the set of accesses can be guaranteed to take place within the Period.

[1] It may be excessively conservative, on top of the worst case assumptions, to include the longest target latency and the longest processing time in the calculation. An alternative would be to sum the {target latency plus processing time} for each separate access, and then add the longest of these sums to the request and response network delays to ensure that the total is less than the Period.

5 A SPECIFIC EXAMPLE

We'll consider, as an example, a small subset of the activity shown in Figure 1, and check the behaviour for the highest priority Virtual SpaceWire Network.

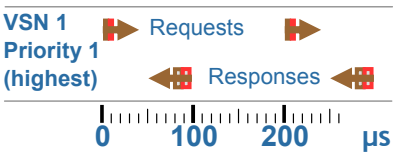

Figure 2: Detail of activity for an example 5KHz control loop

For this there are two initiators and two targets with the initiators and targets sharing a single link between two routing switches. The numbers we use are arbitrary, but are a reasonable estimate based on products that 4Links have characterized. Note that the period of 200µs implies 5kHz control loops, and that these have relaxed constraints on the end-nodes even with a link speed of 50Mbits.

The Delay in both the Initiators in transmitting the first packet is 1µs.

The transmission time for an RMAP Read Request is around 24 Bytes, making a total of 48 Bytes for the two Initiators. At 50Mbits/s, allowing for eight bits of data in ten bits transmitted, and for the Ends of Packet, that takes a total of 9.8µs. We'll allow a 10% overhead on the transmission time, to cover flow control characters and the possible overhead of switching between Virtual Networks. This brings the total transmission time to 10.8µs.

At 50Mbits/s, these Initiators do not insert Nulls into the data stream unless they are starved of flow-control credit, which should not occur when the Target is waiting for a Request.

Cable delay, at under 5ns per metre, and a total cable length of less than ten metres, is sub 50ns and so will be ignored.

Routing Switch latency, for several switches that 4Links has measured, is around 1µs. With two Routing Switches, the total delay in this direction is 2µs. This makes a total **Network delay for requests** of (1+10.8+2) = **13.8µs**.

The **Longest target latency** depends heavily on the devices used and whether the protocol and response are handled in hardware or software. In this case we take, as an example, an RMAP target that uses a processor and software dedicated to the one target and that it responds in **50µs**.

The **Network delay for responses** is a similar calculation to that for requests. In this case any equivalent of the Initiator delay is included in the target latency. The response payload is 20Bytes and the RMAP overhead is another 20Bytes. These 40Bytes for each of the two responses at 50Mbits/s, take 16.2µs. We again allow a 10% overhead on this to arrive at a transmission delay of 17.8µs. There should again be no Nulls inserted because the initiator should not request a response that it can not handle. Routing Switch Latency is as for Requests, at 2µs. This makes a total **Network delay for responses** of (17.8+2) = **19.8µs**

The **Longest processing time** needs to be measured by test equipment or calculated from simulation of the software or, perhaps preferably, by both. In this example, the sum of the two network delays, (13.8+19.8) = **34µs** plus the target latency of **50µs**, is 84µs. With the Period of 200µs, this leaves considerably more than **100µs** available for the processing time.

A more complete example of these calculations is given in [5] and is summarized in Table 2.

Virtual Space-Wire Network (VSN)	Prio-rity	Period, µs	Freq-uency	Number of requests in period (n)	Response Payload, Bytes	Worst case network delay for requests on this VSN in this period, µs (% of period)	Worst case network delay for responses on this VSN in this period, µs (% of period)	Shared link Request direction utilization	Shared link Response direction utilization
1	1	200	5kHz	2	20	13.8 (6.9%)	19.8 (9.9%)	5.4%	8.9%
2	2	1000	1kHz	10	50	57 (5.7%)	157 (15.7%)	5.4%	15.5%
3	3	10000	100Hz	25	200	138 (1.4%)	1214 (12.1%)	1.3%	12.1%
4	4	100000	10Hz	50	200	273 (0.3%)	2428 (2.4%)	0.3%	2.4%
5	5	1000000	1Hz	100	200	542 (0.1%)	4853 (0.5%)	0.1%	0.5%
						Real-Time utilization		12.5%	39.5%
6	6	**Available bandwidth for (lowest priority) bulk data**						>80%	>50%
		Total network utilization possible						>90%	>90%

Table 2: Calculations of latencies and bandwidth for the traffic shown in Table 1

6 More Complex SpaceWire Networks

The calculations above were done for a single SpaceWire link, and obviously SpaceWire is used for more complex topologies, such as the ring shown at right. A conservative measure (again worst-case) of both real-time and data throughput performance could be gained by simply treating the whole ring as a single SpaceWire link. If that gives adequate performance, no further work is necessary. If more performance is needed, each link between routing switches can be considered separately — which is still a much simpler calculation than would be needed for a conventional time-triggered network.

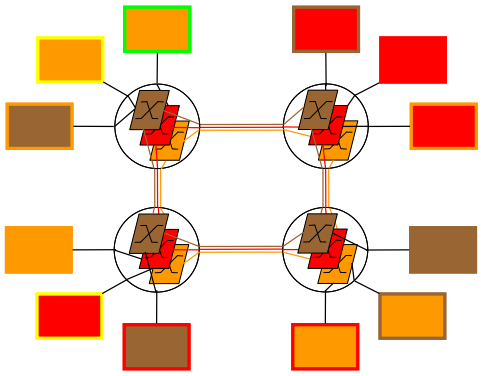

Figure 3: Ring network using Virtual SpaceWire Networks between routing switches

7 Reducing Power and Harness Mass

Most current recommendations for SpaceWire are that all the SpaceWire links should run at the same speed. Otherwise, for a current SpaceWire link multiplexing traffic between many nodes (as in the main example above), the throughput on the shared link drops to the throughput of the slowest link. Virtual SpaceWire Networks remove this dependency, and so permit the peripheral links to nodes or end-points to run at the appropriate speed for the node rather than for the whole network. Reducing link speed at the periphery of the network can result in substantial savings of power (and cost).

The major saving in harness mass is from using a single (Virtual) SpaceWire network instead of one (SpaceWire) network for data and another network/bus for control. Significant additional savings in harness mass can be gained compared with current SpaceWire by multiplexing several slow links over one faster link.

8 COMPATIBILITY WITH EXISTING NODES

All the examples shown have been with all the nodes being current SpaceWire. No change to hardware or software is necessary to any well designed node connected to a Virtual SpaceWire Network routing switch.

9 BENEFITS OF VSN NODES

Nodes that are accessed both for housekeeping and for large volumes of data could benefit from having access both to high priority traffic for the housekeeping and to low priority for the data. As described in [3, 4], such nodes could have a separate SpaceWire link for each priority, or use an extended CODEC that supports two or more priorities/VSNs. Nodes supporting multiple priorities must separate the two or more priority levels to prevent priority inversion.

10 FAULT ISOLATION AND RECOVERY

It is possible for a SpaceWire node to block another, by continuously transmitting (babbling idiot) or by failing to transmit for lack of flow-control credit or other error such as software stalling. Virtual SpaceWire Networks provide isolation for such faults:

1. **From faults at lower priority:** The highest priority Virtual SpaceWire Network sees an empty network, and is completely isolated from faults in lower-priority virtual networks. In general any VSN is isolated from faults in any VSN at lower priority than itself.

2. **From faults at the same or higher priority:** It might appear from (1.) that faults on higher priority VSNs are able to block everything at a lower priority. But, as in our example above, it will be normal for the real-time traffic to take a small proportion of the overall network bandwidth. VSN routing switches could police that proportion and discard data from a node that is exceeding the permitted percentage of bandwidth utilization for that priority level. This would leave up to 80% of the bandwidth available to lower priorities, even in the event of a fault at the highest priority — bandwidth which could be used to recover from the fault.

3. **From faults within a single Virtual SpaceWire Network:** Several existing routing switches perform a gatekeeper function by setting timeouts so that, if a node is blocked for longer than the timeout, the blocked packet is discarded. It can be difficult to calculate the appropriate timeout value if the minimum value to meet one criterion is longer than the maximum value to meet another criterion. This issue is much simpler to resolve when there is a separate VSN for each frequency of access, and the timeouts can be set appropriately for each VSN.

11 CONCLUSIONS

We have presented here a simple solution for supporting real-time requirements on a SpaceWire network. A link speed of 50Mbits/s is amply able to meet the requirements of 5kHz control loops.

The solution can, at the same time and with conservative (worst case) assumptions, offer well above 50% of the network bandwidth for volume data transfers, that may use very large packets, while still providing microsecond response times to real-time traffic.

One of the Virtual SpaceWire Networks could be used for a time triggered protocol.

By replacing only the routing switches in a SpaceWire network, Virtual SpaceWire Networks provide the following benefits for missions:

- the simplicity in both concept and use of Virtual SpaceWire Networks, with a corresponding reduction in mission complexity;
- use of a single physical network for both command/control and, separated by a firewall, for volume data;
- reduction in power consumption, cable/harness mass, and thence cost
- complete compatibility with existing SpaceWire nodes;
- complete compatibility with (and transparency to) higher-level protocols (including CCSDS, SOIS and PnP) running over SpaceWire;
- consistency with the layering of the SpaceWire standard so that no change is required to the ECSS SpaceWire standard;
- greatly improved fault-isolation and recovery.

12 REFERENCES

1. ECSS Secretariat, "ECSS-E-ST-50-12C 31 July 2008, SpaceWire - Links, nodes, routers and networks", ESA-ESTEC, Requirements & Standards Division, Noordwijk, The Netherlands
2. B M Cook (4Links) "Virtual Networks", SpaceWire Working Group Meeting 13, ESTEC, Noordwijk, The Netherlands, 2009-09-16
3. 4Links, "Virtual SpaceWire Networks", White Paper, 4Links Limited, UK, 2009-09-03
4. 4Links, "Characterizing SpaceWire Devices and Networks", White Paper, 4Links Limited, UK
5. 4Links, "Virtual SpaceWire Networks: Example latency and throughput calculations", White Paper, 4Links Limited, UK