"There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies and the other is to make it so complicated that there are no obvious deficiencies."
Professor Sir C.A.R "Tony" Hoare *(inventor of Quicksort and CSP)*

"Make it as simple as possible, but not simpler"
Albert Einstein

"Entia non sunt multiplicanda praeter necessitatem"
William of Occam *(the principle known as Occam's Razor)*

"A designer knows that he has arrived at perfection not when there is no longer anything to add, but when there is no longer anything to take away."
Antoine de Saint-Exupéry

"Our life is frittered away by detail - Simplify, simplify".
Henry David Thoreau

"I have made this letter longer than usual, because I lack the time to make it short."
Blaise Pascal

These quotes were collected in one place by Dick Pountain – we acknowledge and thank him for doing so.

Introduction

Virtual Networks – two or more independent networks sharing common hardware – offer Real-Time performance whilst maximising use of existing SpaceWire software and interfaces. The only component needing to be changed is the routing switch. ALL existing nodes and software can be re-used without change.

The ability to utilise the same hardware for guaranteed fast delivery of critical command and control data and for bulk data transfers will allow significant mass and power reductions compared with multiple bus architectures such as IEEE1553 for control with SpaceWire for data.

One of the several advantages to Virtual Networks is the unification of SpaceWire and SpaceFibre – a notable advance towards a complete solution for data networks.

We have built a proof-of-concept demonstrator that we will bring to SpW WG13 – and will make evaluation hardware available to interested parties soon thereafter. We also plan to make silicon (commercial and flight) available within the next few months.

Content

Time Slots and Virtual Networks - A Comparison

There is a proposal to provide consistency of data delivery times by dividing time into a number of slots and allocating data flows to these slots in order to avoid contention for a given link. This solution appears not to fully meet user requirements (such as the often mentioned need for low latency) and to be complex when compared with Virtual Networks.

A summary comparison of time-slot and Virtual Networks is shown here – more detail is provided in following sections.

Feature	Time-slot proposal	Virtual Network proposal
Low latency data delivery	No. 10ms at 200Mb/s link speed 200ms at 10Mb/s link speed	Yes. ~1us at 100Mb/s link speed ~10us at 10Mb/s link speed
Low jitter delivery	Yes. ~10us (?)	Yes. ~10us
Compatible with existing SpaceWire silicon/software	No. ALL nodes (hardware and software) must be changed	Yes. Keep all existing hardware and software
Large packet support	No. Large packets must be broken into a sequence of small packets	Yes. Large packets are sent unchanged.
Self-timed	No. Needs accurate time distribution.	Yes.
Low-overhead retries	No. Time slots have to be reserved for retries – leading to poor utilisation when no retries are needed (most of the time!)	Yes. Ad hoc retries when required are very low impact
Inherent babbling idiot control	No. Needs an additional access control mechanism.	Yes.
Meets CCSDS/SOIS requirements	Yes, but much more complex than necessary.	Yes.
Mixed speed networks	No	Yes
Control of babbling nodes	No. Requires an additional router function	Yes
Common model for SpaceWire and SpaceFibre	No	Yes
Able to use existing routers	No. The whole network must support time slots.	Yes. Replace only those routers that need to pass real-time traffic

Analysis

"Real-time" requires that actions be performed against hard time deadlines – either for synchronisation or as part of a feedback loop that must remain stable. This implies that data must be delivered with low-latency and/or low-jitter.

Low latency data delivery is especially important in feedback loops where there is a cycle of data transmission and data processing. Any delay in delivery reduces the time available for processing. Data that is delivered before it must be used can always be delayed – action scheduling, where required, can always be applied on top of a low-latency delivery system.

Typical existing control loops require updates at 100ms intervals, with some robotics applications demanding 1ms updates and, to allow sufficient data processing between updates, must be sure of data delivery times much shorter than this. Urgent but unpredictable data, such as fault reports or emergency actions (traditionally the province of "interrupts"), also need very short delivery times.

Existing SpaceWire networks use wormhole routing to transfer data as fast as possible through the network with low-level flow control to eliminate buffer over-runs and lost data. One consequence of this combination is that a packet occupying a route through a network can block other traffic on the links used, leading to delays for other traffic. Some applications use multi-mega-byte packets that occupy routes for significant fractions of a second, much longer than required by the control loops.

The requirement for low-latency delivery can be met using networks with an excess of capacity over demand – but this results in poor use of resource (mass, power, money).

Best use of resource, on the other hand, suggests using a single network for all traffic, combining real-time streams with less critical data streams to use as much of the available network resource as possible.

The ability to handle large volume data at the same time as time critical (and usually relatively short) packets appears, in current proposals, to require significant changes to the fundamental principles of SpaceWire networks – adding complexity such as limits on packet sizes combined with strict control of when and where all traffic flows. These proposals aim to provide predictable bandwidth at the cost of extended delivery times (latency) and major changes to all nodes on the network.

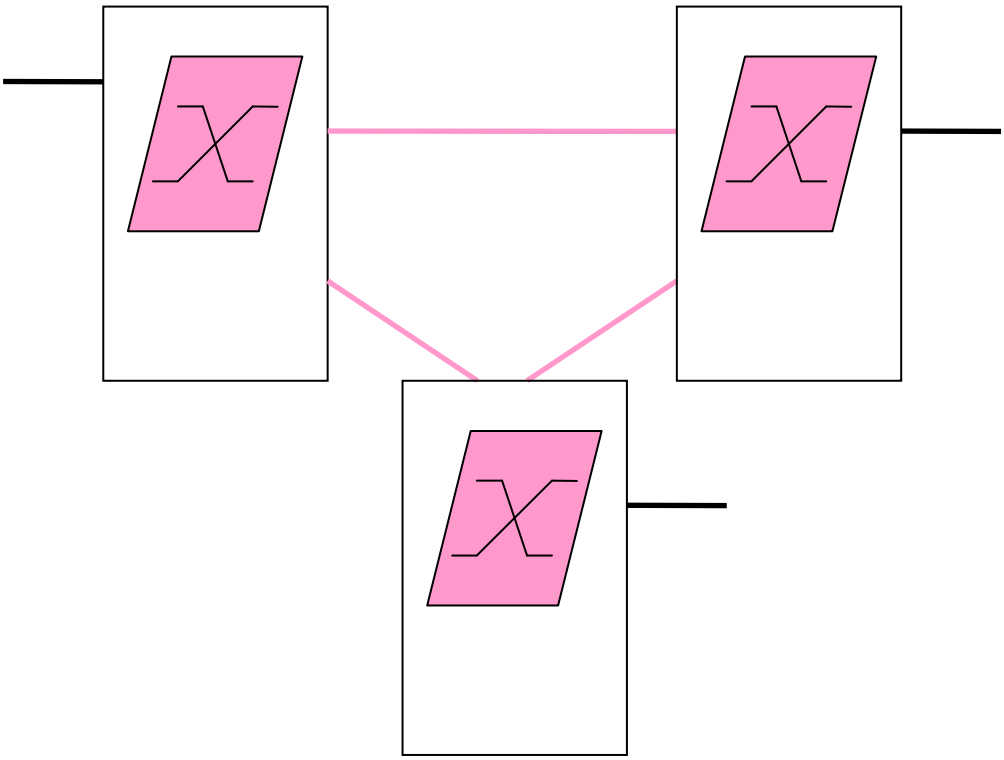
An alternative approach is to return to the simple solution of using more than one network – one, largely under-utilised, for real-time traffic and one that can be filled with non time-critical traffic – but having them co-exist in the same hardware.

Given that there already exist a large number of SpaceWire enabled devices using existing software and SpaceWire silicon we also desire to be able to use these existing developments without change.

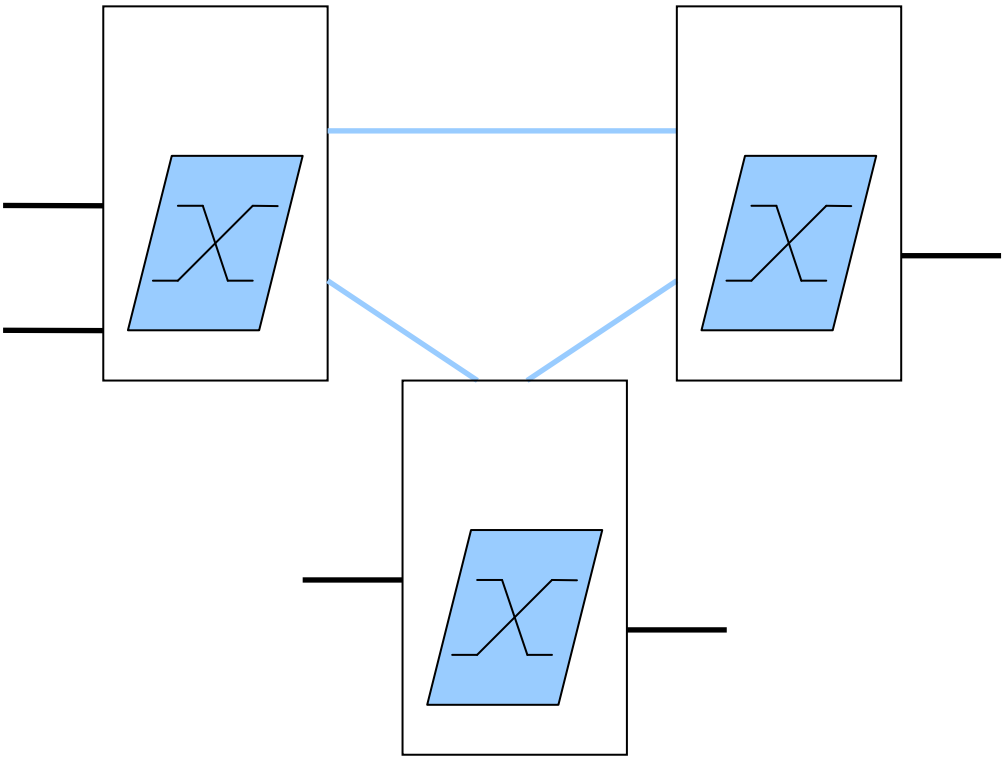
We suggest-that Virtual SpaceWire Networks meet the needs for a very low-latency Reserved service and for a high-utilization Best Effort service, while needing no change to existing node hardware or software, and needing minimal change to the existing SpaceWire standard. We will show also that Virtual SpaceWire Networks bring additional benefits to SpaceWire users for minimal cost and change.

Virtual Networks - the Concept

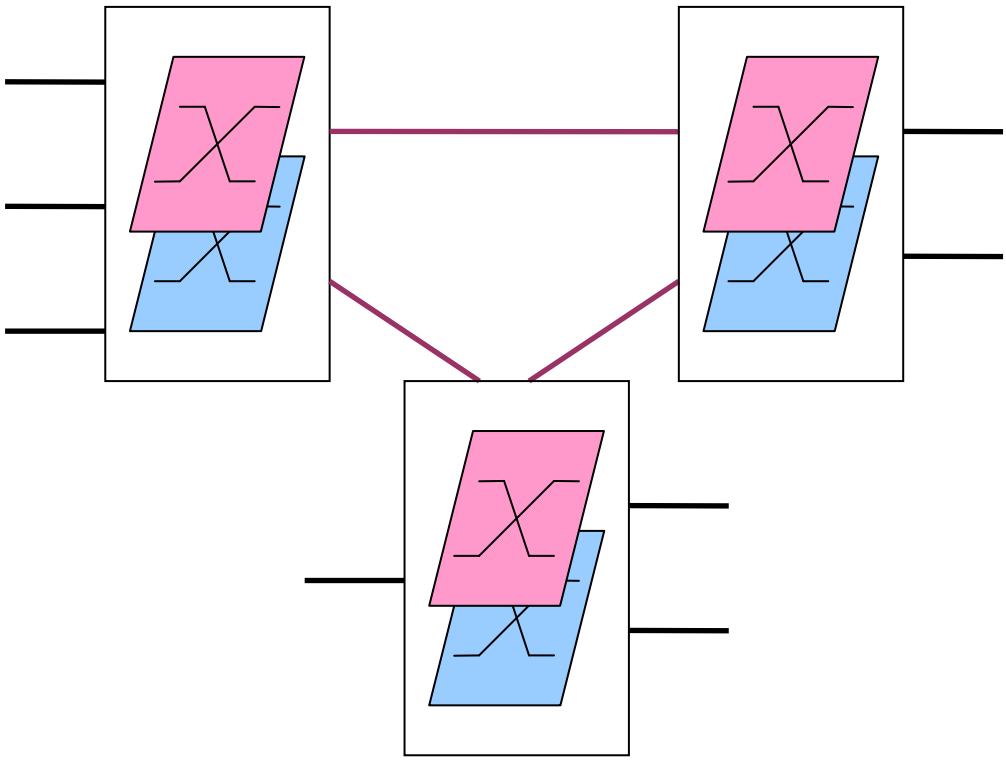
Conceptually, we want to provide a network for high priority traffic:



And another network for low-priority traffic:



Implemented as two *Virtual Networks* on one set of hardware:



The above diagram shows a network consisting of three physical routers where each contains (in this case) two virtual routers, one for each priority level. End nodes connect, by virtue of their location (router/port) or routing header, to one of the internal routing switches. All the upper switches (red) form one Virtual Network, all the lower switches (blue) form another, independent, network (there may be more than two Virtual Networks). Physical links between routers carry interleaved (on a byte-by-byte, not packet-by-packet, basis) traffic with priority given to bytes from the higher priority Virtual Networks.

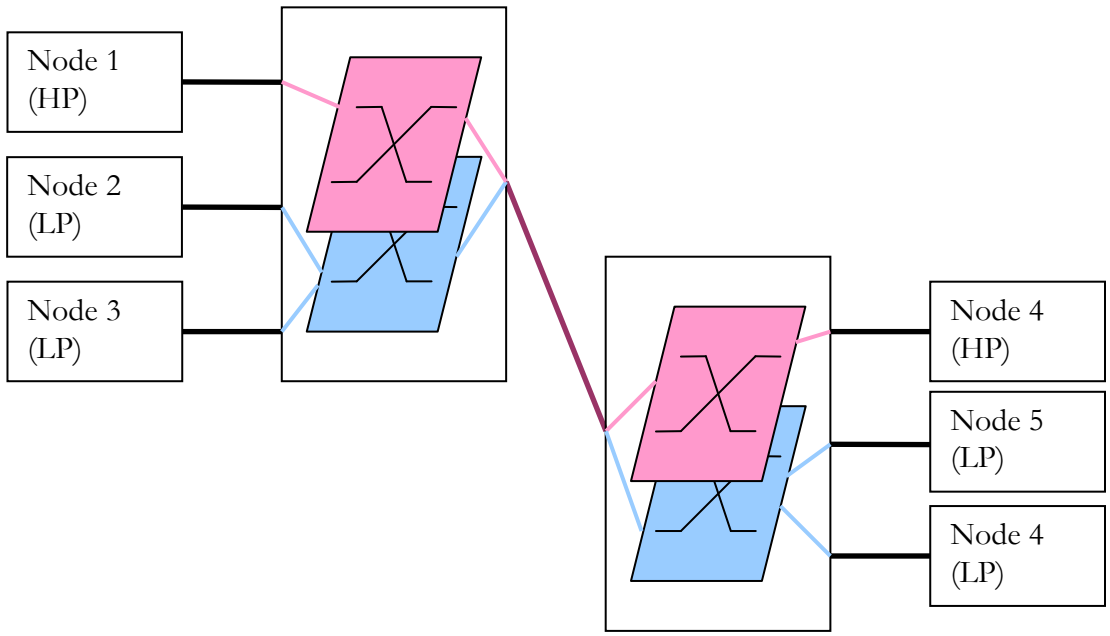
The Virtual Networks are prioritised so that real-time traffic can overtake lower priority traffic.

- Each priority is assigned to a Virtual Network and each Virtual Network has its own Flow Control Tokens. A new packet on a Virtual Network can only use a link between routing switches when the current packet on that Virtual Network has completed. However, at any time, a link can carry (interleave) packets from different Virtual Networks – up to one packet per Virtual Network;
- Single-priority (existing) nodes transmit packet by packet, as in existing SpaceWire networks.

Existing SpaceWire routing switches, such as the Router_SpW_10X, can be used where priority / real-time is not required. This allows a heterogeneous network of "old" and new devices.

Traffic can be directed to a Virtual Network as a function of:

- the routing header (e.g. logical address 67 = high priority)
- the location of the node (e.g. router 1, port 3 = high priority)



Here we see two low-priority data nodes (2 and 3) competing for one low-priority Virtual Network – their packets will be sent one after another on the link between the routers. Data from the high-priority node (1), on a high priority Virtual Network, will be routed separately and its data will be sent as soon as they arrive – between bytes of the low-priority traffic – as though there were no other (low-priority) traffic on the network. The end nodes do not have to be changed in any way.

Performance

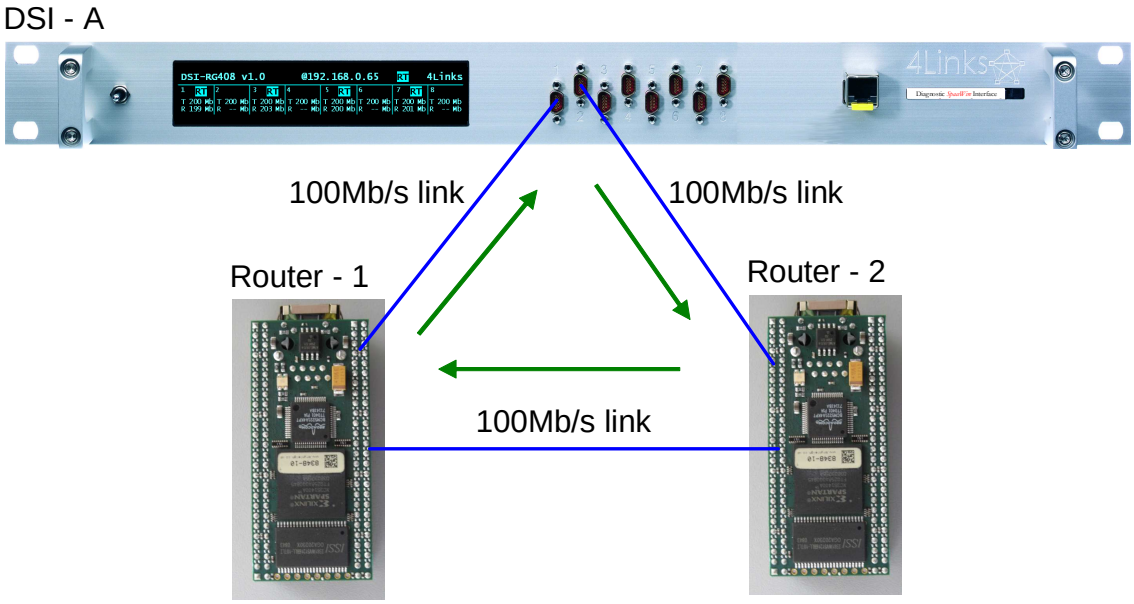
Some key aspects of Virtual Networks can be shown with the arrangement shown below. The network consists of two routers connected by a single 100Mb/s link. Data is injected into the network, and monitored, with 4Links Diagnostic SpaceWire Interfaces (DSI) which are able to accurately timetag data and hence measure latency. DSI-A is used to send packets to Router-1 and receive packets that have passed through the common link and Router-2. DSI-B performs the same function as DSI-A. The combination allows network latency and throughput to be measured. We present two situations – low-latency real-time and link-sharing.

Very low-latency real-time support

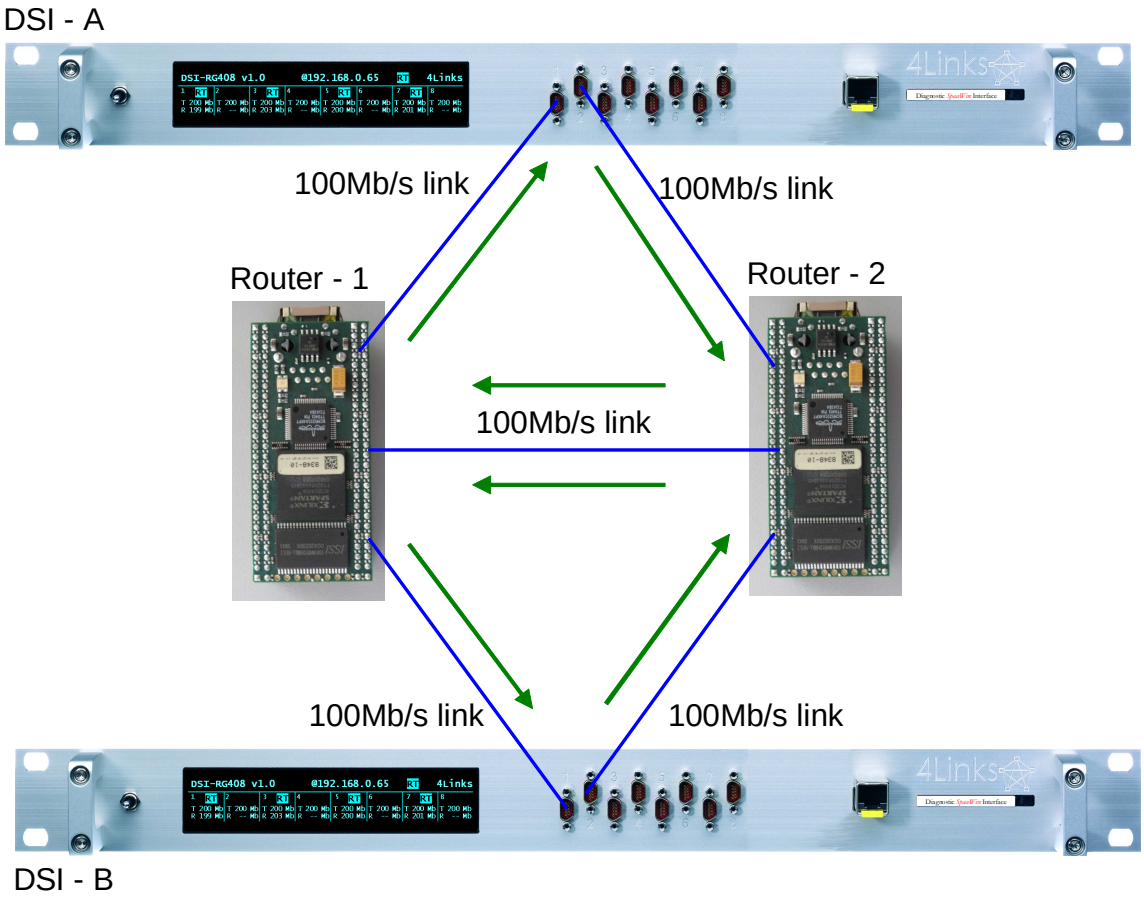
Best-case performance of a network with no other traffic is established by sending packets from DSI-A at a link speed of 100Mb/s and receiving them from Router-2 at 100Mb/s.

Short (5-byte) packets were used for this test although, thanks to wormhole routing, the figures would have been the same for longer packets. The latency through the two routers was found to average 1.169µs with jitter of <0.16µs*. This would provide satisfactory real-time performance for demanding applications.

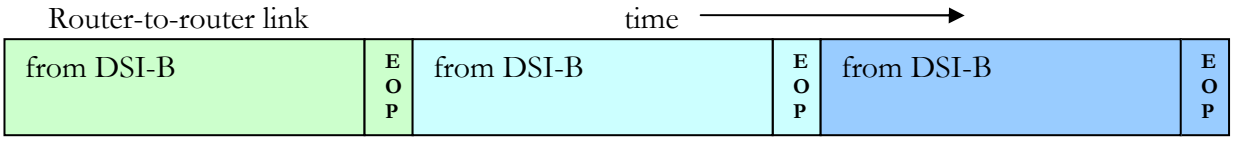
* Measured values: 1.103µs, 1.149µs, 1.142µs, 1.119µs, 1.161µs, 1.175µs. 1.239µs, 1.123µs, 1.162µs, 1.234µs, 1.255µs



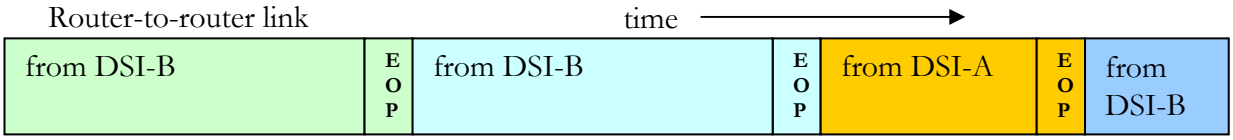
As SpaceWire is currently implemented, we achieve the worst case effect of the router-to-router link being fully occupied by allowing DSI-B to continuously generate 1000-byte packets at a link speed of 100Mb/s (80Mb/s data transfer).



Traffic on the router-to-router link consists of back-to-back packets from DSI-B.



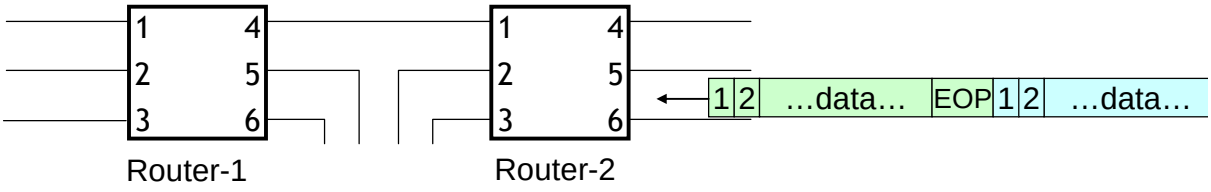
Repeating the above test and inserting packets from DSI-A requires Router-2 to interleave packets. This is done between packets from DSI-B.



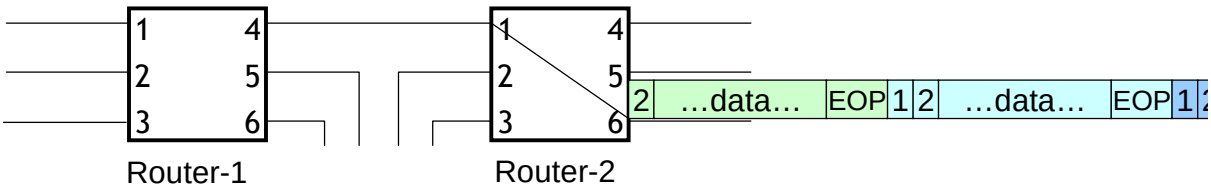
We find the delivery latency of packets from DSI-A to average 52.3µs with jitter of 65.6µs[†]. The delay is caused by the DSI-A packets being inserted between *packets* from DSI-B.

The packet from DSI_A can arrive at any time and must wait for its turn to be sent over a link. At best it can be sent immediately and at worst it must wait for the currently being transmitted packet (which itself may have just started to be sent) to finish. The additional average delay thus corresponds to one half of the time required to send a 1000-byte packet at 100Mb/s – 100µs. The expected jitter is equal to the 1000-byte transmission time of 100µs.

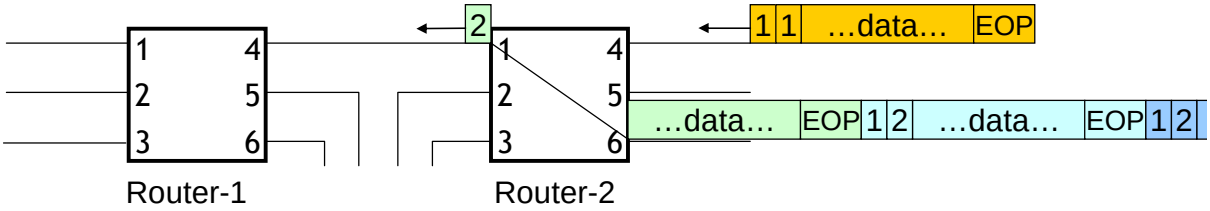
Let's look at that in more detail …Data from DSI-B (shown green) is sent to Router-2.



Its header value of 1 will set-up a path to link 1 on Router-2.

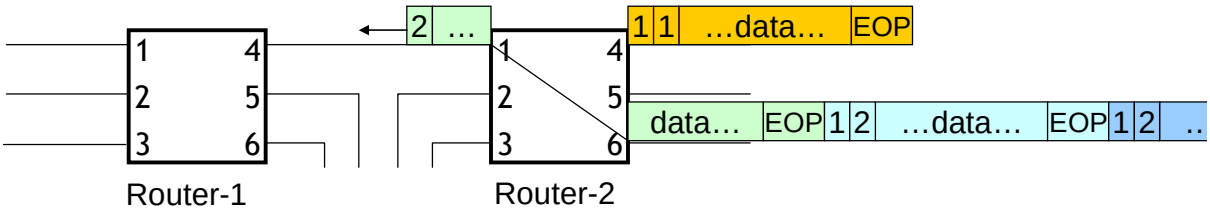


And the rest of the packet will start to flow through to link 1.
At the same time a packet is sent from DSI-A (shown yellow) to Router-2.

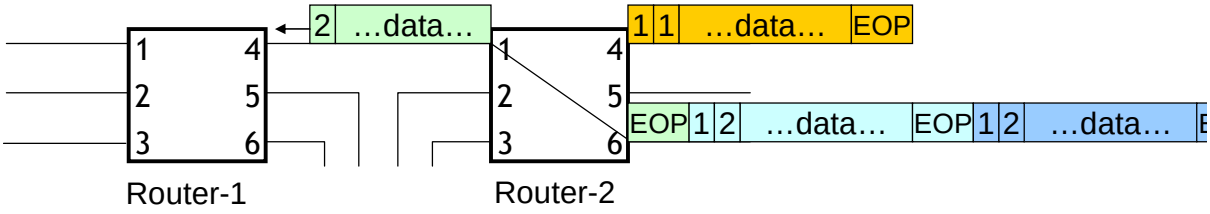


[†] Measured values: 55.383µs, 35.144µs, 41.017µs, 39.399µs, 86.077µs, 58.359µs, 20.523µs, 63.473µs, 65.809µs, 57.618µs

When DSI-A's packet (yellow) arrives at Router-2 it finds its path blocked and must wait…



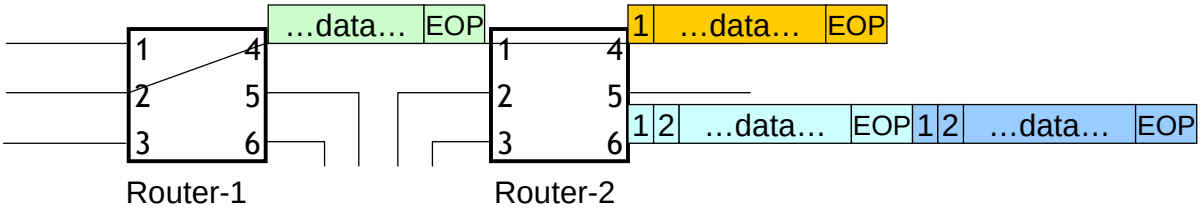
… for the rest of DSI-B's packet (green) …



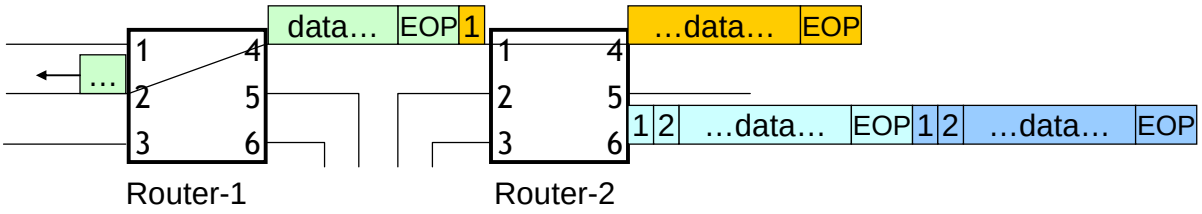
… to flow through Router-2.

When Router-2 has finished with DSI-B's packet (green) it looks at the header of DSI-A's packet (yellow) and sets up a path to link 1.

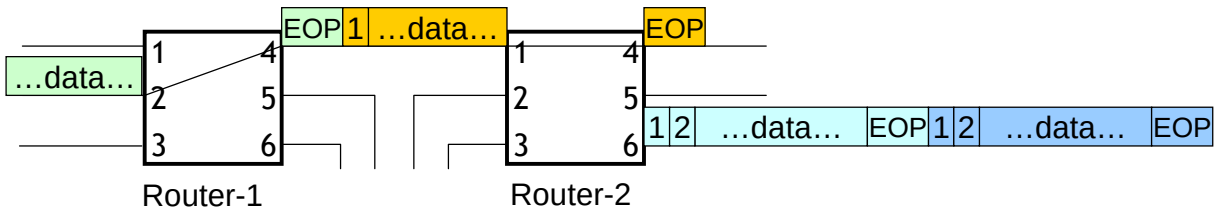
Meanwhile, DSI-B's packet (green) has arrived at Router-1, its header inspected and a path to link 2 set up.



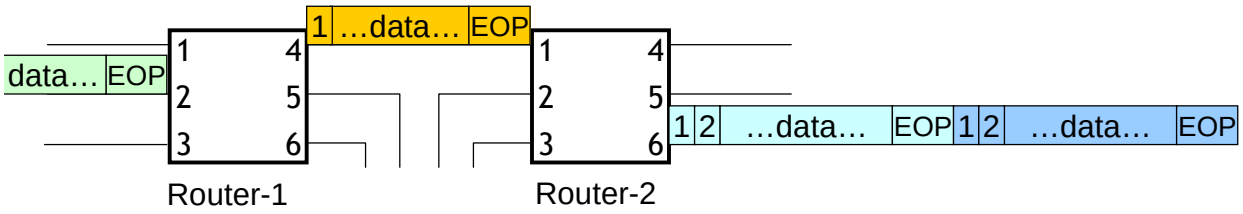
DSI-A's packet (yellow) can now follow DSI-B's packet (green) along the link between the routers.



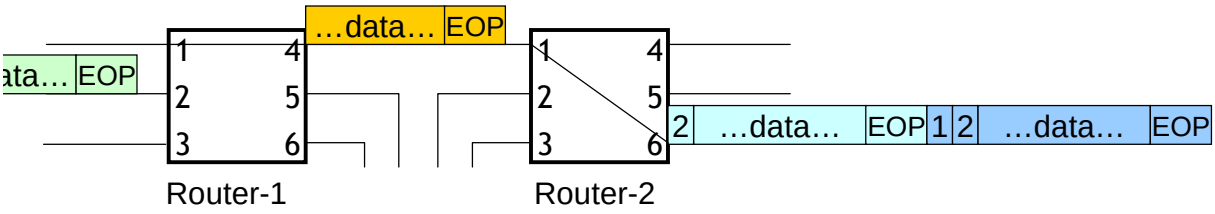
…



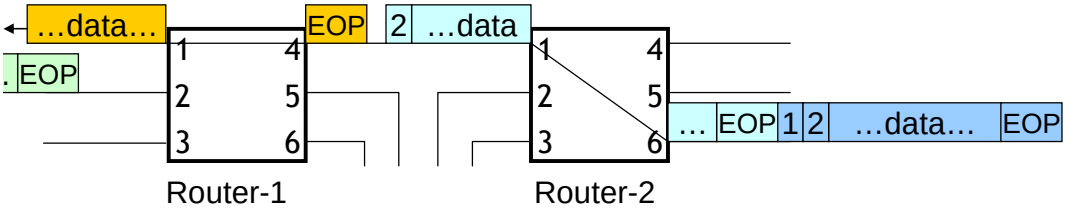
Until DSI-A's packet (yellow) is clear of Router-2 and, coincidentally, DSI-B's packet (green) is clear of Router-1.



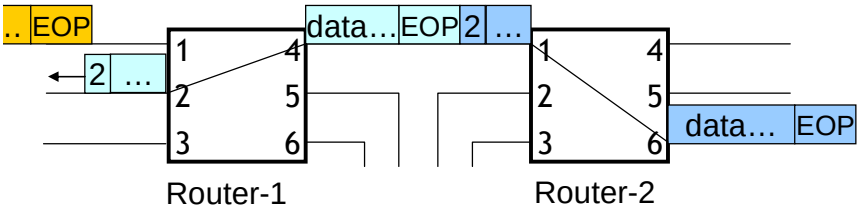
Now Router-2 can look at DSI-B's second packet (blue) and set up a path to link 1.
And Router-1 can look at DSI-A's packet (yellow) and set up a path to link 1.



The packets flow through the routers … including a third packet from DSI-B (darker blue) …



… until Router-1 has finished with DSI-A's packet (yellow) and routes DSI-B's second packet (blue).



Note that the link between the routers passed only complete packets. As a consequence, DSI-A's packet (yellow) – representing the high-priority / real-time packet had to wait for DSI-B's packet (green) to complete before it could continue – a time period depending on the length of the blue packet, possibly a long time in terms of its own urgency.

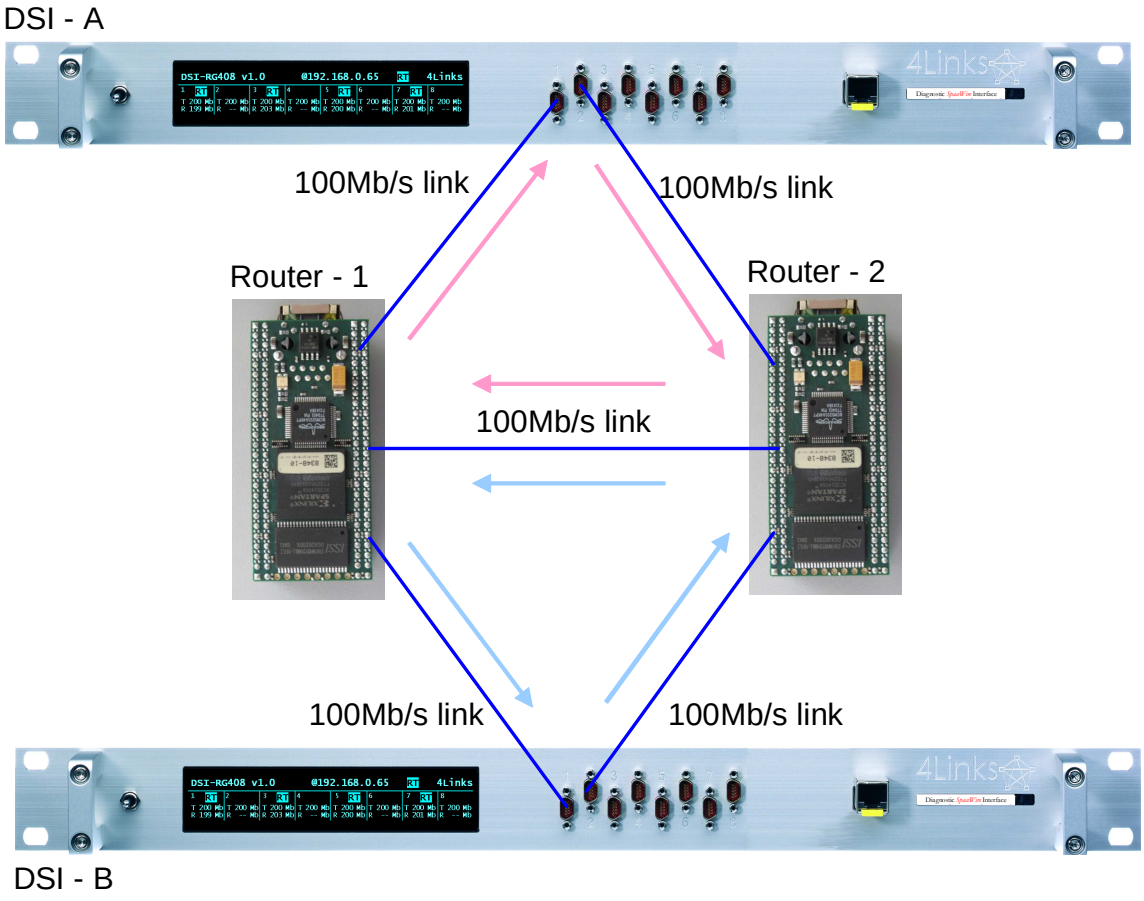
This delay would not occur – and the real-time packet could be sent rapidly onward, if it (the yellow packet) could be routed as soon as it arrived, without waiting for an existing (lower-priority) packet to complete.

This is what the Virtual Network accomplishes.

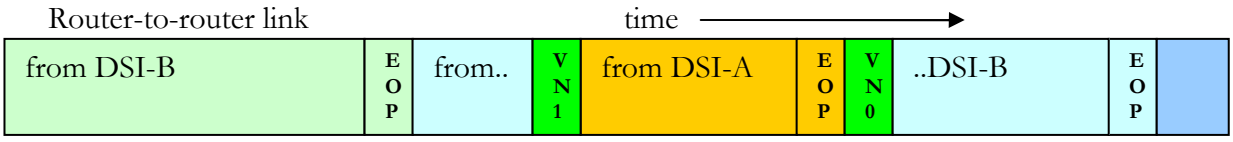
We can show this by making only a small change to the demonstration – changing the routing header on the real-time packet from DSI-A.

We now send DSI-A packets over the same router-to-router link but using a *different* Virtual Network *simply by changing the first (routing) byte of the packet* we get latency measurements averaging 1.45µs with jitter of 0.116µs‡. Apart from a small additional latency of <300ns we appear to have an empty network for these packets – but DSI-B still shows a data rate of 80Mb/s (fully occupied 100Mb/s link).

‡ Measured values: 1.425µs, 1.386µs, 1.454µs, 1.439µs, 1.459µs, 1.502µs, 1.457µs, 1.487µs, 1.469µs, 1.418µs

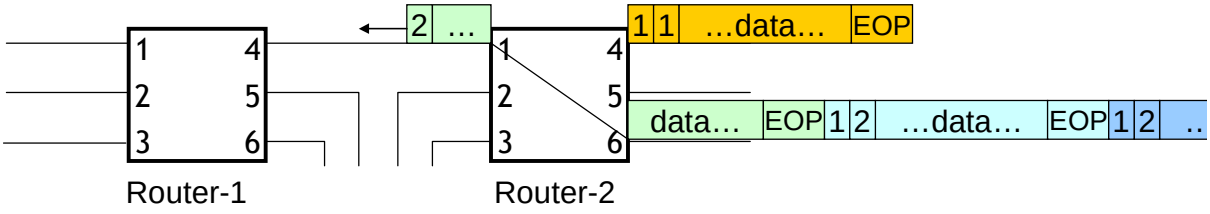


This results from data on Virtual Network 1 (data from DSI-A) being inserted between *bytes (within a packet)* of Virtual Network 0 by using redirect tokens VN1 and VN0.

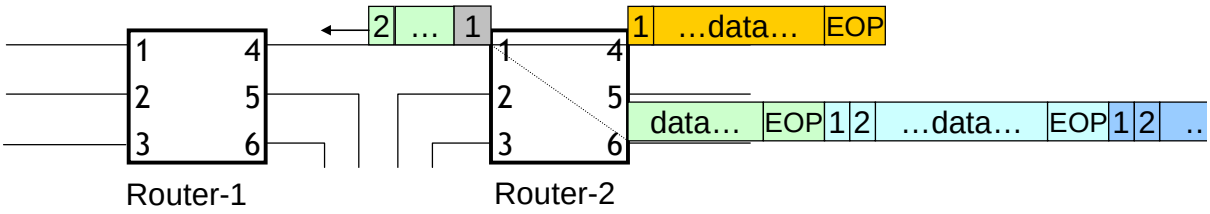


This can be seen in detail in the screenshot below showing a high-priority-Virtual-Network packet within a low-priority-Virtual-Network packet (time codes are used for VN0 and VN1, to indicate the change of Virtual Network – see later).

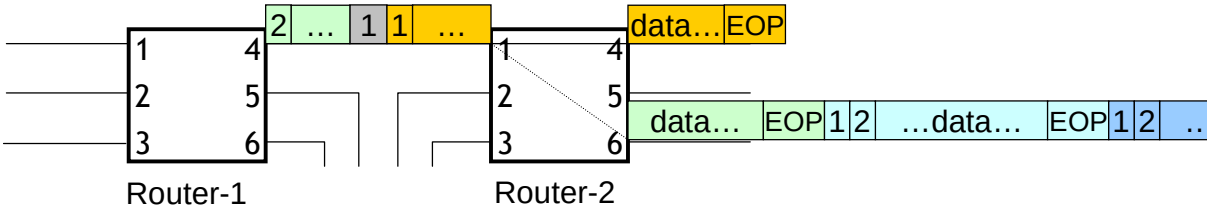
With Virtual Networks the behaviour changes when a higher-priority packet wants to use a link currently in use by a lower-priority packet. We pick up the story at the 4th diagram, when DSI-A's packet (yellow) arrives at Router-2 it and finds its path "blocked" …



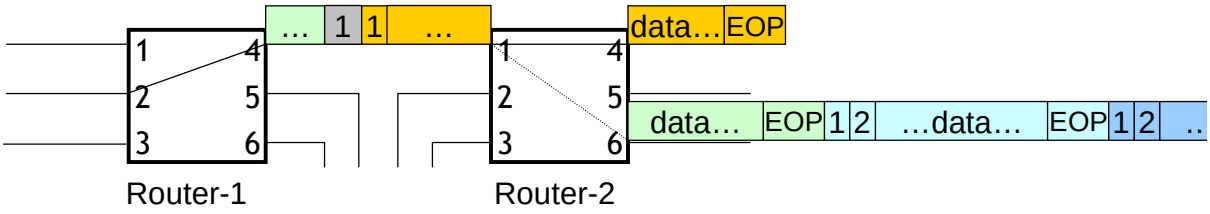
… it does not wait but, with a VN1 redirect token (grey), is transmitted immediately (the path within Router-2 is temporarily changed) …



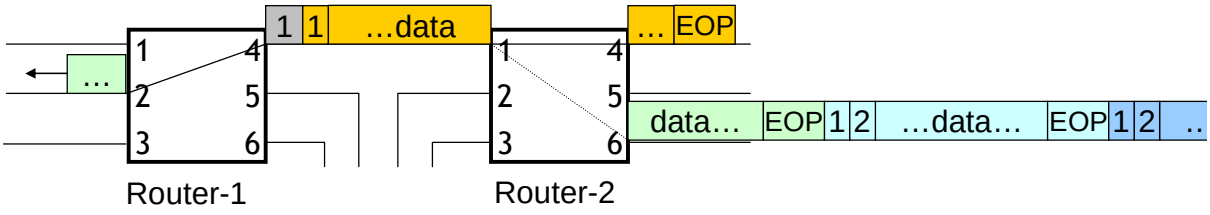
… to allow the higher-priority packet to flow through Router-2.



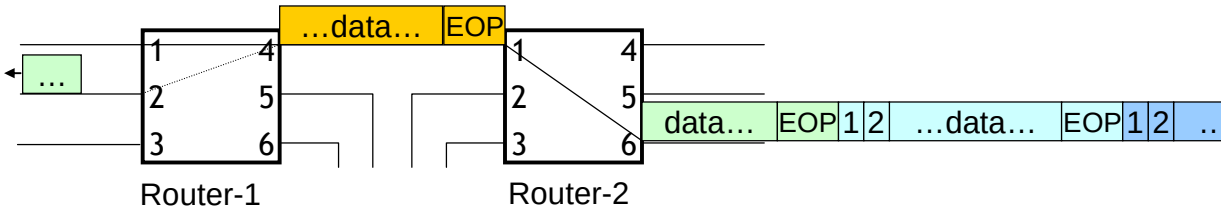
When DSI-B's packet (green) has arrived at Router-1 its header inspected and a path to link 2 set up.



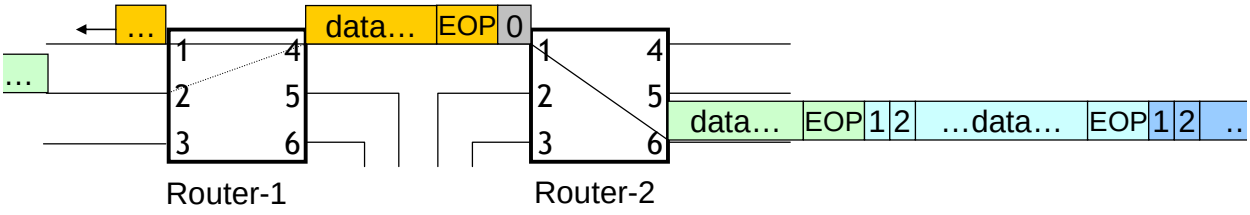
And DSI-B's packet (green) is sent to the required port.



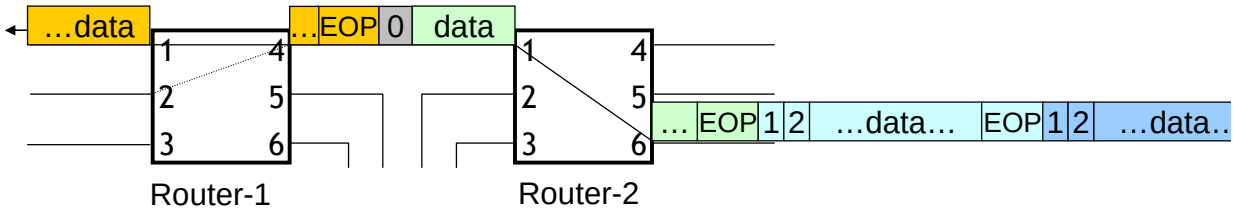
The arrival of a redirect token at Router-1 causes the route to be temporarily suspended and a new route selected using the header of the higher-priority packet (yellow).



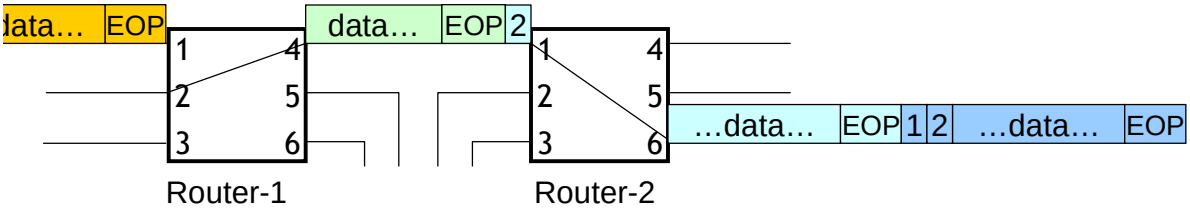
Router-1 passes the higher-priority packet to port 1 – without a redirect token because port 1 (in this example) is connected to a standard node.
Router-2, having passed the high-priority packet's EOP token, can switch back to sending DSI-B's lower-priority packet – inserting a VN0 redirect token.



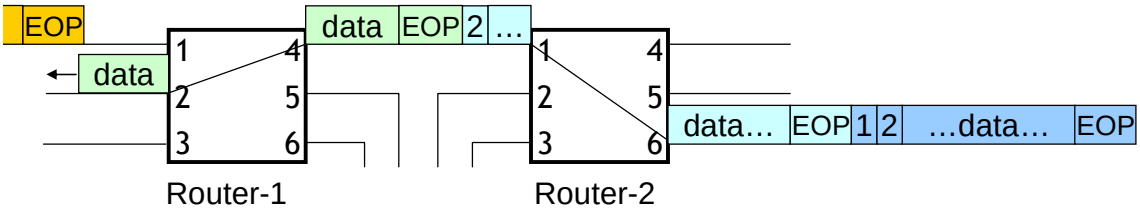
Router-2 now reverts to sending DSI-B's packet (green) to Router-1.



When the higher-priority packet has finished and the redirect token received, Router-1 restores the temporarily suspended path …



… and DSI-B's packet (green) resumes (after a pause) on port 2.



Thus we see that normal packets enter and leave the network – sent and received by normal SpaceWire nodes. Within the network, priority is used to ensure that time-critical data overtakes non-time-critical data.

Summary (for 100Mb/s links and 1000-byte low-priority packets):

	DSI-A data: Latency	DSI-A data: Jitter	DSI B data: Throughput
Empty network	1.2µs	0.2µs	0%
Busy network	52.0µs	100.0µs	~100%
Busy 2-priority Virtual Network	1.5µs	0.2µs	~100%

Higher priority networks can take all the bandwidth they need from lower-priority networks and can thus provide a reserved bandwidth service. Lower priority networks can instantly use any bandwidth not actually in use by higher priority networks (i.e. potentially much more than the difference between link capacity and reserved bandwidth).

Link sharing / multi-speed networks / babbling-idiot control

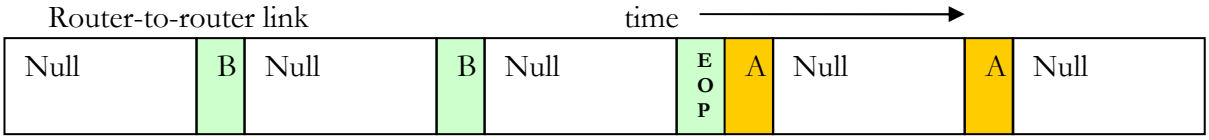
Interleaving on a byte basis prevents slow / stalled traffic on one network blocking traffic on other networks – whatever their priority. i.e. low-priority traffic can be inserted into the gaps between bytes of a slower but higher priority traffic stream

For example, using DSI-A to generate a continuous stream of packets at 10Mb/s to an empty network results in a data throughput of 8Mb/s. If DSI-B also generates a similar stream at 10Mb/s, sharing the common link we find:

	DSI-A throughput	DSI-B throughput
Single network	4.2Mb/s	4.2Mb/s
One Virtual Network for each stream	8.0Mb/s	8.0Mb/s

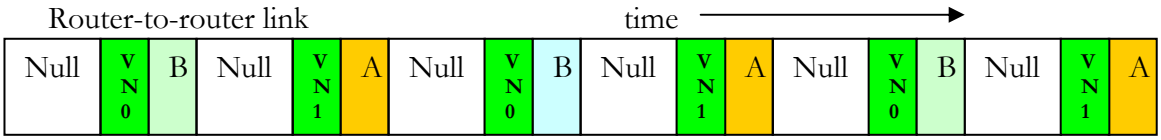
In a single network the *packets* are interleaved: when A's packet has the shared link B's packet must wait, and vice versa. In fact we see a slight improvement over simply sharing, and halving the throughput for each. This is caused by the CODEC buffers storing some data when it can't be sent and then sending it that buffered data in a burst when the link is first available (the shared link runs at 100Mb/s) – at 56-bytes the buffers store ~5% of a packet (of 1000-bytes) and the increase in throughput is correspondingly ~5% higher than might be expected.

Sending a slow data stream over a fast link in a single network leads to dead time when NULL's must be inserted and "wasting" bandwidth.



NULL insertion can be avoided if all data streams operate at the same bit-rate (and data can be supplied at that rate) – this is the currently recommended practice.

Virtual Networks, however, work independently with a link interleaving *bytes,* and can thus provide full throughput for each data stream. (Of course, more than two interleaved data streams can be concurrently supported – up to the available bandwidth of the link.)



This removes the down-side of mis-matched data rates and provides support for multi-speed networks.

One good reason to use multi-speed networks is that the transmit speed of a link sets an absolute limit on the amount of traffic that a "babbling-idiot" (an out of control node that continuously transmits) can inject into a network. The fact that this traffic cannot block all other traffic further limits the impact of misbehaving nodes. Mixed speed networks provide a useful error-containment mechanism. [Data rates can be further limited by restricting the rate at which a router issues flow control tokens to an end node.]

SpaceFibre

Higher speed (Gb/s) links for optical-fibre and copper bring an order of magnitude better performance than SpaceWire links. It is preferred that the networks built with SpaceFibre can be fully integrated with SpaceWire links and networks. This would necessarily be a mixed speed network and, as described above, is fully supported by the Virtual Network model.

Virtual Networks are thus able to unify SpaceFibre and SpaceWire into a single coherent network. Optical components and 8b10b coders/decoders have already been demonstrated but a suitable use of the available codes has yet to be agreed. The Virtual Network model requires only a one-to-one mapping of SpaceWire tokens to 8b10b codes – a simple process. The only, minor, change required is a change to the flit size (the amount of data controlled by a flow-control token) in order to support the higher data rate of SpaceFibre. This does increase the buffer sizes required – but to nowhere near the sizes needed for "frame" buffering in current SpaceFibre proposals.

Compatibility with Existing SpaceWire Products and Systems

Existing nodes and routers can be used for non-time critical networks with the option to replace just the routers for real-time support.

Network routers need to be configured and managed – but end nodes need no more than appropriate routing headers. Apart from the usual configuration parameters (e.g. the routing table content) it is necessary to map end nodes to virtual networks. This may be as simple as extended content in the routing table for header-controlled Virtual Network allocation or per-port configuration for a location-based allocation.

QoS / CCSDS / SOIS

Virtual Networks fully meet the requirements of CCSDS/SOIS.

Data delivery has the native characteristics of SpaceWire – very reliable delivery with extremely few link failures and very low data loss. This *Best Effort* service has proved to be sufficient for a vast range of existing applications, including the highly pervasive Internet.

Allocating priorities to Virtual Networks provides a *Reserved* QoS – the whole network bandwidth is available at the highest priority level (shared between traffic at this level). Lower priority networks have a calculable diminishing guaranteed bandwidth allocation (but may use any spare bandwidth not being used by other-priority traffic).

Assured and *Guaranteed* QoS levels can be provided in the applications (or CCSDS/SOIS interfaces). Since these will be required by only a subset of traffic, and the observed failure rate for SpaceWire links is extremely low, there does not appear to be any need to implement them in hardware. Hardware implementation of these services, if there is sufficient justification, is probably best done at end nodes, not in the network.

Prioritisation is directly supported by Virtual Networks, enabling mappings for different traffic classes, including time-critical data.

SOIS convergence functions will be required for provision of redundancy and, possibly, retries (although this is expected to be handled at application level).

Segmentation is not likely to be required since Virtual Networks can handle extremely large packets without their blocking other data flows.

Resource reservation is a matter of allocating data streams to appropriate Virtual Networks.

Implementation - the detail

It should be noted that the diagrams showing two completely separate routers within a chip represents the concept – an implementation can be simpler. A port can only transfer one token at a time and there is far less duplication of hardware than would at first appear because the same hardware can be re-used. Only some register and receive buffer space must be allocated per Virtual Network.

A SpaceWire link able to interleave bytes rather than packets is required for links between routing switches, links to end nodes do not need this enhancement.

Byte interleaving requires a flag token to indicate a change of Virtual Network – in our proof-of-concept demonstrator a "time code" token is used – although there are other possibilities (the advantage of time codes is that existing CODECs already handle them – it is only necessary that the order of data tokens and time codes is preserved). Each Virtual Network also needs its own flow-control tokens – another "time-code" can be used for this purpose. With 64 values available for each sub-set of time-code tokens, and two new tokens being required per Virtual Network, it would be possible to implement 32 Virtual Networks this way. If more Virtual Networks are required a different token could be used.

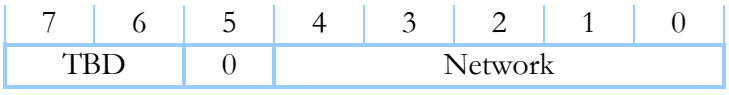
It is possible to use several Virtual Networks as a strict priority hierarchy or to have more than one network at each priority level. For example, it might be that only two priority levels are required but multiple networks at each priority level to allow bandwidth sharing without the possibility of blocking (see next paragraph).

Each Virtual Network has its own flow control tokens. Although one of the Virtual Networks may block due to in-transit (or stalled) packets, other networks (whether at the same or different priority levels) are unaffected.

The suggested time code usage requires one of the sets of values from the reserved set with the 6-bit field divided into two fields:

- 1-bit indicating whether this token is a flow control token or a redirect-data-to-network token (following data is sent to the indicated virtual network until another redirect token is received);
- 5-bits are used to indicate the virtual network (for up to 32 possible networks).

Redirect data to network token



Flow control token

7	6	5	4	3	2	1	0
TBD		1	Network				

The network bits may be further divided into a priority level and network within that priority level:

7	6	5	4	3	2	1	0
TBD		0/1	Priority/Network				

The actual division – from 2 to 32 networks and from 32 different priorities to 32 networks at the same priority needs to be determined by input from users, system designers, … – the SpaceWire community.

Example tokens, assuming the TBD bits are 1, 1:

- a Flow Control Token for Virtual Network 1 would be sent as ESC #E1
- direct data to Virtual Network 1 would sent as ESC #C1
- direct data to Virtual Network 0 would sent as ESC #C0

The flow control tokens control a flit of 8-bytes (for SpaceWire, more than 8 for higher speed links, such as SpaceFibre).

Network 0 uses the normal FCT token.

Links that support Virtual Networks must be distinguished from legacy nodes – so that the additional tokens described above are not sent to legacy nodes.
We let the link start as usual, exactly as defined by the current SpaceWire standard. All Virtual Network capability advertisement, and operation, takes place with the link in the "Run" state.

A node, when it starts a link, can send a sequence to advertise its support of Virtual Networks. A router (or end node) able to support Virtual Networks, when receiving such a sequence, can begin sending Virtual Network flow control tokens. The sequence used to advertise Virtual Network support should be ignored by legacy nodes. We therefore propose using two (or more) consecutive codes from the above set – time codes with the top two bits non-zero. This alone will not be ignored by all legacy devices, some do not decode the top two bits. True time codes are expected to have incrementing bottom 6-bits and thus we propose that the advertising tokens be an invalid (for time indication) time code sequence. One possible advertising sequence is therefore ESC #C1 ESC #C0.

At link start the Virtual Network number is initialised to zero. Receipt of Virtual Network redirect tokens changes the Virtual Network that is to receive all further data until another redirect token is received (which may but would not normally be expected to contain the same Virtual Network number as the preceding redirect token).

Time distribution

Time code distribution (top 2-bits = 00) is not affected in any way by Virtual Networks. Real time codes would be given transmission priority over Virtual Network control "time codes".

Plug-and-Play

Plug and play is supported via port-0 requests and responses. Virtual Networks do not force any particular implementation which may be as simple as we (4Links) showed ~10 years ago or one of the more complex later proposals

User visibility of network status

We expect this to be provided via (router) port 0 requests – possibly as part of the Plug and Play provision.

Link errors

Link errors – bit corruption leading to a parity error / disconnect error (and other protocol errors such as exceeding the flow-control limit) would be handled exactly as at present. Errors in the additionally defined tokens for Virtual Networks would be subject to these parity-check and error detection mechanisms.

Any packet being transmitted when the error occurs is forwarded with an error-end-of-packet (EEP) in place of its normal end-of-packet (EOP) token. This may affect several packets (up to one per Virtual Network) at the time of the error. Upon restarting, the link is in a known state, at Virtual Network 0 with newly sent flow control credit for each Virtual Network.

What about Priority inversion?

Priority inversion occurs when a node is committed to completing a low-priority transaction and delays a high priority transaction – thus effectively giving precedence to the lower priority action. Virtual Networks do not suffer from this problem – but multi-priority end nodes could do so.

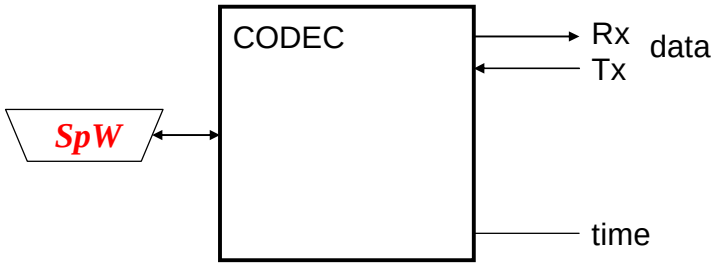
End nodes

Single priority nodes are exactly as at present. The router can allocate packets to priorities either by using the routing header and the routing table or by the physical location of the node (which router/port it is connected to).

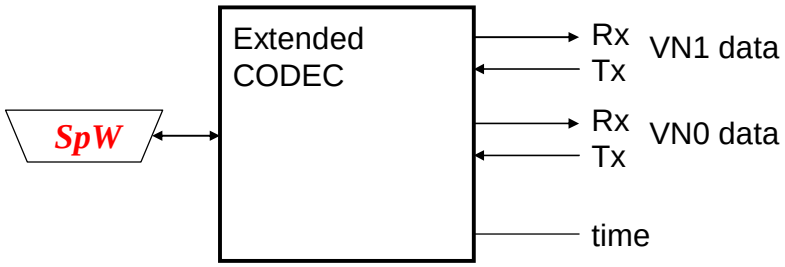
Multi-priority nodes can be built in two ways:

- provide a SpaceWire link for each priority
- use an extended CODEC to pass data with its priority – the Virtual Network concept can be extended into a node if that node can allocate a receive buffer for each VN and process VN redirect and flow-control tokens. The CODEC extension may not need to be any more complicated than a simple post-processor since the Virtual Network only uses existing SpaceWire tokens.

Existing CODEC's provide an interface between SpaceWire and a stream of data in each direction (and time ticks).



An extended CODEC (as implemented for the proof-of-concept demonstration) is similar, except that it provides a data stream for each Virtual Network.



Care must be taken that a multi-priority node does not allow interaction between its handling of data to/from different priority networks that could cause a priority inversion.

Standardisation

As a simple concept we expect the definition of Virtual Networks in the SpaceWire (and SpaceFibre) standard to be a relatively easy and fast process. We would suggest, however, that an evaluation period of use and/or study of realistic test cases be used to inform such standardisation.

Evaluation

We will make available small evaluation boards (Xilinx FPGA based) containing a router implementation in the very near future. We invite their use in representative systems to provide feedback to determine the optimum number of priorities, buffer sizes etc.

Conclusions

- Virtual Networks offer Real-Time performance whilst allowing complete re-use of existing SpaceWire software and interfaces. The existing, simple, model of SpaceWire networks is retained. New nodes handling multiple priorities can be implemented with an extended CODEC.
- The Virtual Network model unifies SpaceWire and SpaceFibre – allowing the latter to be fast-tracked to implementation.

Availability

Proof-of-concept devices exist and can be seen working. Evaluation devices, commercial and flight silicon will be available within a short timescale.

Acknowledgements

We give thanks to the several people who provided feedback on an early draft of this document – it has improved (and grown!) considerably as a result.

4Links Limited, registered in England, Company number 3938960
www.4Links.co.uk
info@4Links.co.uk
Suite E U 2, Bletchley Park, Sherwood Drive
Milton Keynes, MK3 6EB, United Kingdom
T +44 1908 642001, F +44 1908 363463