IAC-06- B5.7.1

ETHERNET OVER SPACEWIRE – HARDWARE ISSUES

Dr Barry M Cook
4Links Limited, UK
Barry@4Links.co.uk
Paul Walker
4Links Limited, UK
Paul@4Links.co.uk

ABSTRACT

We consider the opportunities for combining the best of SpaceWire, such as modularity, high speed, low latency, fault-tolerance, and ease of implementation, with the vast experience of protocol design that has been implemented on Ethernet. We consider how existing Ethernet-based designs can be implemented on SpaceWire networks.
Both technologies can be used to create networks that route packets from source to destination. SpaceWire, however, has a physical layer that has proven easier to build into a Radiation-Hard environment. Ethernet is based on the legacy of a bus and so relies on broadcast with packets visible to all nodes, whereas SpaceWire is entirely point-to-point and allows multiple connections for redundancy – which raises issues for broadcast.
Issues that will be addressed to allow Ethernet to work over SpaceWire include: converting the Ethernet addressing into SpaceWire routing; behavior in the event of faults (which may create a need to re-route); handling broadcast; and considering whether the network topology is static (as in conventional large spacecraft) or dynamic with plug and play (as suggested for responsive space on small satellites, or for the Shuttle/CEV). Further benefits will be described, for example the use of Ethernet protocols designed specifically for unreliable networks may significantly reduce cost by reducing or removing the need for upset mitigation techniques such as triple-modular-redundancy (TMR).

FULL TEXT

INTRODUCTION

Ethernet is a long established technology which has progressed through several generations and implementations. It is the basis for a very significant proportion of data transfers within 'local' area networks – where local can easily encompass thousands of users over campus- or small-town-sized facilities. Take-up is extensive and it is probably the best-known networking technology in the world.

Success for Ethernet is, at least in part, due to the very wide range of protocols that are supported, and have been developed by this ubiquitous data transfer technology. There is a protocol for high-speed transfers, guaranteed delivery, real-time data such as streaming audio and video and dozens more. Protocol development continues as new applications make their demands known and the underlying implementation improves. Networks with data rates of 10Mb/s, 100Mb/s

Ethernet over SpaceWire - Hardware Issues
Presented at International Astronautical Congress, September 2006, and selected for publication in Acta Astronautica

and 1Gb/s are now commonly deployed with higher rates appearing [1].

SpaceWire is a relative newcomer. Its roots go back to a 1995 standard [2] but recent (relatively minor) changes resulted in SpaceWire being standardized by the European Space Agency in 2003 [3]. Although new, world take-up has been extensive in the Space industry with many missions committed to its use.

One reason for SpaceWire being adopted is its demonstrated ability to be used to build highly fault-tolerant networks and systems.

As yet, SpaceWire protocols are very thin on the ground and there is no legacy protocol code for anything like the range of applications that Ethernet offers.

Naturally, the question that has occurred to some is – What if SpaceWire networks could offer Ethernet services as well as supporting the new Space-related SpaceWire, and other, protocols? Such a combination would provide a rich set of protocols for a wide variety of applications – and allow re-use of existing, proven, software.

We have established that this is indeed possible by implementing a proof-of-concept SpaceWire network running native Ethernet protocols. It was achieved by writing a Linux network driver for a SpaceWire interface – no other change being required in the operating system.

This paper compares Ethernet and SpaceWire to reveal why SpaceWire is attractive and discusses the, few, issues that have to be considered in implementing the network. These reveal that some work is usefully performed by software (in the device driver) and a companion paper [4] looks more closely at these software requirements.

A NOTE FOR THE PURISTS

The formal definition of Ethernet, as contained in IEEE802.3, uses precise terms for aspects of the standard. These terms are not always used by the world at large and other, not formally specified, words commonly substituted. SpaceWire does not share exactly the same terms of Ethernet's formal definition – but has much in common with Ethernet's informally used terminology. In order to avoid much tedious explanation, we have used the common set of terms in this paper and hereby apologize to the Ethernet community for our informality.

Layer 7 – Application
Layer 6 – Presentation
Layer 5 – Session
Layer 4 – Transport
Layer 3 – Network
Layer 2 – Data link
Layer 1 – Physical

Fig. 1: ISO OSI 7-Layer Model

THE CONTEXT FOR COMPARISON

In order to give a framework for comparing Ethernet and SpaceWire we will use the International Standards Organization (ISO) Open Systems Interconnect (OSI) "7-layer model" – fig 1. This model provides a simplified and idealized sub-division of the component parts in a communication system. It is a layered model with low-levels representing hardware implementation and higher levels the software forming ever more abstract interfaces up to the user.

From the bottom to the top we define the layers (adapted from [5]):

1. Physical – the electrical, mechanical and functional control of data circuits that connect to physical media;
2. Data link – establishes communication from station to station across a single link;
3. Network – establishes communication from station to station across a network;
4. Transport – provides reliable end-to-end communication and flow control in the higher level software;
5. Session – provides mechanisms for establishing reliable communications between co-operating applications;
6. Presentation – provides mechanisms for dealing with data representations in applications;
7. Application – provides mechanisms to support end-user applications.

As we shall see, when comparing Ethernet and SpaceWire, the lower layers differ considerably in detail but not in intent. Suitable implementation of lower layers results in layers 4 and above able to be identical for either implementation.

Layers 1 to 3 can be hidden within a device driver leaving the rest of the network code, and all of the user application code, unchanged.

THE PHYSICAL LAYER

Ethernet twisted-pair communication uses 2- or 4-pairs and multi-level signals at a very limited range of data rates. Communication at 100Mb/s requires 3-level signals on one twisted-pair in each direction. Some signal processing is required to interpret the 3-level signals.

At 1Gb/s, Ethernet uses 5-level signals on 4-pairs in each direction. Not only is the signal processing required for 5-levels more complex than for 3-level signals but there are 4 such signals to extract and, in addition, each twisted-pair carries signals in both directions adding another complication. Considerable signal processing effort is required, greatly complicating the link design and consuming significant power.

For most terrestrial applications it is reasonable to offer only 10, 100 and 1000Mb/s – a network change has to be worth the effort and an order of magnitude improvement justifies the change. For Space applications needing, say, a little more than 100Mb/s the power and complexity increase forced by a move to 1Gb/s is unattractive.

SpaceWire communication is pure digital, using Low Voltage Differential Signaling (LVDS). These two-level binary signals are recovered by a simple line-receiver with no signal processing requirement. The line code is a Gray-code which allows bit-boundaries to be recovered from the data stream – the transmit clock is recovered at the receiver.

As a result a very wide range of speeds is automatically supported by SpaceWire (currently from <2Mb/s to over 600Mb/s). Bit-to-bit speed change capability is a useful by-product that allows spread-spectrum clocking to reduce electromagnetic interference and also allows reduced power consumption by throttling back the transmit speed when there is no active data to transmit.

THE DATA LINK LAYER

Both SpaceWire and Ethernet transfer packets of data over a physical medium and thus may be considered equivalent. Electrical and other low-level differences do, however, favor SpaceWire.

Underlying model and Flow-control

Ethernet maintains its legacy support of a single wire connecting devices as though on a bus where one transmitting node can be received by all receiving nodes on that wire. The model used is essentially 'fire-and-forget' where, once a node has control of the bus, it sends a packet regardless of a receiver's capacity to receive it. Switch based networks with only two end-points on a wire can be operated with a crude flow-control mechanism to avoid overflowing receive buffers.

SpaceWire was always a point-to-point link with routing switches and fully integrates a low-level flow-control mechanism. Receive buffer overflow is impossible with SpaceWire – data is sent only after receipt of flow-control credit.

Limits

Ethernet sends blocks of data known, properly, as 'frames' but informally as 'packets'. As SpaceWire has adopted the term 'packet' we shall use this term for both systems.

As a result of its origins as multiple connections on a shared wire, Ethernet places limits on the size of a packet. A minimum size is specified to allow detection of collisions between nodes trying to drive the wire at the same time. A maximum packet size is also specified, for reasons that are not clearly spelled out in the standard. Gigabit Ethernet has an option for larger packets, limited by the error-detection capability of the cyclic redundancy check.

SpaceWire places no limit on the size of a packet and is thus able to accommodate any Ethernet packet.

THE NETWORK LAYER

Differences at this layer provide not only a very compelling argument for SpaceWire but also introduce the most serious differences needing resolution.

Topology and Redundancy

Ethernet is based on the model of a single wire connecting all nodes – although most implementations now use many wires and apparently decouple them, to a lesser or greater extent, with hubs and routers. A packet transmitted can be assumed to be visible to all nodes and the nodes select which packets to accept. In effect, all packets are broadcast. This model is accurately implemented with network hubs but optimized by routers which can learn node locations and filter packets – sending them only to appropriate nodes, when known.

Multicast is a natural characteristic of a single wire but not of a collection of routers. Routers that accept a multicast packet and transmit on all (other) ports can only be connected together with care. Any loops within a

network will result in a multicast packet circulating indefinitely and hence must be avoided. As a result there is a topology limitation in Ethernet networks: the network must be, mathematically, a tree and not a graph. This is unfortunate since the provision of redundancy through additional links and paths through a network will result in a graph structure.

SpaceWire networks, however, are intended to be built with redundant paths and loops in the network. As a result, they do not support multicast or broadcast – with the exception of time codes. This is our first significant issue as multicast/broadcast is a fundamental assumption of Ethernet and for some of its protocols. We shall address its resolution in the software issues companion paper.

Adjacent Ethernet routers may be connected by multiple links – 'link aggregation'. Additional hardware and an additional protocol are required to establish this connection. The benefit is a local increase in bandwidth and/or redundancy. There do not appear to be many Ethernet routers implementing aggregation.

SpaceWire can implement a similar technique with group-adaptive-routing (grouping). A set of ports may be deemed a group and packets directed to whichever link in a group is idle at the time of the request. Grouping also results in bandwidth increase and/or redundancy – but without any limitations on where the links in a group end up. There is very little hardware required for grouping and no negotiation.

Wide-scale redundancy in Ethernet can be provided by controlling and disabling links such that a tree structure results. There is a defined algorithm for this – the 'Spanning Tree' algorithm [6] – which, unfortunately, is not guaranteed to produce the best set of connections for maximum performance. Worse, the disabled links are completely cut-off, they cannot contribute to normal operation and they cannot be monitored. When a fault is discovered the algorithm has to be re-run on the whole network – and traffic stops while this takes place.

Grouping, by contrast, allows non-local redundant connections, keeps all available links in operation and can use all available resources. Other plug-and-play techniques can be used in addition to reconfigure on-the-fly so that only traffic needing re-routing is interrupted.

One way to see the difference is to view Ethernet's redundancy as passive whilst SpaceWire offers active redundancy.

Routers

SpaceWire routing switches and links require very little buffer memory as a result of the flow-control built in to the low-level protocol. Routers use 'wormhole' or 'cut-through' routing whereby a packet is directed to an output port as soon as that port is known – before the whole packet has been received. It is possible for the front of a packet to have traversed several routers and be arriving at its destination before the end of the same packet has left the source node. Low-latency data transfer is thus achieved.

Ethernet routing switches normally wait for the whole packet to be received into a buffer before forwarding it. The latency of a packet traversing several routers may be considerable.

Best Effort?

Ethernet makes no secret of the fact that it is only a 'best effort' delivery mechanism but SpaceWire, with its built-in flow-control mechanism, is often seen as a guaranteed delivery product. While SpaceWire may lose fewer data than Ethernet it is not perfect. It is always possible that a link, router or node may fail resulting in the loss of data. If we acknowledge less-than-perfection and use protocols to overcome the problem – such as those developed for Ethernet we may be able to simplify SpaceWire implementations. Rather than construct fully radiation tolerant components with additional hardware to protect register values, we might allow occasional radiation-induced errors resulting in loss of data because we have a higher-level protocol to correct the loss. Radiation hardness may be moved from components to protocols.

Routing

Both Ethernet and SpaceWire interpret leading bytes of a packet to control routing of that packet through switches. The precise detail is, however, different and will be discussed in later sections.

TIME CODES

One exception to the no-broadcast rule of SpaceWire networks is the specific case of time codes. These are single byte values that are intended to be sent at regular intervals from a time source. The values normally increment and only expected values are broadcast. Any time code that loops back to a

node that has seen it before is discarded and thus infinite loops are avoided.

THE TRANSPORT LAYER

Little needs to be said here, except to note that where SpaceWire has only a very small set of defined protocols, Ethernet has a large number of tried and tested protocols with software implementations available for use. Such SpaceWire protocols as do exist, however, are targeted to the Space industry.

ETHERNET PACKET STRUCTURE ABD ROUTING

Each Ethernet packet is defined to consist of

- A destination address – the address of the node to which the packet must be directed;
- A source address – where the packet came from;
- An indicator of either the length of the packet or of the protocol used;
- Data;
- A Cyclic Redundancy Check over the packet.

Only the destination address is important to us for the purpose of routing. A unique 48-bit value is used so that no two devices (in the whole world) have the same address. It is possible to attach any two or more devices to a network and be sure they do not have the same address.

One bit of the address is used to indicate whether the packet is intended to be received by a single destination (unicast) or by many destinations (multicast and broadcast). A unicast packet may be broadcast to all destinations – the decision as to whether to accept a packet rests with the receiver. It is not possible to rely on the routing network only to deliver packets addressed to a receiver.

The routing network starts by not knowing where any packet is to be directed. A packet with an unknown destination address is broadcast to all nodes. As nodes respond, their source addresses are noted by the network and routes to them become known. This knowledge soon results in a packet being sent only along the required path to its destination.

Dynamic networks, where nodes move between ports, are often not well supported as the mechanism for changing internal tables of locations can take too long to drop a route and re-discover the new location when a node moves.

SPACEWIRE PACKET STRUCTURE

Each SpaceWire packet is defined to consist of

- Data.

SpaceWire does not specify any structure on the data in a packet but leaves it to the node receiving the data to interpret it.

There are extensions to the standard that suggest some structure. One example is the 'Protocol Identification' [7] which corresponds to Ethernet's Type/Length field in indicating how the rest of the data in the packet should be interpreted.

SPACEWIRE ROUTING

There is no specified addressing in the packet. Instead, each switch interprets the first byte of the packet for routing purposes.

The first byte is interpreted in different ways, depending on its value

- 0: direct the packet to the switch
- 1-31: direct the packet to the physical port indicated
- 32-255: use the value as an index into a routing table that indicates what to do with the packet.

Address 0 is used to control / communicate with the switch, for example to set the routing table entries.

Addresses 1 to 31 constitute 'physical routing' and the packet is forwarded after deleting the first byte. This exposes the second byte of the original packet to the next router. A complete path through the network may be explicitly specified by the packet sender – 'source routing'.

Addresses 32 to 255 constitute 'logical routing' where a table in the switch is used to determine how the packet is to be forwarded. This allows the output port or group of ports to be specified and also determines whether the first byte is to be deleted or retained.

Logical routing retaining the first byte allows a packet to be directed through several switches to the destination with only a single addressing byte.

A mix of physical and logical routing may be used.

ETHERNET OVER SPACEWIRE

Two situations may be identified, routing a unicast packet to its destination and routing multicast / broadcast packets to all destinations. We will identify here the principles of what we need to achieve in the hardware and leave details such as precise

values to use to the companion software issues paper.

Unicast Packets

Each Ethernet packet to be routed will begin with a 48-bit destination address. Somewhere in the network will be a destination node with that address.

If we can satisfy the following constraints

- The network topology will not change;
- Each device is always connected to the same port on the network;
- The address of each device is known.

Then there is a simple mechanism to achieve routing.

1. Allocate a logical address to each device;
2. Statically configure each routing table to forward packets the correct device;
3. Translate each 48-bit destination address to its corresponding 8-bit logical address.

Run-time effort is restricted to address translation which need be little more than a table look-up (albeit a sparse table of 48-bit addresses).

A degree of fault tolerance can be achieved by using group adaptive routing over alternate paths – provided some care is taken over topology, see below.

It may be argued that minor changes, such as a link failure, can still be considered to be a static network in that a static configuration may not need to change. More significant changes, such as multiple link failures, routing switch failures, or powering-up a cold-redundant unit are very likely to be beyond a static configuration.

Changing topology or device location (including adding spare units) requires the ability to re-configure routing switches and, possibly, translation tables.

Several years ago, 4Links demonstrated a plug-and-play network where any topology could be used and devices connected anywhere, and the topology changed and the devices moved. The routing tables and address translations automatically changed to match the network. At least some of the techniques used there could be employed in this application. The companion paper describes the basic operation of that dynamic system.

Multicast / Broadcast Packets

We have already discussed the problems with multicast and broadcast and their implementation by multiple unicast messages. The impact on the system is that translation to a SpaceWire logical address must be extended to replicate packets to all known logical addresses.

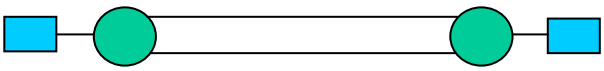

Fig. 2: Simple Fault Tolerant Network

SpaceWire switches are permitted to 'distribute' packets in a limited way and this may be used to reduce the number of packet copies transmitted.

LIMITATIONS OF STATIC CONFIGURATION

Static configuration relies solely on group adaptive routing for fault tolerance. This is a powerful technique as fault detection and fail-over to an alternate path is very fast – of the order of micro-seconds.

Simple networks, such that of Fig. 2 are well suited to this form of recovery. Devices are shown as rectangles and routing switches as circles. Lines represent links. If one of the two links between routers fails the other immediately takes all the traffic. Groups may include more than two links for greater resilience, greater bandwidth or both.

Fig. 3 shows a more complex network with more switches. Grouping is now less able to recover from faults. For a packet traveling from left to right, grouping allows a choice of routed from the first switch. If either link fails the packet is delivered via the other. If, however one of the links to the final switch, on the right, fails this will not be visible to the switch that needs to take an alternate path, the one on the left. Packets will still be routed to either of the middle switches and than may not be able to proceed. This can be remedied by ensuring an alternate path at each switch, as in Fig. 4.

Dynamic configuration would have been able to reconfigure around the disastrous faults in the network of Fig 3. Moreover, given the network of Fig 4, dynamic re-configuration could tolerate two faults.

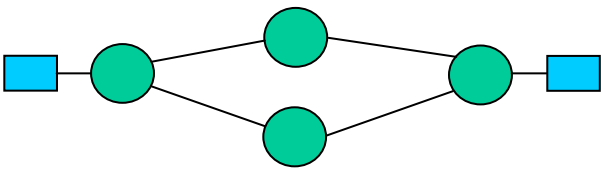

Fig. 3: More Complex Fault Tolerant Network

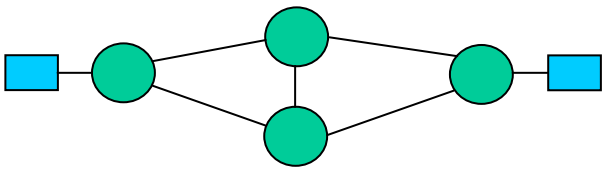

Fig. 4: More Complex Fault Tolerant Network with Increased Resilience

CONCLUSIONS

SpaceWire is easy to implement and very well suited to the construction of highly fault tolerant networks. Ethernet offers a rich set of tried-and-tested protocols – and software. Ethernet and SpaceWire deliver largely similar low-level services, except multicast and broadcast. Translation of Ethernet addressing to SpaceWire addressing can be achieved in more than one way and is the subject of a companion paper.

Ethernet over SpaceWire can be delivered but some software support is required.

REFERENCES

[1] IEEE Std 802.3 – 2002: IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

[2] IEEE Std 1355-1995: Heterogeneous InterConnect (HIC) (Low-Cost, Low-Latency Scalable Serial Interconnect for Parallel System Construction).

[3] ECSS-E-50-12A 24 January 2003 Space engineering: SpaceWire – Links, nodes, routers and networks.

[4] "Ethernet Over SpaceWire- Software Issues", B M Cook & P Walker, in Proceedings of 57th International Astronautical Congress, Valencia, 2006.

[5] Ethernet – The Definitive Guide, Charles E Spurgeon, O'Reilly.

[6] IEEE Std 802.1D – 2004: IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges.

[7] ECSS-E-50-12 Part 2 Draft B SpaceWire Protocol ID Jan 2005