AIAA Infotech@Aerospace 2007 Conference and Exhibit
7 - 10 May 2007, Rohnert Park, California

AIAA 2007-2930

SpaceWire Plug-and-Play: An Early Implementation and Lessons Learned

Barry M Cook and C Paul H Walker
4Links Limited, Bletchley Park, Milton Keynes, England
barry@4Links.co.uk, paul@4Links.co.uk

[Abstract] In the early days of SpaceWire, the authors built a demonstration SpaceWire network that could provide a level of fault-tolerance, through redundancy, unmatched by any other technology, yet with a performance some two orders of magnitude greater than previously offered. This was done with a Plug-and-Play system that could be assembled in an arbitrary fashion and continued to work by reconfiguring itself as faults were introduced and spare units brought on-line. The paper describes this demonstration network and the features we implemented to enable Plug-and-Play operation. We also evaluate the network's features, some of which appear to work very well and others suggest further work is required. Most importantly, perhaps, is that we have learned what issues (or at least some of them) have to be considered in the design of such systems.

I. Introduction

SPACEWIRE is a development from the IEEE 1355 standard that was originally intended for inter-processor communication in highly concurrent systems. The authors' experience stems from the design of IEEE 1355 systems and this has enabled them to be key players in the formulation of SpaceWire.

In the early days of SpaceWire it was necessary to convince people that time spent in its definition would be rewarded with a usable network. In order to support this missionary work the authors built a demonstration SpaceWire network. The principle requirement was to demonstrate that SpaceWire, with very simple hardware and protocols, could provide a level of fault tolerance, through redundancy, unmatched by any other technology - and with a performance some two orders of magnitude greater than previously offered. This was done with a Plug-and-Play system that could not only be assembled in an arbitrary fashion but continued to work by re-configuring itself as faults were introduced and spare units brought online. As time has progressed the emphasis has shifted from reliability (taken as proven) to flexibility in support of responsive programmes.

We will describe this demonstration network and the features we implemented to enable Plug-and-Play operation. We will also evaluate its features, some of which appear to work very well and others suggest further work is required. Most importantly, perhaps, is that we have learned what issues (at least some of them) have to be considered in the design of such systems.

II. SpaceWire

SpaceWire[1] is an unusually simple networking technology, which was derived from earlier work on the IEEE standard 1355[2], which was in turn derived from the inter-chip communications of an early system-on-chip microcontroller[3]. The designers' principles were based heavily on the principle of "If in doubt, leave it out", and so the technology has the minimum necessary and sufficient to support networking. The result is a very cleanly layered protocol stack.

In spite of its simplicity, SpaceWire removes many of the constraints imposed by other technologies, allowing almost complete topological flexibility and hence any required degree of redundancy for fault-tolerance. Well-designed SpaceWire networks should have the following qualities:

- They allow the network to grow by adding additional nodes, almost without limit
- They allow bandwidth in the network to grow, by adding cables
- They allow any topology, including loops
- They allow spare capacity, so that if a cable or node fails, another can take over
- They respond in microseconds to such failures
- And they are remarkably easy to use.

Our goal in producing a SpaceWire demonstration was to show to the developers and early users that SpaceWire delivered on the promise of the qualities described, and hence to give them confidence to invest in the technology.

III. Our SpaceWire demonstration

What became known as "The 4Links SpaceWire Demo" combined cameras, displays, routing switches and a computer, as shown, without the SpaceWire links, in Figure 1. The cameras provided high-bandwidth data, and we also wanted to show that SpaceWire could equally well be used for low-bandwidth controls, so we also included a dimmer switch and incandescent lamp.

We needed a means to manage the network, to direct traffic from one of the cameras to one of the displays, to demonstrate the fault-tolerance available from a redundant network. This was done with a Plug-and-Play system that, like SpaceWire itself, provided the minimum necessary that was sufficient to do what was required.

A. Mapping the network

The computer sent out simple probe packets to discover what devices were connected, and the computer collected the responses to create a topological map. The computer then displayed a map of the discovered network. Figure 2, and all the subsequent figures, is a screenshot from the computer. In this case it shows a map of the three routers, with no other devices yet connected. Like many other network protocols, we used Unique Identifiers to differentiate between the different devices, and the routing switches are no different in this respect from any other device in the network.

As we add devices to the network the new additions are updated in real-time on the computer screen. Figure 3 shows the addition of a camera and a display.

B. Managing the network

To connect a data source to its destination, we used conventional 'drag and drop' with the cursor on the computer screen. So, in Figure 4, we drag the cursor from Camera A to Display A, and immediately the images captured by the camera are seen on the display.

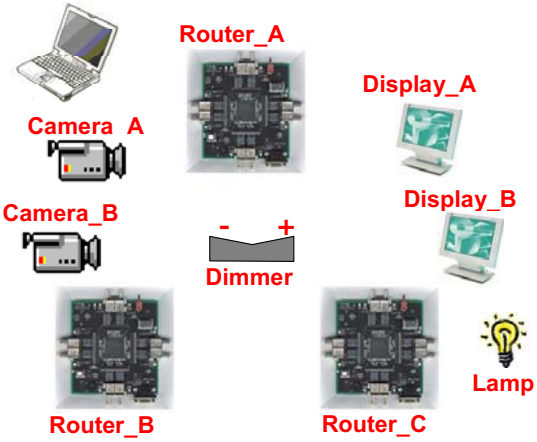

Figure 1. The initial hardware of the SpaceWire demonstration.

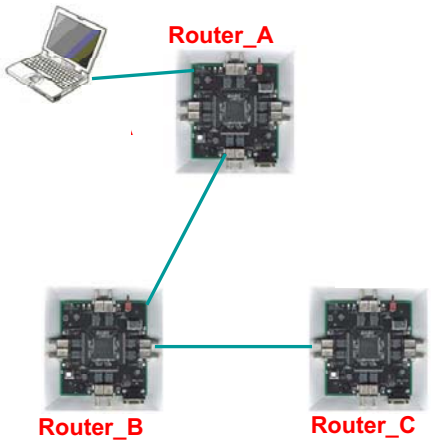

Figure 2. Network of computer and three routers shown on computer display.

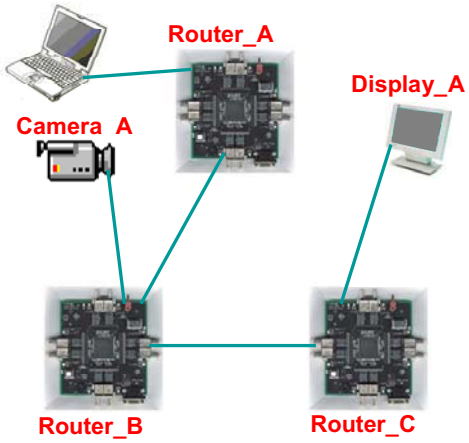

Figure 3. Computer screen showing Space-Wire Links added to camera and display

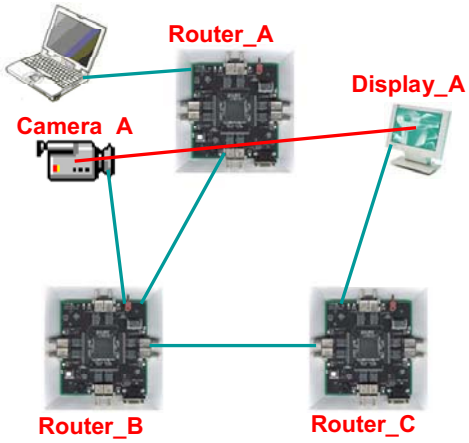

Figure 4. When cursor is dragged from camera to display, image appears on display

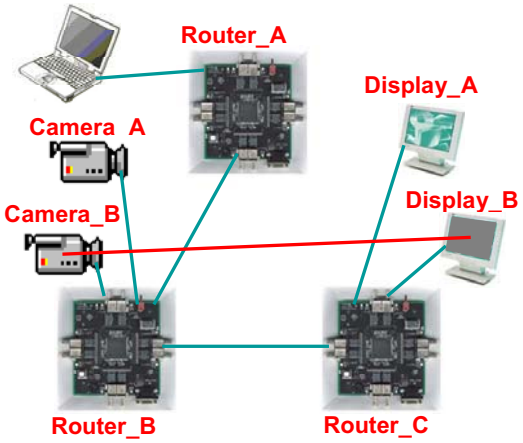

Figure 5. Adding another camera and display — not enough bandwidth

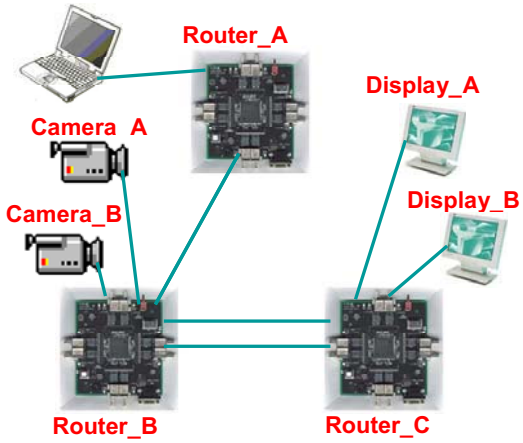

Figure 6. When another link is added, the second camera's image appears on the display

C. Coping with Insufficient Bandwidth

The demonstration was developed fairly early in the days of SpaceWire with low-cost commercial FPGAs. These nevertheless gave a respectable 200Mbits/s SpaceWire link speed. The camera images were 120Mbits/s which fitted comfortably within the bandwidth of a single SpaceWire link.

When, as in Figure 5, another camera and display were added, and the cursor dragged to make a connection between Camera B and Display B, the required bandwidth was now 240Mbits/s from an available 200. The network management therefore rejected the request for the connection between Camera B and Display B.

The request for the connection is remembered, however, and as soon as another SpaceWire link is added to provide enough bandwidth, as shown in Figure 6, Camera B's images are shown on Display B.

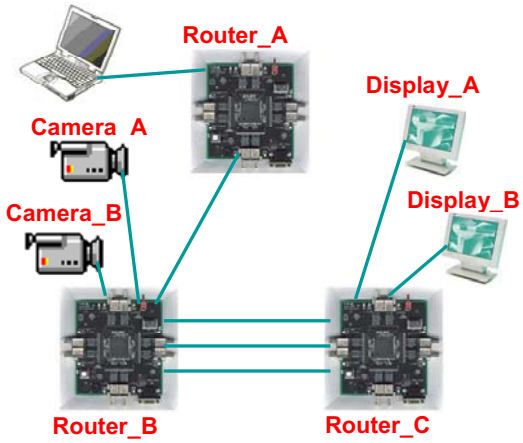

Figure 7. When a redundant link is added, traffic is shared between the redundant set of links.

D. Redundancy and Fault-Tolerance

When another SpaceWire link is added, as in Figure 7, the traffic is shared between the redundant set of three links. Any of these links can now be removed and there is still enough bandwidth and so the images reach the displays. The removal of a link may occur while a packet is being sent along that link, and so that packet would be corrupted, and indeed very occasionally a glitch was observed in the display that resulted from such corruption. The packet only held a single scan line of the image, however, and the next scan line was received, via one of the remaining SpaceWire links, correctly. As the interval between scan lines is of the order of 60 microseconds, the fault was recovered from within the 60 microseconds.

SpaceWire provides complete topological flexibility so that redundancy in SpaceWire networks can be achieved with loops, as shown in Figure 8 or with parallel links, as shown in Figures 6 and 7. Many conventional networks, including USB, FireWire and several implementations of switched Ethernet, would become entirely non-functional if connected

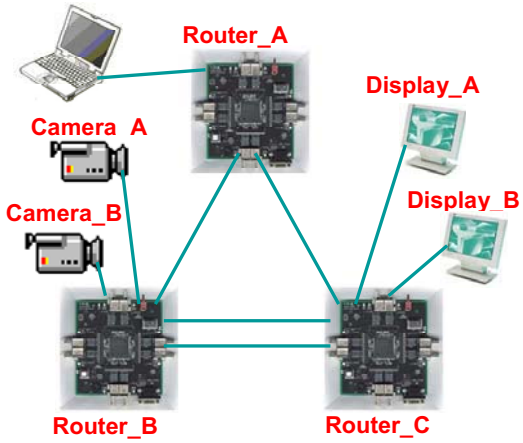

Figure 8. Redundancy in the network can include loops.

in the network of Figure 8, but fault-tolerance can only be achieved with redundancy and SpaceWire's freedom in this respect gives it great advantage. The flexibility also brings new opportunities for the Plug-and-Play system that may have been discarded for other network technologies that lack the topological freedom.

The fault-tolerance demonstrated resulted entirely from the properties of SpaceWire and our Plug and Play system. A reaction often received when people saw the demonstration was that they thought they had a choice between fault-tolerance and performance, and that we had shown them that it was possible to have both at the same time.

E. Plugging devices anywhere in the network

The topological flexibility of SpaceWire also makes it possible for the Plug and Play system to handle devices being plugged in anywhere in the network. Figure 9 shows Camera A reconnected to Router A instead of Router B. While the cable is unplugged, obviously the images do not reach the display, but as soon as the new connection is recognized by the Plug and Play system, the display shows the camera's images again.

This flexibility might appear to have no obvious benefit for satellites such as those used for Earth and Astronomical Observation (rather than for the Shuttle or Orion vehicles or for Operationally Responsive Space). But if a problem comes to light late in the integration and test, the opportunity to overcome that problem by a simple rearrangement of the topology could save both money and delay, even to the extent of rescuing the mission.

F. Redundant Routers and Links

In the examples shown above, we have had redundant links between routers but the routers themselves have been single points of failure. Taking two SpaceWire links from each device to two separate routers removes the single points of failure. Links can be removed to simulate failures as already discussed. Additional fault tolerance is available in that power can be removed from Router B or from Router C and paths remain for the traffic to bypass the router that has been disabled.

The computer and the router to which it is connected are not protected by redundant devices and paths, which is a problem that will be discussed below under Lessons Learned. In fact if the computer is disconnected, traffic will continue to flow as had been set up before the computer had failed. And if Router A is disconnected — and the paths have been set up to avoid it where possible because it is a single pojnt of failure, then again the traffic will continue to flow.

G. Simple devices and protocols

Neither SpaceWire nor our Plug and Play design are limited to high-performance devices. We included an incandescent lamp controlled by a dimmer switch. As soon as these were plugged in, anywhere in the network, the dimmer switch could be used to control the brightness of the bulb.

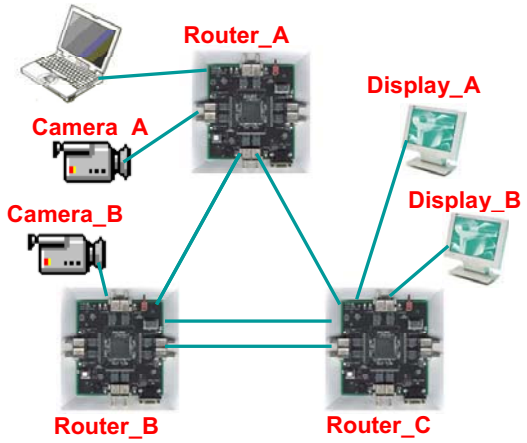

Figure 9. Devices can be moved around the network. In this case Camera A is moved from Router B to Router A, and the camera's images are displayed just as before

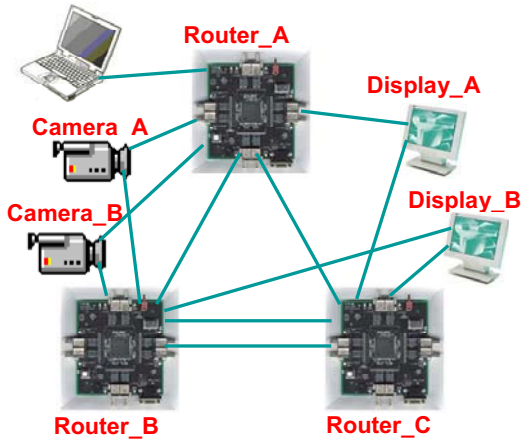

Figure 10.Additional SpaceWire links are added to provide further redundancy and fault-tolerance, allowing removal of power from Router B or Router C

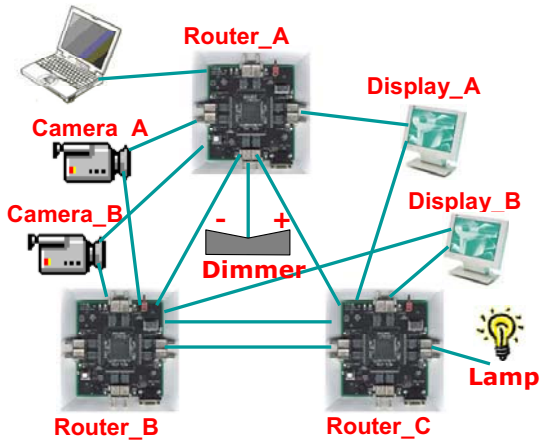

Figure 11. A dimmer switch and lamp are added as examples of simple devices that do not require a large overhead to be used in the SpaceWire Plug and Play network.

All the protocols used are extremely simple, following the principle of including only what was necessary but all that was sufficient for the communication. In the demonstration described so far, there are four protocols used, for network mapping, network management, cameras to displays and dimmer to lamp. Keeping the protocols so simple made them require very little logic. For simple devices like the dimmer and lamp, their being used in what would appear to be a high-performance network does not carry a cost penalty. Indeed having all the devices interfaced by SpaceWire and being capable of being plugged in anywhere in the network is likely to reduce costs significantly compared with having a multitude of network technologies and having to bridge both hardware and software between them.

H. Alarms

The network discovery process includes a simple status check on the devices in the network, and so can report an alarm when any change occurs that requires attention. An example would be removing the bulb from the lamp, which is reported in the device's status, and so generates the alarm shown in Figure 12.

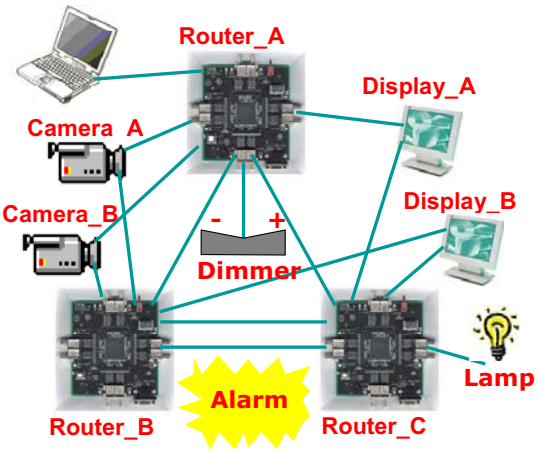

Figure 12. When the bulb is unplugged, an alarm is triggered.

IV. Lessons Learned

The universal reaction from people seeing the demonstration was that it was convincing, that SpaceWire delivers on its promise. The demo fulfilled the purpose entirely of persuading people to invest in SpaceWire, which is now widely used in the space industry world-wide.

What many fewer took away from the demonstration was that SpaceWire provides the benefits if it is combined with a Plug and Play system such as we showed. It was acknowledged that Fault Detection, Isolation and Recovery (FDIR) created major problems, and that Plug and Play would reduce costs in system integration and test, but the space industry was not ready for Plug and Play at the time. Fortunately, as the track in this conference is evidence, there is renewed appreciation of the potential benefits.

From our experience of designing and building the system, from discussions with the many people who saw the demonstration, and from our own reflection since building it, we have learned a number of lessons, which are described below.

A. Requirements

There is no point in building something if you have no idea what it is being built for, and there is a huge range of design choices available that depend on the requirements. We took a number of fairly obvious requirements, such as no single point of failure and scalability, and observed and heard others that may be less obvious.

A recurring issue (and one reason for the delay in adopting Plug and Play) is that satellites are often considered to be fixed configurations that have no obvious need of reconfiguration. On the other hand the Shuttle or Orion vehicles, the International Space Station, or a planetary space station or set of collaborating rovers, are dynamic — every mission is different and the configuration changes during the mission. A more recent requirement is from Operationally Responsive Space, where the need is to launch a satellite within days of its being specified, and a dynamic Plug and Play capability helps to make it possible to assemble a kit of pre-designed and fabricated modules in the knowledge that they will work together without any additional design or configuration.

Safety-Critical needs appear to be achievable, provided that the required behaviour can be specified, for example in terms of the allowable (or non-allowable) data loss or recovery-time. A consequential need for safety-critical systems is that their design is robust. Perhaps, however, the need for robust design is more generic: if the Plug and Play system is the basis of initial configuration and subsequent FDIR, the whole mission depends on it, so if robust design is required of any other part of the mission, it must surely be required of Plug and Play.

Quality of Service (QoS) must be considered. The usual implementation, which we adopted, is actually denial of service and this is not obviously correct for a space application.

B. Implementation

While the requirements that we know about are fundamentally important, any modular building technique, including Plug and Play, has another requirement. That is that it must be able to handle situations in the future that the designers have not considered. The normal approach to doing this is to add everything that can be thought of in the hope that everything is actually covered. The alternative approach that we have taken is to provide a simple framework that obviously does the minimum that is necessary, which is obviously sufficient for the present but which also provides for other things to be added later as optimizations or new capabilities.

There are significant implementation trade-offs between the complexity of the protocols and hardware/software support. Again, we have chosen the simple protocols — above all because we can see that these are correct and robust, whereas the complex protocols are difficult to grasp and their robustness must therefore be open to question.

C. The single point of failure

We demonstrated fault-tolerance for SpaceWire cables and routers, but the computer remained as a single point of failure. For the demonstration, this choice was probably correct, but it could be frustrating that there were some cables that we could not unplug and some devices we could not switch off. In a revised implementation, there must be redundancy in the control of the network. Of course this opens the door to complexity, but we believe it is best handled by simplifying the protocols still further so that they can be implemented in any device on the network, and in hardware as easily as in software.

D. Build it

Perhaps the strongest lesson we learned was from actually building a real SpaceWire system, with a real Plug and Play capability. Even with the rather small system that we built, it was possible to test out a large variety of network topologies and interactions, and we would regard building such a system as a sine-qua-non for any future Plug and Play work.

E. Build it again

The authors are in the process of rebuilding our demonstration as a phase-two proof of concept, retaining all the benefits of the original demonstration while testing solutions to the issues that we have discovered and learned. For this we are using our reconfigurable EtherSpaceLink family of SpaceWire test, simulation, and validation equipment, together with small and simple SpaceWire interfaces that have built-in Plug and Play and which make it easy to interface a wide variety of devices to Plug and Play SpaceWire. Our purpose this time is to provide what is needed for others to build their own Plug and Play SpaceWire networks, to validate the concepts and capability for themselves.

V. Conclusions

Our SpaceWire Plug and Play demonstration showed that SpaceWire could be used to build Plug and Play networks with arbitrary topology and arbitrary redundancy and hence any required degree of fault-tolerance. It was a highly successful proof of concept and was acclaimed as being in advance of what had been seen either inside or outside the space industry. Many obvious requirements were met, such as the fault-tolerance and scalability, and dynamically changing networks. The demonstration did not, however, meet all the requirements that we have learned, and there are implementation aspects that we would approach differently as a result of the experience. The authors are in the process of rebuilding the demonstration in a form in which Plug and Play developers and users can gain similar experience from building their own systems.

References

1. European Cooperation on Space Standardization, *ECSS-E-50-12A (24 January 2003) SpaceWire – Links, nodes, routers and networks*,
2. Institute of Electrical and Electronics Engineers, *IEEE Standard 1355, 1995, Heterogeneous Inter-Connect*, New York, 1995
3. INMOS Limited, *IMS T424 Transputer Reference Manual*, Bristol, England, 1985