A Link-Layer Broadcast Service for SpaceWire Networks

Allison Roberts, Sandra G. Dykes, Robert Klar, Christopher C. Mangels
Southwest Research Institute®
6220 Culebra Road
San Antonio TX 78238-5166
210.522.3248
{allison.roberts, sandra.dykes, robert.klar, christopher.mangels}@swri.org

Abstract—SpaceWire, a recent ESA standard, is gaining popularity because of its simple circuitry, low power consumption, and high-link speed. However network management on SpaceWire networks is hampered by the lack of a link-layer broadcast mechanism which is required for services such as the Address Resolution Protocol (ARP) and the Dynamic Host Configuration Protocol (DHCP). Currently, address resolution and host IP assignments require manual configuration. This paper describes a link-layer broadcast service and encapsulation service for SpaceWire that is implemented in the host node software drivers and requires no change to routers or host node interface hardware. We believe this work will help move SpaceWire towards a future of Plug And Play networks, decreasing the cost and time required to develop and integrate space systems.[1][2]

TABLE OF CONTENTS

1. AUTOMATIC CONFIGURATION OF SPACEWIRE

SpaceWire was introduced as a high-speed (10-400Mbps), low-power, and low-cost network for spacecraft. It is in use on several current and future space missions from NASA, the European Space Agency (ESA), the Japanese Space Agency (JAXA), and the Russian Space Agency. These missions include the Geostationary Environmental Satellite Program (GOES-R), the James Webb Space Telescope, and the Lunar Reconnaissance Orbiter [5].

While most current applications of SpaceWire support statically configured networks, it is easy to envision a future where it would prove advantageous to use SpaceWire in a dynamically configured environment. For example, it may be suitable for manned missions, such as NASA's Crew Exploration Vehicle (CEV). Introducing the human element introduces the potential for the addition or removal of devices from the network. Another possible scenario would be coordinated collaborative measurements between SpaceWire-based Science Instruments on multiple spacecraft as they move through orbits, creating dynamic, short-duration networks. SpaceWire has also been suggested for some terrestrial applications, such as streaming video. Users of such applications would undoubtedly want to make use of automatic network configuration support.

To achieve the goal of automatic network configuration, we offer a plan which leverages support from well established existing standards, including Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP) and introduces a dynamic routing protocol. None of these technologies are currently supported directly by SpaceWire networks as defined by the standard (ESA specification ECSS-E-50-12A) [4]. One element missing from SpaceWire that would help support such standard protocols is a link-layer broadcast. ARP utilizes broadcasts to discover the mapping between an IP (network-layer) address and a unique hardware (link-layer) address. DHCP clients use broadcast to discover the DHCP server that dynamically assigns a unique network address. Routing protocols for wired networks typically rely on router-to-router communication, rather than link-layer broadcast (although broadcasts are fundamental to ad hoc routing protocols). SpaceWire differs from most wired networks in that the host nodes must specify the packet destination address and, if regions are used, must also specify the address of the appropriate regional gateway. Broadcast could be used to disseminate the addresses of regional gateways to the host nodes. While this method does not address dynamic updating of the router forwarding tables, it does provide a simple "first step" that dynamically configures route information on host nodes.

This paper presents three central concepts. The first concept is that of a unique address for a SpaceWire host. The SpaceWire standard specifies that hosts may be assigned a logical address (LA) which is unique within a region. However network-wide uniqueness is not guaranteed since, a LA may be reused by another host in a different region. Regions are inferred from information in the router forwarding tables and do not have explicit

1———————————————
[1] 1-4244-0525-4/07/$20.00 ©2007 IEEE.
[2] IEEEAC paper #1484, Version 4, Updated January 15, 2007

identifiers. In our approach, each region is explicitly assigned a unique region identifier (RID). We combine the RID with the host node LA to form a host address that is unique across the entire network. A unique address is required to support a standard ARP protocol [9].

The second concept we introduce is that of a link-layer broadcast mechanism. An earlier version of our broadcast protocol was introduced at the 2006 Space Internetworking Workshop [3]. Implementing link-layer broadcast entirely in the host nodes allows the use of ARP, DHCP, and other configuration protocols without modification to existing Commercial-Off-the-Shelf (COTS) SpaceWire routers or host interface hardware. The broadcast algorithm conforms to the SpaceWire standard [4], adding only the concept of the region identifier.

The third contribution of this paper is a method to dynamically and automatically update route information on host nodes using the link-layer broadcast mechanism. Our method broadcasts *regional gateway update* messages to all hosts that contain the logical addresses of gateways between each region. To send a SpaceWire message, a host node need only add the logical gateway address(es) and the destination's logical address. Hosts do not need to know network topology or path information. This reduces the problem of route discovery to maintaining router forwarding tables.

Our approach to unique address formation, broadcast, and dissemination of routing information to hosts can be confined to software on the host nodes, and in particular to SpaceWire interface drivers. No changes are required to the SpaceWire specification, to SpaceWire routers, nor to host interface hardware.

While extending the breadth of SpaceWire network services, it is important to remain compatible with the established SpaceWire standard and to support existing hardware. This improves interoperability and encourages acceptance. The broadcast service and routing protocol described in this paper are designed to be entirely compatible with the SpaceWire standard. Although last year we presented a packet encapsulation service [2] to provide a mechanism for identifying packet application or stack association, we have chosen to instead implement the encapsulation service more recently introduced by the SpaceWire Working Group [6]. The improvements presented in this paper are designed to support multiple higher-level network stacks (IPv4, IPv6, SCPS-NP) as well as custom applications that directly interface to the driver.

2. SpaceWire Network Layer

SpaceWire packets are addressed using a series of bytes prepended to the data packet. An address in the range [1..31] is known as a "path address" and specifies which physical port the packet should travel through. An address in the range [32..255] is known as a "logical address." When a SpaceWire router receives a packet with a logical address, it consults the routing table to find the next packet destination. The logical addresses of 254 and 255 are reserved. For more detail on SpaceWire addressing, refer to [4].

SpaceWire routing is performed by configuration of routing tables stored on the routers. The standard method for a router table update is a manual configuration. Routers may be configured to delete the first byte of the address as the packet is forwarded, a technique known as header deletion. Header deletion is commonly applied to path addressing, to reveal the next physical port the message will be directed to.

Since SpaceWire uses a single byte for logical addresses, there are only 222 unique logical addresses available. To allow for larger networks, or to support the re-use of logical addresses, SpaceWire regions were introduced. Each region is guaranteed to have a non-repeating set of logical addresses.

Regional Gateways

When a packet needs to travel between SpaceWire regions, a special logical address known as a gateway is used. Nodes in different regions typically use different gateways to reach the same destination. This requires that each host node know the gateway as well as the logical address to each potential destination. Routers are configured to perform header deletion on gateway addresses.

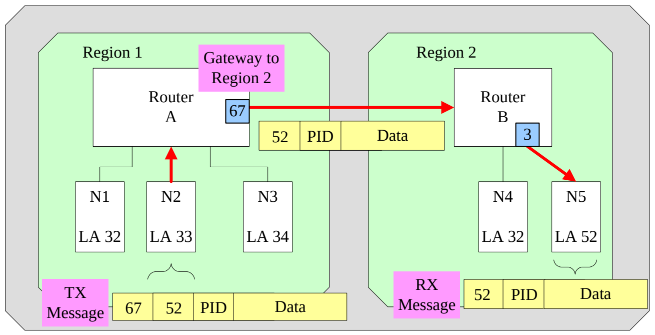


Figure 1. SpaceWire Gateway Addressing

Figure 1 shows an example of gateway addressing. To direct a packet from Region 1, Node 2 to Region 2, Node 5, the gateway address "67" is first used to move to the new region. When the packet is directed across the region boundary, the gateway address is stripped off, revealing the destination logical address of "52".

3. SpaceWire Messages

Packets are designed to require a minimal amount of overhead. The advantage of very small packet header size is a reduction in the operating cost of missions designed with custom packet handling algorithms. The packet header is only required to contain the destination (Figure 2).

The SpaceWire packet destination is specified as a series of bytes. Each byte may either specify the next hop in the network by designating the router output port (path addressing) or it may specify a reference destination that will be translated to an output port through a routing table (logical addressing). The series of bytes in the address is followed by the packet data.

Address	Data

Figure 2. SpaceWire Message Format

A recently proposed addition to the SpaceWire standard is the definition of a protocol identifier [6]. The protocol identifier (PID) indicates the network stack or application process on the destination node. The PID may be read from the second byte of the packet when it arrives at its destination. The PID values between 1 and 239 are reserved for assignment by the SpaceWire working group. Values from 240 to 255 are used for experimental protocols [7]. If the network makes use of the protocol identifier, the packet must retain a leading address byte. If the system is using path addressing where the routers will apply header deletion to the address bytes, the protocol identifier specification states that the first byte will be a dummy logical address of 254 (see Figure 3).

PA	PA	Dummy LA	PID	Data

Packet with Path Address (PA)

LA	PID	Data

Packet with Logical Address (LA)

Figure 3. SpaceWire Addressing with Protocol ID

4. SpaceWire Broadcast Protocol

Broadcast loops occur when a broadcast packet is sent back to its original destination and then sent out again, creating an infinite loop or broadcast storm. The SpaceWire standard sidesteps this problem by not specifying a general broadcast method. In our proposal, we will introduce a few simple implementation rules to prevent this condition.

The SpaceWire broadcast protocol we introduce here is built upon the SpaceWire standard. It may be implemented in software as part of the network driver on the host nodes. It uses the concept of a designated "broadcast server" to distribute messages to nodes on the local router or to the entire network.

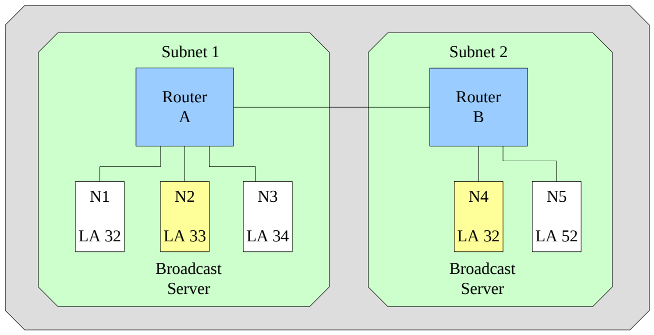


Figure 4. Broadcast Servers and SpaceWire Subnets

SpaceWire Subnet

We define the concept of a SpaceWire Subnet to indicate all nodes attached to the local router.

Broadcast Server

One host node in each SpaceWire subnet is designated as a broadcast server (Figure 4). This is envisioned as a secondary role of the host node; a broadcast server need not be a dedicated system.

To broadcast a message throughout the SpaceWire network, the host node sends a broadcast request to its broadcast server. The broadcast server then sends the message to all other broadcast servers with each one distributing the message to the nodes on their subnet (e.g., all nodes connected to the local router).

Figure 5 shows the format of a broadcast message. The proposed broadcast protocol identifier is 254. The PID is followed by the 1-byte broadcast protocol header and the message payload. The broadcast header is either a zero or one, identifying whether the server is to distribute the message to subnet nodes (*Type 0*) or to other broadcast servers (*Type 1*).

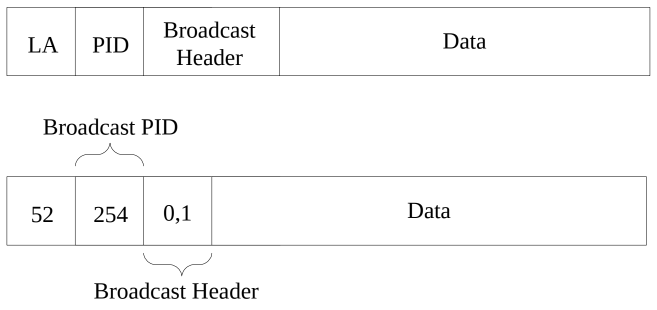


Figure 5. Broadcast Message Format

Messages from the stack

When the SpaceWire driver receives a broadcast message for transmission from a local application or from its network stack, it distributes the message as a Type 0 message (Figure 6) to all ports on the local router, which may include its own port. The SpaceWire driver recognizes broadcast messages from IP (Internet Protocol) or other network stacks because it contains a special network address to indicate broadcast. For example, IPv4 uses 192.168.255.255 to indicate a link-layer broadcast on the 192.168.0/16 network.

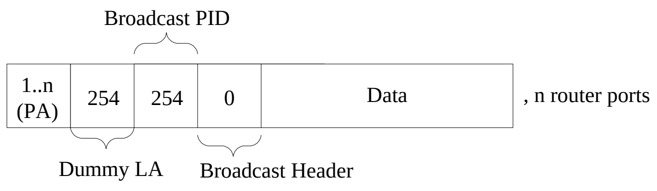


Figure 6. Type 0 Broadcast Message

Receipt of a Type 0 Message

When a broadcast server receives a Type 0 message from one of its host nodes, the server changes the broadcast header byte from a value of zero to one and distributes the Type 1 message (Figure 7) to all other broadcast servers. Figure 8 shows the broadcast server in Subnet 1 sending a Type 1 message to all other broadcast servers (in this example, there is only one other broadcast server to send to).

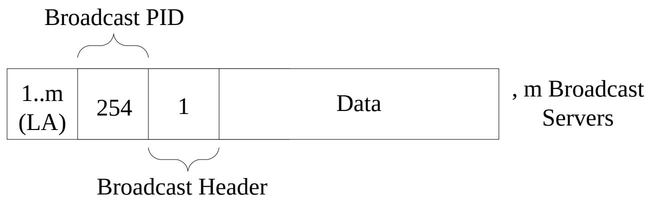


Figure 7. Type 1 Broadcast Message

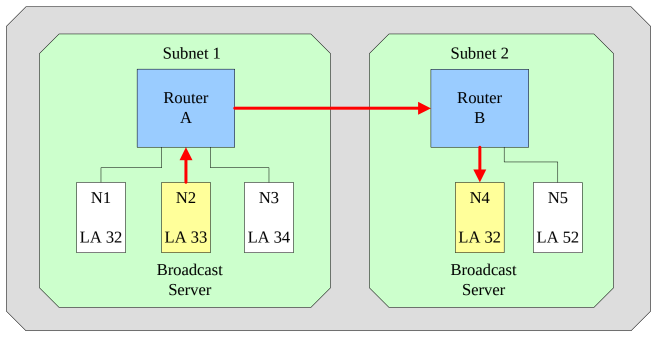


Figure 8. Server Broadcast, Type 1 Message

Receipt of a Type 1 Message

When a broadcast server receives a Type 1 message from another broadcast server, it changes the broadcast header byte from a value of one to zero and distributes the Type 0 message to all other ports on the local router. Figure 9 shows the broadcast server in Subnet 1 sending the Type 0 message.

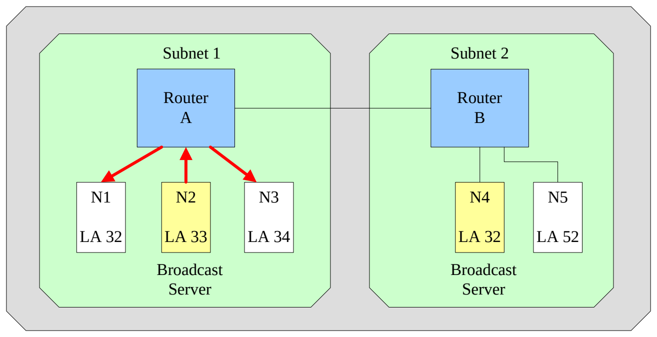


Figure 9. Subnet Broadcast, "Type 0" Message

Router configuration to prevent loops

Messages generated from the broadcast service are not sent back to the broadcast server because this would cause a broadcast loop, an infinite cycle of broadcast messages. In addition, by configuring the routers to drop messages with the logical address of 254 (Figure 10), we prevent local broadcasts from propagating to other routers, blocking a secondary avenue for broadcast loops. The manner in which the logical address 254 is used by the protocol identifier specification does not preclude having the router drop a packet which starts with 254.

In the following example, the local router is connected to a second router on port four. When the packet is sent out of port four, the path address of “<4>” is stripped (header deletion) and the leading byte of the packet is now “<254>”. A router receiving this logical address is configured to drop the packet. A node receiving this address understands that this is a dummy logical address associated with a SpaceWire path address and processes the packet.

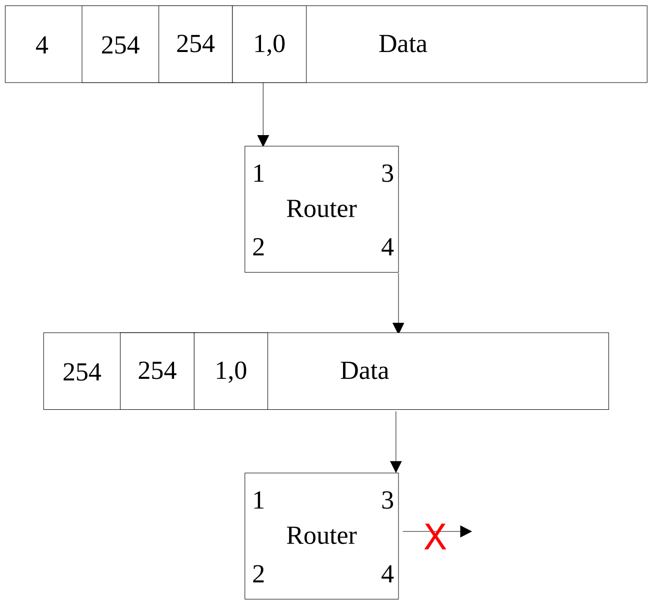


Figure 10. Router Blocking of Address 254

5. Broadcast and Higher-Level Services

The SpaceWire Standards body is currently investigating methods for supporting "Plug And Play," or the automatic recognition of devices when they are added to the network. The current proposal suggests that a device should announce itself when it is attached to the network [8]. One idea for this announcement is based on a router-centric protocol, where the routers initiate the announcement of the new device. Unfortunately, this would not be compatible with existing SpaceWire routers. If the device was able to initiate its own announcement through our broadcast protocol, "Plug And Play" could work with existing router hardware.

As a first step towards Plug And Play, we propose an automated method for routing configuration built on our broadcast protocol. In this section, we give three examples of how broadcast can support automatic network configuration and management.

Standard ARP

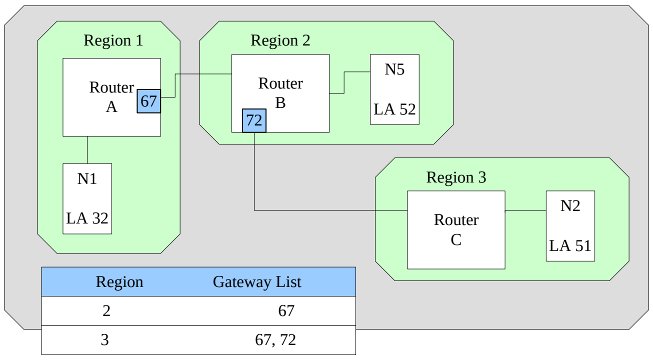
We define a SpaceWire hardware address based on a device's logical address and region. To create this SpaceWire hardware address, we define a region identifier to identify particular SpaceWire regions. Each region has unique identifying number between 0 and 255. The combination of the region identifier and the device's logical address define a unique SpaceWire hardware address.

When a device wishes to send a packet to a transport-layer address (such as an IP address), it must translate this address to a unique hardware address. Because we now have defined a unique hardware address and have built a link-layer broadcast service, this translation can be performed using standard ARP software. This is a major improvement in automatic configuration for SpaceWire networks.

Gateway Routing and Host Updates

If logical addresses are used and there is only one region, then routing is the responsibility of the routers; no route information need be maintained on the host nodes. However, when regional addressing is used, the host nodes must maintain gateway addresses to each region. It is far more cumbersome to maintain route information on host nodes than on routers. Currently this requires manual configuration and maintenance. To alleviate this problem, we have designed a method that uses our broadcast mechanism to *automatically* disseminate gateway routing updates to the host nodes.

In our approach, each host node maintains a table that gives the list of gateway addresses needed to reach a region from that node. The network driver forms the destination address by prepending the appropriate gateway addresses to the destination's LA.



Region	Gateway List
2	67
3	67, 72

Figure 11. Region to Gateway Translation

Figure 11 shows an example of how gateway tables are used. To send a packet from Region 1 to Region 2, the gateway "67" would be used. To send a packet from Region 1 to Region 3, the gateways "67" and "72" would both be prepended before the destination logical address.

The benefit of maintaining regional gateway tables on the hosts is that the tables need only be updated when there are changes to regional topology or gateway information. Adding a node to a region or changing node logical addresses does not require any change to the gateway tables. It is likely that nodes would be added or removed from regions more frequently than regions would be added or removed from the network.

To make this approach more adaptable than the traditional SpaceWire routing tables, it is important to reduce the maintenance overhead for the gateway tables. Because they are maintained on the host nodes and may differ for nodes in different regions, this is not a trivial question. The solution lies in using our broadcast service. Gateways tables can be dynamically and automatically managed by broadcasting update information whenever a change occurs. *In this way, manual configuration of routing information on host nodes can be eliminated.* This is an important step towards building Plug And Play networks on a regional level.

Standard DHCP

With our broadcast service, SpaceWire networks under IP stacks can utilize standard DHCP software. DHCP dynamically assigns a unique network-layer (e.g., IP) address to a device. DHCP is used on most terrestrial IP networks and has potential for reducing development costs for space systems.

The addition of ARP, DHCP, and dynamic routing made possible by our SpaceWire broadcast service would provide the building blocks for basic Plug And Play network configuration.

6. IP OVER SPACEWIRE

Standard IP may be embedded into SpaceWire packets. A SpaceWire driver receiving a packet with a protocol identifier for IP will strip off the SpaceWire headers and forward the embedded IP message to the IP stack. In this way, many of the standard services, such as ARP, may be automatically supported once the embedded data packet has been extracted. Since IP is the backbone of most of the commonly used internet services, a few additions to SpaceWire, such as broadcast, will enable IP applications to run seamlessly over SpaceWire.

7. BROADCAST PROTOCOL DEVELOPMENT

To demonstrate the broadcast protocol we are developing a protocol simulation and network driver under a Southwest Research Institute® (SwRI®) internal research project. The driver will support the SpaceWire Link Interface Module (SLIM) 3U CompactPCI network card developed at SwRI® [1]. It will implement the broadcast protocol described in this paper. The simulation will be used to validate the protocol and benchmark operation for large networks with various topologies. Both the simulation and the driver will use the same core protocol primitives.

Driver Development

Since the low-level interface details of the SwRI® SLIM were readily available, we chose this card as the basis for our driver development. The driver is based on embedded Linux with a 2.6 kernel. The driver will implement the SpaceWire packet encapsulation and broadcast service. The encapsulation service will initially be designed to recognize IP, but could easily be expanded to handle other high-level protocols.

When messages arrive on the network interface, they are inspected by the encapsulation service to determine their protocol type. The messages are then passed to the appropriate network stack hooks. For example, IP messages are passed to the IP stack for processing, or raw messages may be passed straight to a custom application for processing. On a broadcast server, the driver would pass broadcast messages to the broadcast service to be processed by the SpaceWire broadcast protocol. As described in Section 4, if a broadcast message is a Type 0 message, a Type 1 message will be sent to all other broadcast servers. The message will also be passed up the stack to the encapsulation service for further processing. If the message is a Type 1 message, a Type 0 message will be sent to all other ports on the server's local router.

Messages targeted for transmission to the network are passed to the encapsulation service to be tagged with the appropriate SpaceWire protocol identifier. If the encapsulation service detects that the outgoing message is a broadcast, the message will be passed to the broadcast service. Once in the broadcast service, a list of SpaceWire path addresses will be generated to target the message as a Type 0 broadcast to all local router ports.

The driver may be configured on startup using a configuration file or through ioctl() (input/output control) calls from an application. The driver configuration information includes a list of the SpaceWire hardware addresses for the broadcast servers and an initial list of known regions and their associated gateways.

SpaceWire Demonstration Network

Our test network uses three SwRI® SpaceWire SLIM cards as well as a COTS SpaceWire link interface to illustrate device interoperability. We are using a 4-Links SpaceWire-PCI network card and two eight-port STAR-Dundee SpaceWire routers [10].

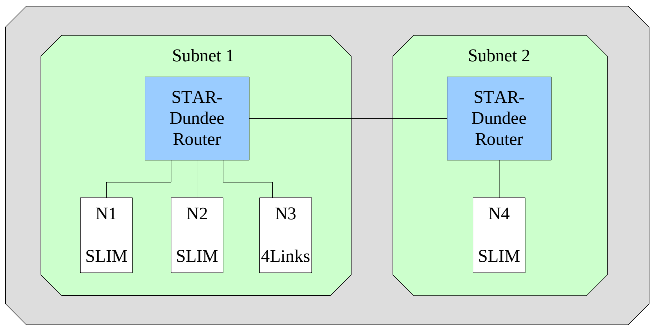


Figure 12. SpaceWire Demonstration Network

Using the network configuration shown in Figure 12, we can demonstrate broadcast message propagation over a simple network, over a multi-region topology, and between two adjoining routers. This not only demonstrates the broadcast action of the protocol, but it also demonstrates that the protocol does not produce broadcast loops.

Simulation

To further exercise our SpaceWire protocol implementation, we are developing a network simulation. The simulator uses the core encapsulation and broadcast functions from the SLIM driver to process packets. A wide range of topologies may be specified through configuration files. The simulated devices have the ability to act as routers, broadcast servers, or ordinary host nodes. Each device specified in the network topology generates an individual process. The processes are linked by socketpairs. Each socketpair represents a physical network link between devices on the network. Configuration files are also used to designate scheduling and data content of message transmission. The standard configuration files used by the SpaceWire driver are also used for the simulation.

We are testing a number of topologies. We will evaluate the transmission of messages from broadcast servers as well as ordinary network nodes. Statistics quantifying the overhead required to perform the broadcast to all nodes will be tabulated. Information about router loading will also be collected. Preliminary results on three network configurations are shown below.

The first network is a simple, single router topology with eight host nodes attached (Figure 13). Broadcast messages are generated by any of the host nodes and propagated to all the other nodes by the SpaceWire broadcast protocol. In this example, the end node originating the message would send a Type 0 message to all other ports on the router. In this topology, a broadcast server is not required, since there is only one subnet. The network driver performs the same broadcast protocol operations on each node. The existence of the broadcast server here provides the capability to support the future attachment of other subnets.

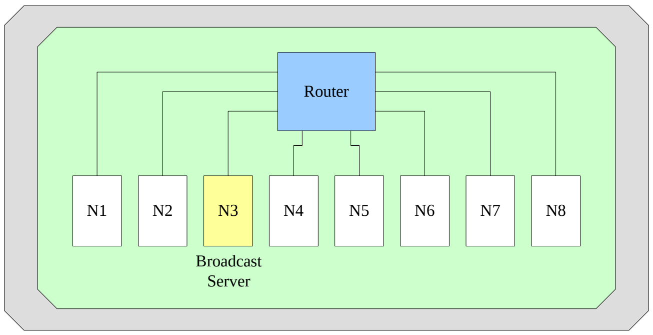


Figure 13. Single Router Simulation Topology

To evaluate broadcast server interactions across multiple subnets, a larger, eight-router, forty-eight host node topology was evaluated (Figure 14). Each router has a broadcast server. Any of the six nodes attached to a router may be designated as the broadcast server. Messages originate from a host node as a Type 0 broadcast sent to all the ports on the local router using path addressing. All routers are configured to drop messages starting with the dummy logical address of 254. Since the path address is stripped off by the local subnet router, the first byte of any Type 0 message received by a router in the adjoining subnet will have the value of 254, causing the message to be dropped. This mechanism prevents broadcast loops. When the subnet broadcast server receives a Type 0 message, it sends a Type 1 messages to all other broadcast servers on the network. The other broadcast servers issue Type 0 messages to their subnets, completing the broadcast to all host nodes.

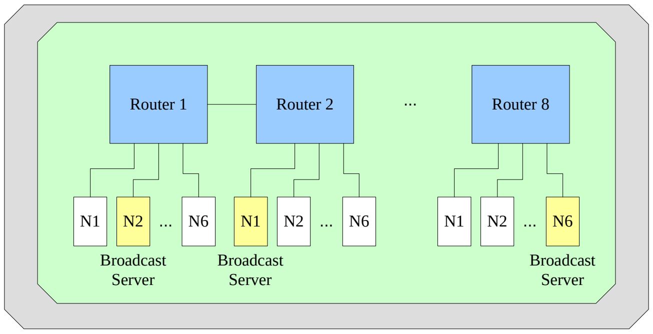


Figure 14. Multiple Router Simulator Topology

The third topology uses redundant router linking, a common technique in space missions (Figure 15). Four routers with six host nodes each are connected by multiple paths for alternate data routes in case of link failure. This network contains physical loops, providing a good illustration of the loop-free nature of our SpaceWire broadcast protocol.

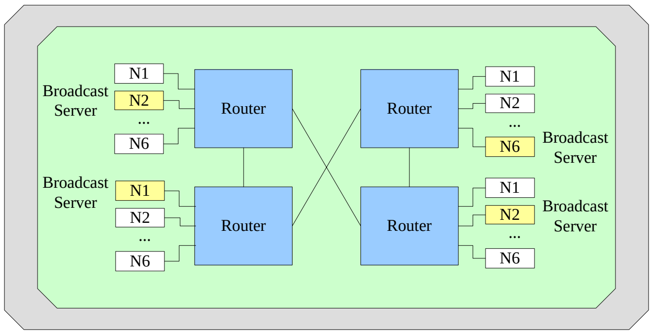


Figure 15. Redundant Router Simulator Topology

Broadcasts for the redundant router topology occur in the same way as those in the multiple-router topology. Type 0 messages are sent from the broadcasting host node to the rest of the local subnet and the broadcast servers distribute the messages across the remainder of the network.

The preliminary simulation results show that the SpaceWire broadcast protocol is performing a complete and efficient distribution of messages across the network. The messages transmitted contained one byte of data, four or five header bytes and an end-of-packet marker (EOP). While a one byte payload is not a typical packet size, it is sufficient to demonstrate that data will be routed correctly across the network to provide a broadcast. The header length varies because during path addressing (Type 0 messages) the extra dummy logical address byte must be included. The address bytes are followed by the broadcast protocol ID and the one byte broadcast type. Following the broadcast header information, an additional protocol ID byte is used to indicate the type of information imbedded in the broadcast.

	Single	Redundant	Multiple
Host Nodes	8	24	48
Total Messages Read	7	23	47
Total Bytes Read	42	138	282

Figure 16. Simulation Host Node Statistics

In each topology tested, Figure 16 shows that all host nodes (N-1) received exactly one copy of the broadcast message. In the simulation, the nodes know which router ports they are attached to; the node that initiates the broadcast does not send a message to itself. There were no host nodes missed in the distribution and no extra message copies were delivered.

An additional benefit of the broadcast protocol is that it distributes the work load to all the routers in the network. If a broadcast is performed by sequential unicast, the router on the subnet initiating the message has the burden of forwarding one message for every device on the network. The broadcast protocol requires each router to forward one message to each of its own nodes, with only a few additional messages required to contact broadcast servers located in other subnets. Router traffic statistics (Figure 17) show that the router load is well balanced across the network.

	Single	Redundant	Multiple
Routers	1	4	8
Total Writes	7	34	83
Writes Per Router	7	10,9,7,8	13,14,12,11,10,9,8,6
Avg Writes	7	8.5	10.375

Figure 17. Simulation Router Statistics

The router data also shows that the number of messages encountered by the router (Figure 17) is greater than the number of messages received by the host nodes (Figure 16). This is due in some cases to router ports which drop messages because there is no device attached. It is more commonly due to Type 0 messages received from adjoining routers which are dropped intentionally (through configuration) to prevent broadcast loops and to the action of forwarding Type 1 messages to broadcast servers on other subnets.

The simulation results indicate that the broadcast protocol is functioning as designed and is ready to be integrated into the driver. It distributes messages across the network using only a small number of extra control packets. The message overhead of two bytes required for the broadcast is a small addition to the packet size. Further analysis and comparison to other broadcast methods will be performed in the future.

8. Related Work

Non-broadcast ARP

At the 2006 IEEE Aerospace conference, we introduced the idea of non-broadcast ARP [2]. During the early ARP implementation, it became clear that a general link-layer broadcast ARP would be more useful. A broadcast protocol would be able to support the standard ARP, as well as being generally available for use by other services such as DHCP or for application-layer broadcast.

SpaceWire Plug And Play Working Group

The SpaceWire Plug And Play Working Group has proposed a router-based approach to the Plug And Play detection of devices [8]. The routers would store addresses to all known devices on the network and broadcast configuration or topology state changes by sequential unicast. Using our SpaceWire broadcast protocol instead could reduce the workload of routers when delivering these messages across the network. Having support for standard broadcast dependent services could also aid in the automated configuration of Plug And Play devices.

9. Conclusions

In this paper, we have introduced a straightforward, software-based approach to implementing a SpaceWire link-layer broadcast. We have described the protocol, driver development and a SpaceWire demonstration network. Preliminary simulation results illustrate that the protocol provides a complete, loop-free broadcast.

The broadcast protocol is compatible with the existing SpaceWire standard and existing COTS SpaceWire routers and interface hardware. The broadcast protocol and the unique SpaceWire host address, based on the combination of logical address and region identifier are necessary to support standard IP ARP and DHCP over SpaceWire as well as a new concept of regional gateway routing. By providing these additional building blocks to SpaceWire networks, we have taken a step towards supporting more industry-standard network applications (such as those based on IP) and towards Plug And Play networks over SpaceWire. Continuing research will help to further reduce development cost and deployment time of SpaceWire-based systems, making SpaceWire a more attractive choice for future space missions.

REFERENCES

[1] Mark A. Johnson, Buddy Walls, Kristian Persson, Sandra G. Dykes, "Design of a Reusable SpaceWire Link Interface for Space Avionics and Instrumentation," MAPLD, 2005.

[2] Sandra G. Dykes, Buddy Walls, Mark A. Johnson, Kristian Persson, "A Non-Broadcast Address Resolution Protocol for SpaceWire Networks," IEEE Aerospace, Big Sky, Montana, March, 2006.

[3] Robert Klar, Sandra G. Dykes, Allison Roberts, Chris Mangels, Buddy Walls, Mark A. Johnson, Kristian Persson. "Internetworking Over SpaceWire: A Link-Layer Broadcast Service for Network Stack Support," Presentation at the Fifth Space Internetworking Workshop, September 2006.

[4] ECSS-E-50-12A, "Space Engineering: SpaceWire – Links, nodes, routers, and networks," ESA-ESTEC, January 2003.

[5] ESA SpaceWire Web site http://spacewire.esa.int/content/Missions/NASA.php

[6] Glenn Rakow, Richard Schnurr, Steve Parkes,"SpaceWire Protocol ID: What Does It Means To You?," IEEE Aerospace, Big Sky, Montana, March, 2006.

[7] Steve Parkes, "New Addition to Standard: Protocol ID," SpaceWire 101 Seminar, MAPLD 2006.

[8] Patrick McGuirk, "SpaceWire Plug and Play (PnP)," SpaceWire 101 Seminar, MAPLD 2006.

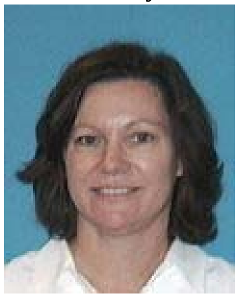
[9] IETF RFC 826, "An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware," IETF, November 1982.

[10] STAR-Dundee SpaceWire Routers, http://www.star-dundee.com

BIOGRAPHY

Allison Roberts is a Research Analyst in the Automation and Data Systems Division of Southwest Research Institute®. She is currently developing the SpaceWire broadcast protocol simulation and is involved with the deployment of SpaceWire on future NASA missions. Ms. Roberts has also worked with health and status monitoring systems for the testing of commercial aircraft and with NASA Langley's Space Shuttle infrared damage detection software. She has a B.S. in Physics and an M.S. in Applied Science from the College of William & Mary.

Sandra G. Dykes is a Principal Research Scientist in the Automation and Data Systems Division of Southwest Research Institute®. Her primary interests are in the areas of network protocols, routing, and cooperative infrastructure systems. Dr. Dykes has a B.S. in Chemistry from the University of Texas at Austin, an M.S. in Chemistry from the University of Texas at San Antonio, and an M.S. and Ph.D. in Computer Science from the University of Texas at San Antonio. She is a member of IEEE Communication Society and IEEE Computer Society.



Robert Klar is a Senior Research Engineer and Group Leader of the Embedded Systems and High-Reliability Software Group in the Automation and Data Systems Division of Southwest Research Institute®. He is currently involved in SpaceWire research and the development of flight software for the Magnetospheric Multiscale (MMS) Mission. He has a B.S. in Computer Engineering from Texas A&M University and an M.S. in Electrical Engineering from St. Mary's University. He is a member of the IEEE Computer Society.

Christopher C. Mangels is a Research Analyst in the Automation and Data Systems Division of Southwest Research Institute®. He is currently developing the driver for the SpaceWire hardware. He has a B.S in Computer Science from Southwest Texas State University.

ACKNOWLEDGEMENTS

The work was achieved with funding from the SwRI® Internal Research & Development program (R9623). The authors wish to acknowledge support from the SwRI Division 15 SLIM board development team and to thank Steve Parkes of STAR-Dundee for his enthusiastic support and assistance with SpaceWire routers