INTEGRATION OF INTERNET PROTOCOLS WITH SPACEWIRE USING AN EFFICIENT NETWORK BROADCAST

Session: SpaceWire Networks and Protocols

Short Paper

Robert Klar, Sandra G. Dykes, Allison Bertrand, Christopher C. Mangels

Southwest Research Institute, San Antonio, TX, 78238

E-mail: robert.klar@swri.org, sandra.dykes@swri.org, allison.bertrand@swri.org, christopher.mangels@swri.org

ABSTRACT

SpaceWire is gaining popularity for space applications because of its simple circuitry, low power consumption, and high link speeds. Unlike the ubiquitous Ethernet which dominates the terrestrial Internet, SpaceWire does not include a link-layer broadcast. Common protocols that provide support for automatic network configuration such as the Address Resolution Protocol (ARP) and the Dynamic Host Configuration Protocol (DHCP) often rely on broadcast for discovery. Since SpaceWire does not explicitly provide provisions for these, Internet Protocol (IP) addresses and address resolution tables on each host must typically be manually configured. For large networks, this can be both a cumbersome and error-prone process.

This paper describes an efficient, loop-free, link-layer broadcast service for SpaceWire that supports automatic IP network discovery, configuration and management. Because it is implemented in the host software device drivers, our simple approach requires no changes to existing router or host interface hardware. It facilitates an easy integration path for existing protocol stacks and enables the use of standard ARP and DHCP implementations. The broadcast protocol is based on the concept of a SpaceWire subnet which consists of a router and its directly connected hosts. One host per subnet acts as the subnet server. The subnet server distributes broadcast messages locally to hosts on the same router. To extend a broadcast globally, the subnet server sends the message to the other subnet servers. We include performance results from simulation experiments, analytical results, and a prototype implementation. This research enables the convenient use of existing higher-level protocols and applications, thereby providing much promise for reducing the cost and time required to develop SpaceWire systems.

INTRODUCTION

Since its standardization by the European Space Agency (ESA) [1], SpaceWire, a high-speed low-power network, has rapidly been adopted for application on space missions by NASA, ESA, and other major space agencies [2]. By design, the standard was kept simple. The standard specifies SpaceWire “as a means of sending packets from a source node to a destination node” but does not specify packet contents beyond basic addressing [1]. Consequently, system designers have developed different communication protocols, most requiring static configuration.

Such implementations can become costly because recurring engineering is required to adapt the network configuration to support each different mission. This is contrary to one of the primary purposes of the standard – to reduce system integration costs.

Fortunately, the SpaceWire Working Group (SWG) provided some remedy for this situation by creating a standard encapsulation header [3]. This enables the multiplexing and demultiplexing of packet types belonging to different higher-level protocols, including IP stacks. The encapsulation header may also be used for automatic network configuration protocols such as ARP and to support emerging Plug-and-Play protocols for SpaceWire.

Underlying automatic protocols are the fundamental mechanisms of node discovery, route discovery, and address assignment. To that end, this paper presents three key concepts:

- A mapping of SpaceWire nodes to network unique addresses,
- A loop-free link-layer broadcast for SpaceWire to support integration of higher-level protocols, and
- Application of the broadcast to support some standard Internet protocols, i.e. Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP).

UNIQUE ADDRESSES FOR SPACEWIRE

When transmitting, a SpaceWire node addresses a packet using a consecutive sequence of preceding bytes. An address byte in the range (1 ... 31) is a *path address* and specifies a physical output port on the router attached to a node for transmission. An address in the range (32 … 255) is a *logical address* that must be resolved by a SpaceWire router using a forwarding table to determine the output port for transmission. Logical addresses 254 and 255 are reserved. The SpaceWire standard provides for further extension of the address space by assigning each node to a unique region where logical addresses are not duplicated [1]. Typically, regions can be inferred from the router forwarding tables.

The problem for packet delivery is that SpaceWire logical addresses may be reused in different regions and therefore are not necessarily unique. We solve this problem by assigning each region a unique identifier called the RegionID, and using the combination of RegionID and logical address as the unique hardware address.

SPACEWIRE BROADCAST

In order to describe the broadcast, we define the term *SpaceWire subnet* (Figure 1) to indicate a single router and all of the nodes directly attached to its ports. A node on each subnet serves as a *subnet server*. (Although a subnet server could easily be embedded in router hardware, a node implementation was chosen to make use of off-the-shelf routers.) The subnet server has a special role and requires more information about the network than other nodes. It must be aware of the output port(s) on the local router from which it receives packets and have an address table for the subnet servers on other subnets. This information can be gained through methods of network discovery such as the Plug-and-Play protocols currently being proposed [4].

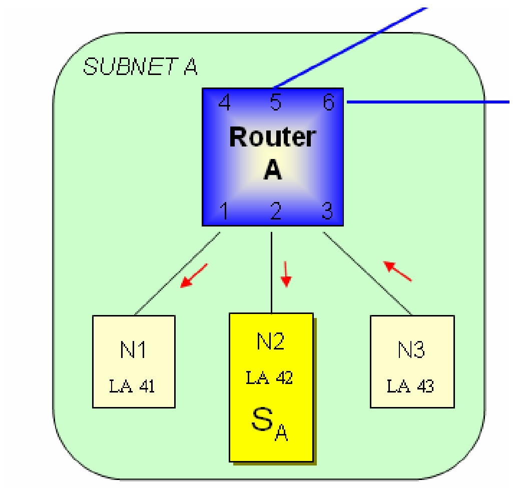

Figure 1. A SpaceWire Subnet

A protocol identifier (PID) of 253 was used to designate a broadcast in the encapsulation header. Following the header, the next byte was used to designate a broadcast message type. To implement broadcast, only two types are needed. These are designated Type 0 and Type 1. Type 0 messages are always sent using path addressing as a simple way to target messages to all devices attached to the subnet router. Type 1 messages are sent using logical addressing.

When a node wishes to send a broadcast, it sends a Type 0 message to all nodes on its local router. When the subnet server receives the Type 0 message, it resends the message as a Type 1 message to the other subnet servers in its broadcast table.

When a subnet server receives a Type 1 message, it resends the message as a Type 0 message to each port on its local router except itself. It passes the message contents directly to its higher-level protocol message handlers. In this way, the message is distributed efficiently throughout the network.

In networking, a broadcast loop occurs when the same packet is received more than once and resent, causing a network blockage or possibly a broadcast storm. In order to prevent such loops, the routers on the network are configured to drop messages with the logical address of 254. Since Type 0 messages are sent with path addresses where the output port is removed by the router, this guarantees that neighboring routers receiving the Type 0 message will discard the packet. Also, subnet servers must never send messages to themselves.

INTEGRATION WITH INTERNET PROTOCOLS

When a node receives a message, it strips off the broadcast encapsulation header to find another encapsulation header. The protocol identifier in this second encapsulation header can be used to designate higher-level protocols such as IP, ARP, and DHCP, supporting integration with a standard IP network stack.

ARP provides a way to translate a network address to a hardware address used for transmission at the link-layer. Because we have defined a unique network address based on a region identifier and a logical address, we can now make use of ARP. In our implementation, we chose to use a single byte for the region identifier.

Integration with a network stack also allows DHCP to be used for configuration of SpaceWire networks. DHCP dynamically assigns a unique IP address to a device and can provide other optional configuration information.

SIMULATION, IMPLEMENTATION AND TESTING

To demonstrate the broadcast, we developed a software simulation and a network device driver as internal research project funded by Southwest Research Institute (SwRI®). The simulation and the driver use the same core primitives.

Simulation and analytical study of the broadcast protocol were used to demonstrate correctness and provide comparison to a sequential unicast. For broadcast, an optimization where the broadcast only transmits to active ports on the attached local router was also considered. The results from the analysis matched those produced by the software simulation.

Although many topologies were considered, as an illustrative example, we chose to describe the results for two:

- A linear topology comprised of a linear sequence of eight routers with six hosts on each router. Each router has a single link to each adjacent router.

- A mesh topology comprised of a network with redundant links as might be found in a fault-tolerant spacecraft. Four routers with six hosts each were attached with redundant links between them.

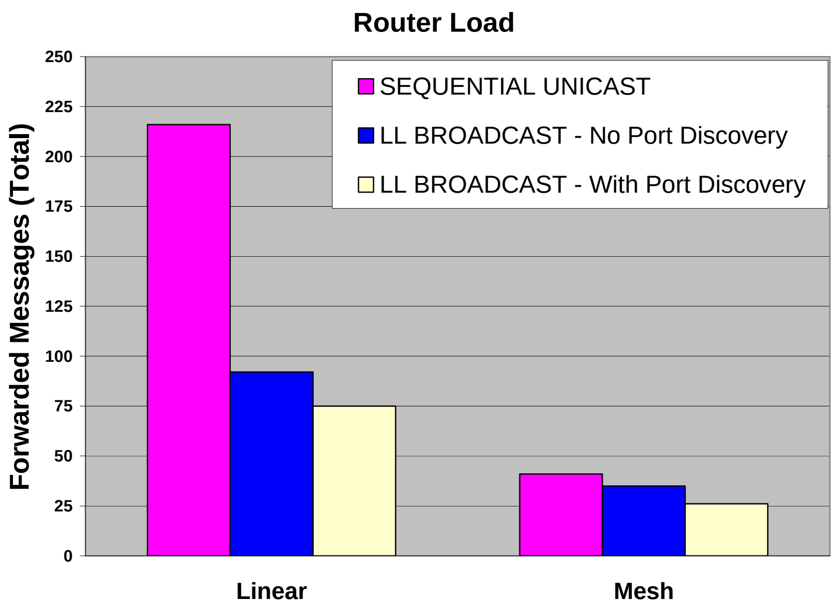

Figure 2. Comparison of Broadcast vs. Sequential Unicast

A study of the total number of packets required to complete the network-wide broadcast (Figure 2) shows that for the linear topology, our broadcast protocol reduces the number of packets transmitted by more than half. For the redundant mesh topology, our broadcast also requires fewer messages than sequential unicast.

A network device driver was developed for the SpaceWire Link Interface Module (SLIM), a 3U CompactPCI network card developed at SwRI® [5]. The driver was developed on an embedded Linux 2.6 kernel and implements the SpaceWire encapsulation header and broadcast service. The device driver is being tested using a configuration of three SLIM cards and two eight-port STAR-Dundee SpaceWire routers [6].

CONCLUSION

We have introduced a broadcast for SpaceWire that may be used as a foundation for support of standard network software stacks. Enabling the use of standard network stacks provides potential to reduce recurring engineering costs by enabling more standard, off-the-shelf applications to be used in space mission contexts.

REFERENCES

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

[2] ESA SpaceWire Website: http://spacewire.esa.int/content/Missions/NASA.php

[3] CSS-E-50-11 Draft B, "Protocol Identification," ESA-ESTEC, February 2005.

[4] Patrick McGuirk, et. al. "SpaceWire Plug and Play (PnP)," AIAA Infotech 2007.

[5] Mark A. Johnson, et. al., "Design of a Reusable SpaceWire Link Interface for Space Avionics and Instrumentation," MAPLD 2005, September 2005.

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