SAND2011-5948 C

SAND2011-5948C

SPACEWIRE NETWORK SIMULATION OF SYSTEM TIME PRECISION

Session: Networks and Protocols

Long Paper

Brian Van Leeuwen, John Eldridge, Jacob Leemaster

*Sandia National Laboratories***

Albuquerque, USA

E-mail: bpvanle@sandia.gov, jmeldri@sandia.gov , jeleema@sandia.gov

ABSTRACT

Many applications sharing a SpaceWire network require synchronized system time and SpaceWire can be employed to distribute system time. However, in its current form general system time distribution capability is lacking. In this paper we present a system time distribution approach that employs a broadcast extension to the SpaceWire protocol. Broadcast messages distribute specific time values while SpaceWire Time-Codes clock-in or trigger the specific time contained within the broadcast. The broadcast approach is effective in minimizing network resource usage by distributing the broadcast-time message in a partial-parallel method. Additionally, for the objective of identifying the timing precision and jitter for specific network architectures and network states, high-fidelity models were developed to quantify the timing variations and to analyze overall SpaceWire networked system performance.

1. INTRODUCTION

The European Space Agency (ESA) in collaboration with other international space agencies supports a serial data link standard to enable the transfer of large amounts of data onboard satellites. The standard named SpaceWire and defined in [1], is a satellite communication network based in part on the IEEE 1355 standard of communications. A SpaceWire network is typically comprised of a number of links, nodes and routers. SpaceWire routers are necessary since a SpaceWire node will only support a few links and thus can only be directly connected to a limited number of nodes. Routers also reduce the number of point-to-point links and enable redundant paths in case of link failures. The current standard describes a mechanism that can enable modern satellite systems to transfer large amounts of data on board the satellite. However, the standard currently lacks a time distribution capability to enable time synchronization among the various applications on the SpaceWire network [1].

Additionally, a high-fidelity modeling and simulation capability to perform analysis of SpaceWire networked systems is lacking. This analysis capability should provide precise time

** Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

synchronization results under various proposed network architectures. This analysis capability should also provide results as the architecture under study dynamically changes when faults occur and redundant paths are utilized. To meet these analysis objectives we created high-fidelity model representations of SpaceWire nodes, links, and routers that can be configured to represent any proposed network architecture.

In this paper, we present a time distribution mechanism that can be implemented in a SpaceWire network that employs the standard SpaceWire Time-Code function along with a custom SpaceWire broadcast capability. Together, the Time-Code function and broadcast capability enable a means to distribute time to the various applications utilizing the SpaceWire network.

Our approach required that the general broadcast extension be a layer upon the existing SpaceWire standard. That is it would be compatible with the existing protocol and not necessitate the modification of existing intellectual property or the revision of the existing SpaceWire standard. Rather, the objective was to extend the standard to include the new capability. Our approach met this objective.

2. SYSTEM TIME DISTRIBUTION WITH SPACEWIRE BROADCAST EXTENSION

The current SpaceWire standard lacks both a general time distribution function and broadcast function. Multiple solutions have been proposed each with their own benefits and deficiencies [2][4]. To this end, we designed a time distribution function that utilizes a SpaceWire broadcast capability. The time distribution function is designed to work in concert, be network efficient, and fully backward compatible with the current SpaceWire standard.

2.1 SYSTEM TIME DISTRIBUTION

The time distribution function distributes what we consider to be *system* time. In our time distribution mechanism, *system* time refers to distributing actual time versus the SpaceWire standard Time-Code. The standard SpaceWire Time-Code comprises the SpaceWire ESC character followed by an eight-bit data character. The data character contains 6-bits of system time and two control flags. A time-master node asserts a periodic “tick” and immediately sends out a Time-Code with the 6-bit time field incremented prior to transmission [3]. This Time-Code mechanism is limited to a 6-bit resolution and increments each network device’s internal time counter from the current Time-Code value to the next. The counter, which is intended to prevent looping retransmission of the Time-Code and not necessarily to carry a time value, rolls from its maximum value of 63 to zero because of its 6-bit field size limit.

Our time distribution approach accomplishes synchronization of system time. This is done by the time-master node sending a system wide broadcast containing what the system time will be at the next Time-Code “tick.” The broadcast message is transmitted a predetermined time period prior to the transmission of the Time-Code. The predetermined time period is an estimated value that is equal to the worst-case time for the broadcast message to propagate throughout the network.

The time-master node transmits a Time-Code tick indicating to the network that the time described in the previous time message is now current. Thus, the various network applications have access to an unambiguous system time.

Unambiguous system time is a 32-bit integer representation and it is broadcast to all nodes in the network. When the endpoints receive a SpaceWire Time-Code the broadcasted system time message is accepted as the current time after having been validated by combinational logic. The time endpoint evaluates whether it is synchronized with the rest of the SpaceWire network with every received SpaceWire Time-Code "tick." If the endpoint believes itself to be synchronized with the rest of the endpoints in the SpaceWire network, it considers itself to be "locked" and asserts a corresponding signal.

The endpoint determines if it is "locked" in the following way: After every "tick," the expected value of the next system time message is calculated. The calculated value is considered to be the value of the current system time message plus one. If the next received system time message matches the expected value, the endpoint concludes that it is synchronized with the rest of the network.

If the next received time message does not match the expected value, or no system time message is received by the next "tick," then the endpoint assumes that a synchronization error has occurred, indicates that it's no longer "locked," and will simply increment its system time as a "best guess." If the time message arrives late, it will not interfere with operation so long as the subsequent time message arrives on time, as the new message will override the late message.

In our approach it takes two correct time message/tick pairs to achieve a synchronization "lock." It is a known issue that if a series of two or more time messages are consistently late by a tick period (or a consistent multiple of the tick period), then the timekeeper will erroneously indicate a lock and synchronize to the late packets as they appear identical to a correct time message/tick sequence. Expanding the number of previous packets considered when calculating the expected time value would reduce the likelihood of such a situation at the cost of increasing the number of correct packets it takes to achieve a "lock."

2.2 BROADCAST

Our time distribution function employs a hybrid broadcast approach derived from work described in [4]. The approach was modified with the goal of distributing system time, be compatible with existing SpaceWire hardware, and not suffer from loops or broadcast storms. The approach creates a "broadcast server" to be hosted by each router in the network. Our implementation of this approach has two main configurable aspects: which local ports will receive broadcasts and a list of the logical addresses of all *other* broadcast servers in the network.

A packet intended for broadcast is transmitted to the local "broadcast server," which then forwards the packet to all other broadcast servers in the network. Once this is completed every

broadcast server in the network will forward the packet to the appropriate local ports on its respective router. The broadcast servers use several techniques at the protocol level to guarantee that no loops, infinite broadcast storms, or spurious re-broadcasts occur. The broadcast mechanism used for our SpaceWire Broadcast Server (SpWBS) includes several stages:

Local-to-Server Stage - A SpWBS receives a Local-to-Server type packet containing the broadcast message

Server-to-Server Stage - The initiating SpWBS sends a Server-to-Server type packet containing the broadcast message to every other enabled SpWBS in the network.

Server-to-Local Stage - Once a SpWBS receives a Server-to-Server stage packet or the initiating SpWBS finishes the Server-to-Server stage transmission it sends Local-to-Server type packets with the broadcast message to every enabled and connected local port.

This broadcast approach has efficiencies in that it partially distributes bandwidth utilization across the network and obtains parallelization of the Server-to-Local stage of broadcast. The approach requires that every router with nodes receiving broadcast messages have an attached SpWBS and it requires an additional header byte to distinguish between Local-to-Server, Server-to-Local, and Server-to-Server type messages.

The SpWBS broadcast approach includes mechanisms to prevents broadcast storms. All local ports and broadcast server addresses are disabled by default and must be explicitly enabled by server configuration. A configuration error that results in a Server-to-Local packet to be received by another SpWBS will be detected by identification of an incorrect header byte and prevented from further broadcast.

3. MODEL DEVELOPMENT

Our SpaceWire model development is done in the OPNET Modeler network simulation environment [5]. OPNET Modeler includes an extensive model library of network devices; however, OPNET Modeler does not include SpaceWire models in its standard model library. Fortunately, OPNET Modeler includes the capability for users to develop nodes based on custom protocols. To analyze the performance of our time distribution mechanism detailed SpaceWire models are developed.

The models include many features of the SpaceWire standard including the functionality at the various communication stack levels in both the end nodes and wormhole router. The models faithfully implement the disassembly of application layer data and the reassembly of the resulting NChars at the destination node. Additionally, processes such as the startup sequence, flow control, Time-Code process, and realistic representation of various buffering and queuing functions.

A modeling objective was to have representative models of the various SpaceWire modules and protocols to support system design activities in all phases of a project. This modeling would range from custom protocol extension analysis to assessing SpaceWire architectures and their operation under stressful scenario conditions resulting from link and node failures. In pursuit of this objective, we developed models to be modular. The modular approach enables the combination of end nodes and routers in various architectures. Figure 1 illustrates an example SpaceWire network and one of the nodes in the example network. In this example, each node is comprised of three specific modules; an application node, a wormhole router, and a broadcast server. The SpaceWire router is the connection point that combines the various applications nodes and broadcast servers.

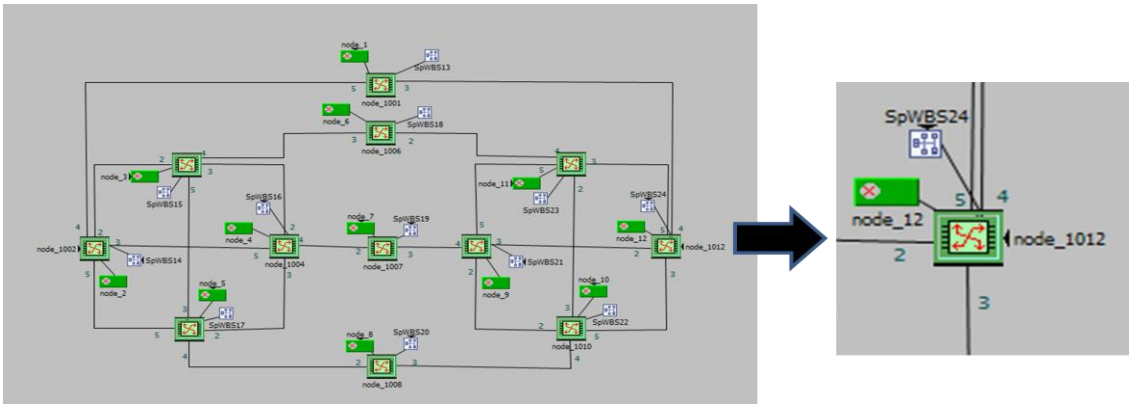

Figure 1: Example SpaceWire network topology.

Figure 2 illustrates a description of the custom SpaceWire node model and a single process model as developed in OPNET Modeler. On the left side of Figure 2 is the SpaceWire application node model that includes the protocols used in each layer of the SpaceWire communication stack. The node model includes various application types that access the network. Specifically, a state-of-health (SOH) application that periodically shares state of health (SOH) details. A standard application layer can be a data producer, such as a sensor, or a data consumer, such as a telemetry downlink, or a broadcast application that creates messages intended for broadcast.

Each of the square blocks in a node model represent a single or multiple process model state machines and implements the protocol of interest. Figure 2 (right side) illustrates an example process model. The process model illustrated in Figure 2 is a root process that can spawn child processes. Child processes are particularly applicable in modeling the wormhole router. The root process spawns a child process for each data flow through the wormhole router. In many cases multiple child process are in various states as they represent multiple simultaneous data flows through the router.

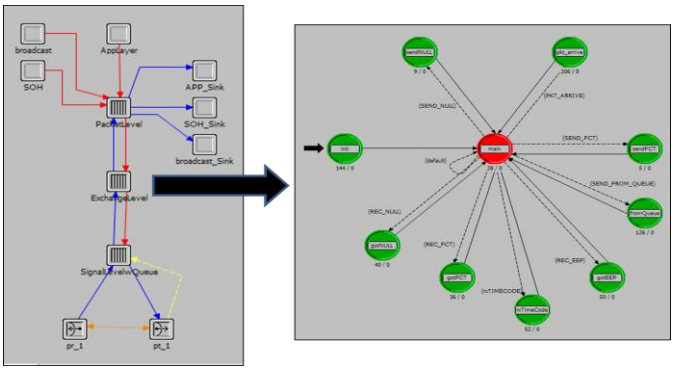

Figure 2: SpaceWire application node model (left) and an example process model (right).

OPNET Modeler includes rich mechanisms to create network traffic. In our time synchronization analysis, we are able to clearly identify when messages supporting system time distribution are created and when they arrive at their intended target. Additional application layer traffic can be generated to represent actual data files being transported through the network as NChars. Thus, Time-Code traffic is created and introduced into the network along with typical application layer traffic and its impact on delaying Time-Code messages.

4. System Time Distribution Precision Analysis with High-Fidelity SpaceWire Model

To demonstrate our time synchronization analysis capability we created a SpaceWire network comprised of 12 nodes, routers, and broadcast servers as shown in Figure 1. The architecture, constructed in OPNET Modeler, uses the various custom nodes and process modules. In this demonstration case Node 10 is considered the time master and thus originates both the Time-Codes and the system-time broadcast messages. Employing our time distribution mechanism, Node 10 will create a broadcast message immediately following a Time-Code transmission. The broadcast message will be transmitted to the broadcast server associated with the router shared by the broadcast server and Node 10 (i.e., Node 1010). This broadcast message contains the time that the next transmitted Time-Code will clock into the various network slave nodes. Since Time-Codes are not delayed by full application layer file transfers the broadcast will not arrive at a slave node prior to the previously sent Time-Code. However, there is no guarantee that the broadcast message will arrive at the slave nodes prior to the arrival of the following Time-Code transmission. In cases, where the following Time-Code arrives at the slave node prior to the broadcast time message the system is said to have lost synchronization "lock." We examine the network in Figure 1 for time synchronization precision.

5. Results and Discussion

The network in Figure 1 with Node 10 producing both the Time-Codes and the broadcast messages is assessed for time distribution delay variation. In this analysis, we record the receipt of a broadcast message and the time the broadcast message time value is clocked into the slave

node's clock. The node's time is then compared with a global absolute time. The difference of the absolute time and node clock time is recorded and plotted in Figure 3 as a probability density function (PDF).

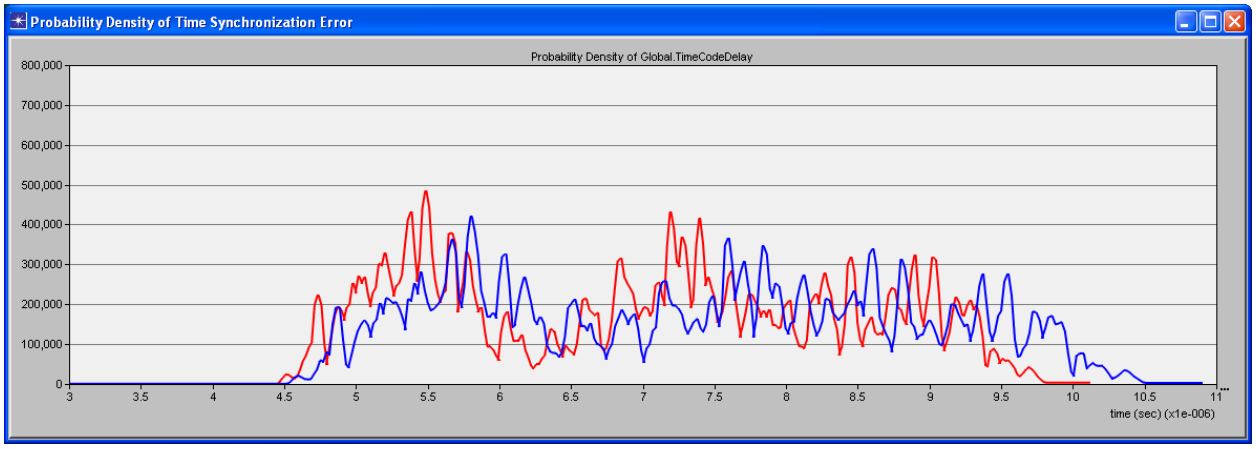

Figure 3: Resulting PDF of the time synchronization error when network is lightly loaded (red trace) and heavily loaded (blue trace). Note the Y-axis should normalized by dividing by 50E6.

Figure 3 describes a time synchronization error averaging approximately 7.0 µsec. The plot has three regions centered at approximately 5.5 µsec, 7.3 µsec, and 9.0 µsec. Each region describes the variation in time synchronization based on the number of hops to forward the Time-Code. Each additional hop adds more variation and thus leads to more spreading of the plot as you move from left to right on the time axis. The variation between the lightly loaded network (red trace) and the heavily loaded network (blue trace) results from additional NChars on the network that may delay the transmission of a Time-Code. The variation is not significant since a NULL can cause a delay of up to an 8-bit transmission time whereas an NChar can cause a delay of up to a ten-bit transmission time. In the demonstration network, the SpaceWire links operate at 10 Mbps. Also note that the Time-Code period was 6 msec. and maximum application-layer file size was less than 60 Kbits and thus were easily within range so the network would not loose time synchronization lock. We elaborate on synchronization lock in Section 6.

6. Future Work and Conclusions

Our approach's time synchronization resolution is limited by the frequency of Time-Code transmissions. The frequency of Time-Code transmissions is limited by the requirement of sufficient time for the broadcast system time message to propagate throughout the network. We believe it is possible to decouple the need for a one-to-one correlation of system time messages and Time-Code transmissions to obtain an improved synchronization error. However, the theoretical upper limit of system time synchronization precision is limited by the latency and jitter inherent in SpaceWire Time-Code function. Time-Code enhancement techniques [7] could be incorporated into our time distribution approach to improve time synchronization.

Additional features will be incorporated into the OPNET Models to expand the representation of the SpaceWire protocol and the nodes. Specifically a model of Remote Memory Access Protocol (RMAP) for SpaceWire will be developed. RMAP provides a standard method of reading and writing to registers and memory across a SpaceWire network. This will further our analysis capability of application performance.

Additionally, we have developed a Live/Virtual/Constructive capability at Sandia [6] that combines real devices, emulated devices, and simulated devices in a single hybrid experiment. We have identified use cases in our SpaceWire development activities that will benefit from merging our SpaceWire models into hybrid experiments to assess satellite network development ideas at various stages of the development. This approach is expected to support assessing the behavior of actual hardware prior to the availability of complete system hardware.

We have demonstrated a viable system distribution approach that can be employed without modification to the SpaceWire standard. The time distribution approach has been modeled in a high-fidelity simulator and our analysis has identified the range of time synchronization for various SpaceWire network architectures. Our broadcast solution has been fully developed in VHDL and tested in actual custom hardware. We continue with further integration and testing in actual hardware in our development activity.

7. REFERENCES

[1] European Space Agency, "SpaceWire - Links, nodes, routers, and networks." *ESA-ESTEC Requirements & Standards Division*. 24 January 2003.

[2] Klar, R., Dykes, S., Bertrand, A., Mangels, C., "Integration of Internet Protocols with SpaceWire using an Efficient Network Broadcast." *International SpaceWire Conference 2007.*

[3] Parkes, S., "The Operation and Uses of the SpaceWire Time-code." *ISWS International SpaceWire Seminar 2003.* November 2003.

[4] Roberts, A., Dykes, S. G., Klar, R., & Mangels, C. C. (2007, March). A Link-Layer Broadcast Service for SpaceWire Networks. *Aerospace Conference, 2007 IEEE* , 1-10.

[5] OPNET Technologies, Inc., www.opnet.com.

[6] Van Leeuwen, B., Urias, V., Eldridge, J., Villamarin, C., Olsberg, R., "Performing cyber security analysis using a live, virtual, and constructive (LVC) testbed," *IEEE MILCOM 2010,* October 2010.

[7] Cook, B., "Reducing SpaceWire Time-code Jitter." www.4links.co.uk/bibliography/Reducing-Time-Code-Jitter-on-SpaceWire.pdf , October 2003.