

# High Speed Shared Memory Networks in Hardware in the Loop Applications

Troy Troshynski  
Avionics Interface Technologies  
(A Division of Teradyne)  
Omaha, NE U.S.A.

*Abstract* - As aircraft avionics systems become more and more integrated and complex, the overall costs and duration of system verification and validation efforts continue to increase and contribute to the rising costs of new system design. In order to control costs, reduce schedules, and improve overall quality, Distributed Hardware-in-the-Loop (HIL) simulations are increasingly being used to verify integrated modular avionics systems. As these systems become increasingly complex, the distributed system must interface with an ever increasing number of aircraft interfaces, sensors, and actuators which often drives the requirement for a distributed, real-time processing architecture for the HIL system.

This paper provides a brief technical overview of distributed HIL test systems and also explores the use of a high speed Shared Memory Network for data sharing and time synchronization between real time processors in a distributed HIL test system.

## INTRODUCTION

Avionics applications have become a network of integrated modular systems with increasingly more importance on the network used to move data between each subsystem. Generic maintenance communications, high-bandwidth video data, and event-driven real-time data must be integrated into the avionics communications network.

Initially, most avionics systems relied on specialty hardware busses and proprietary interfaces to integrate modular systems. Standard bus architectures like MIL-STD-1553 in military applications and ARINC-429 in commercial applications were the norm. However, as the systems have become more distributed, these busses were simply not fast enough or scalable enough.

The Ethernet commercial standard (IEEE 802.3) slowly began to be used as the backbone for communications in avionics systems. Ethernet is the defacto-standard that has been in

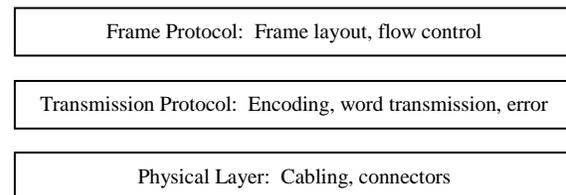
use for distributed commercial applications for decades. It is low cost, and a proven technology for distributed systems applications. It is a natural progression for Ethernet to become the backbone in distributed, integrated modular avionics systems.

Although Ethernet offers many advantages, there are specific avionics applications that Ethernet does not apply itself well to. These applications still require networked distribution of data but require other attributes that Ethernet is not very good at. These attributes include:

- High-volume Data Movement
- Deterministic frame transmission and reception
- Redundancy
- Real-time Event-driven data transfers

For these types of network applications, specialty subnetworks are available to allow communication between the subnetwork and the main Ethernet backbone.

There are quite a few standardized subnetworks that are well defined and in-use today. For example, the ARINC-664 Network Specification defines a network to accommodate deterministic, dual redundant switched network. ARINC-664 includes redundancy and deterministic behavior that is required for distributed Avionics network applications.



*Figure 1 – SMN Layers*

The ARINC-664 standard is very similar to the standard IEEE802.3 Ethernet except that a unique identifier is contained in the MAC layer called a Virtual Link. The Virtual Link is used

by the network switch for the physical connections, and data routing needed to pass data between devices. The use of custom ARINC-664 switches ensures that frames are routed in a deterministic manner, and dual-redundant end-systems ensure reliability of delivery. ARINC-664 has been adopted by Boeing on its 787 commercial aircraft program and Airbus on its A380, A350 commercial aircraft programs.

ARINC-664 and other similar Ethernet derivative standards, like AS6802, are subnetworks with specific real-time attributes, but they lack the capability of real-time event driven transfers. In hardware in the loop applications, event driven transfers are key. For deterministic, event driven networks, the Shared Memory Network (SMN) is the appropriate choice.

### **Shared Memory Network Overview**

There are three reasons why the Shared Memory Network (SMN) is a great fit as a subnetwork for tightly coupled event driven avionics applications:

- Simple to implement
- Fast and scalable
- Low latency with no protocol overhead

SMN is a simple network where data written to memory locations on their local node appears in the RAM on all of the other nodes, and that writes on the other nodes will also appear in their RAM attached systems. All data associated with any local event is shared quickly with other nodes on the network. Interrupts associated with unpredictable events can be sent to specific nodes on the network, or broadcast. The implementation is very simple, All data is sent at real-time rates close to the 2.125Gbps line speed with virtually no overhead. Because of the small overhead, there is minimal latency associated with data transfer over the network.

SMN's limited protocol can be defined as three (3) functional layers. These layers are illustrated in Figure 1.

#### **Simple to Implement**

The Shared Memory Network is based on reflective memory technology. This is a method where multiple systems in a network share data. The network shared data operates like a dual ported memory. Each Shared Memory Network

node is 256 Mbytes. The network operation is very simple, when an end-system writes data to a memory location on its SMN node, this data is replicated (or reflected) on all of the other nodes on the network at the very same memory location.

The SMN is an optical ring topology where every End-System is a data source, and a Receiver. There is no routing fabric, all data is broadcast immediately when written to a node. The physical interface for a SMN port is a fibre optic connection. Each fibre optic port is simplex, unidirectional, serial connections. One line for transmit, another for receive. Fibre optic cables are lightweight, and easy to maintain. There are no large multi-pin cable bundles and connectors. Either single mode or multi-mode optical cabling can be used. The choice of media type is dependent upon distance, power requirements, and weight. The different physical media options, with no degradation of performance, is a real benefit to the Avionics design engineer.

#### **Fast and Scalable**

Shared Memory Network is extremely fast. Currently, 2.125 Gbps network speeds are all in use today. All data transfers occur at line speeds over the network. The roadmap exists for increasing speeds to 4, 8 or 16 Gbps in future implementations. High-bandwidth applications will scale upward to access the full line bandwidth without changing the application. Adding additional End-Systems into an existing network is as easy as just adding nodes to the SMN network ring topology. There is no network address re-configurations, routing table re-configurations, or network switching changes that are required. Avionics applications can simply add nodes and re-define the application for sharing the 256 Mbytes memory.

#### **Low Latency with no protocol overhead**

The Shared Memory Network implements a protocol optimized for low latency transmission of data. All data transmissions are broadcast at network line rates (2.125 Gbps). Each node uses the same small defined protocol (how to move the data on/off the network). However, the user application never has to account for protocol or message delivery. Each node application simply reads/writes to the shared memory. This enables the networked system to operate as a tightly coupled application.

The SMN is analogous to a multi-threaded application that shares a common system memory. Each thread operates autonomously, while the applications decide on the partitioning and use of the shared memory. In addition, similar to multi-threaded applications, the SMN supports the use of interrupts. Therefore, SMN nodes can generate interrupts for other nodes on the network, and pass critical data along with the interrupt. This is critical for event driven hardware-in-the-loop and critical real-time applications where an event must be handled immediately by a node in the network.

### SMN Operation

The Shared Memory Network consists of nodes connected together in a ring topology. Each network node interface uses a Transmit queue for sending node data, and a Re-transmit queue for re-sending data received over the network. Any node can write to the local Shared Memory RAM, these writes are packetized into small frames, and placed in the transmit queue that inserts the frames into the network traffic during idle times. The queue consists of one (1) to sixteen (16) 32-bit words that can be transmitted at a time. In addition to the data, there is a two (2) word header with information about the following data (like node transmit ID, memory index of the data, and number of data words). There is a checksum word that follows datawords being sent. Once placed in the transmit queue, this data is sent over the ring to the next node.

Each node has a re-transmit queue that receives all incoming data from the network. The Re-transmit queue determines if the message received is from itself. If it is the originator of the message, then the message is cleared from the queue. If the message is from another node, then the Shared Memory information is deposited locally, and pending no other messages in the Transmit queue, it is re-transmitted immediately on the network. See Figure 2.

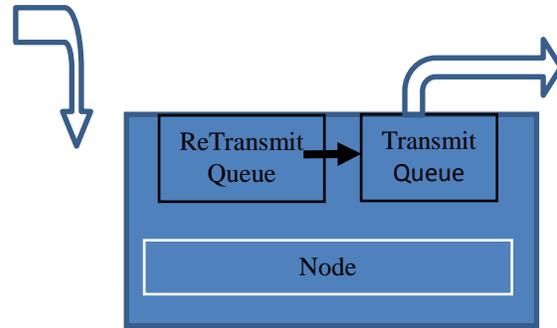


Figure 2 – SMN Node

As stated previously, the Shared Memory Network protocol is simple. SMN, like any network architecture, transmits blocks of user information called *packets*. Before sending the actual data over the physical link, additional specific control information is added to the packet. The combination of the control information and the data is called a *frame*. For SMN a frame has two 32-bit words of control information (header) and anywhere from 1 to 16 32-bit words of payload (pure data containing the user information). Likewise, every frame is followed by an End of Frame delimiter (EOF) which is a 32-bit word signaling the end of the frame and a checksum for packet integrity checking. If the user data required to send is too large to send in one frame, then multiple frames must be grouped together and this forms a sequence.

Besides frames of data, SMN also identifies an idle primitive sequence data to be sent over the link. This type of data is SMN’s mechanism to perform control and signaling over the simplex transmission link.

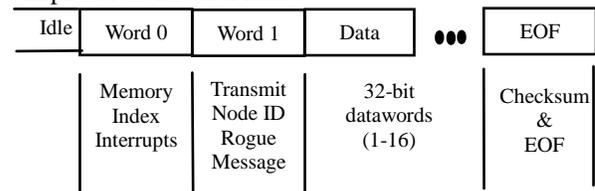


Figure 3 – SMN frame Format

Preceding any frame of data, there must be a minimum of 1 IDLE fill words. Figure 3 shows the Fibre Channel frame format.

### SMN Simulation

Deterministic Data transfers of small amounts of data (4 to 64 bytes) are critical for simulation and hardware in the loop avionics systems applications. A Hardware in the Loop

application must get data to another systems reliably and on-time. Prior to implementation, it is important to completely understand the behavior of each individual component and the whole system during normal operations, as well as behaviors and reactions to system faults. Understanding the movement of data over the network as well as understanding the amount data that is being passed is critical. Setting up the simulation will occur in two phases:

- Setting up and Testing individual End Systems
- Testing End Systems in the Network

### Individual End System Testing

Avionics Integrated Modular Systems are delivered to the airframe manufacturer from many suppliers. Each Modular System specializes in a particular function for the aircraft (i.e. Navigation, Collision Avoidance Radar, Health Maintenance). Individual testing is required for the unique System as well as simulating the other End-Systems. Therefore, an SMN must be able to simulate missing data sources in order to stimulate the Unit Under Test. Testing associated with individual Modular End Systems include:

- Does the Source data End System ports transmit correct at appropriate sync intervals?
- How many transmit errors are produced over a long time period?
- Does the Modular End System receive and process objects correctly?
- How does the Modular End System react to receiving incorrect Objects, frames with protocol errors, CRC errors, or frames with bad data in them?

### Network System Tests

Individual End Systems are eventually integrated into the Modular avionics network. Effective testing is required during integration to insure that each End System operates efficiently. This type of testing requires that test equipment is placed in-line between the data source, and the recipient. Network frame flow testing reveals flaws in the system design and implementation. Testing associated with the Integrated network include:

- Monitoring each End System for frame errors associated with frame movement over the network.

- Checking the deterministic flow of containers from the End System source.
- Checking the redundancy of the network (if implemented).

### SMN Simulation Example

A good example of a Hardware in the Loop application that requires an SMN is a vehicle self-preservation system. The typical self-preservation system requires three functions

- Threat detection
- Analysis of options (computing)
- Avoidance and/or Fire Control System

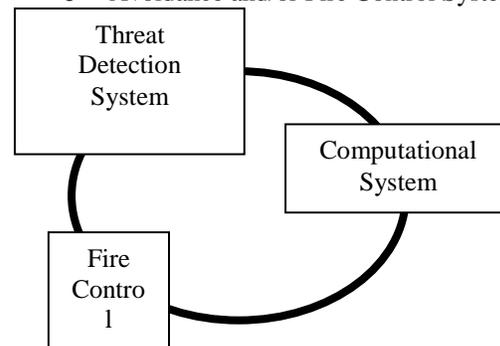


Figure 4 – Self Preservation System

In this example, the Threat detection and Avoidance/Fire Control functions are associated with hardware sensors that will either collect data, or control devices used for avoidance/fire control. A computational element performs the calculations associated with analyzing the threats and determining the best course of action. This simple modular architecture splits easily into the three functions required.

A self-preservation system of an armored vehicle best demonstrates the criticality of a deterministic network. The self-preservation system includes a system for monitoring sensors that detect threats, a computational system, and a weapons system that will eliminate the threat rather than avoid it. Each of these systems are not co-located within the vehicle, so a network interface instead of a bus architecture, is best suited to connect these modular functions. I will show how standard IEEE 802.3 Gigabit Ethernet and Gigabit ARINC-664 can be used versus SMN as the network that best fits this requirement.

### Ethernet and ARINC-664

In our example, the armored vehicle proceeds at a variable speed along a path.

Initially, only low volume maintenance communications occurs between the three modular systems. When the Threat detection system obtains information from its sensors that a threat has been detected. For example, a hand-held missile system operating within firing range of the vehicle is detected. Only a couple of seconds are available to acquire the threat, analyze options and fire the weapon to eliminate the target (or possibly communicate with another system to avoid the incoming threat).

The Threat detection system must interrupt the routine maintenance network traffic and immediately report its threat information to the Computational system. The amount of data reported by the Threat detection system would be small, typically less than 64 bytes indicating position and range information. Additional display information might follow, which is more data.

In a standard Ethernet network, the Threat detection system would send this information out as an Ethernet packet. If there is not existing traffic on the network that might cause a collision (which would delay the data), the packet could be sent without any delay, but any traffic on the network would cause an unknown delay.

In the ARINC-664 network (deterministic, time division multiplexed with defined latency and jitter), the Threat detection system would send the information out as an Ethernet packet. Since an ARINC-664 network assigns each Virtual Link a Bandwidth Allocation Gap (BAG) time which guarantees no collisions on the network with other traffic, the packet could be sent with a maximum latency of 1 millisecond after receipt of the threat data.

In addition, both Ethernet and ARINC-664 must transfer a minimum of 64 bytes in a frame. In our application only a small amount of data, 16 bytes, is required (notably location information consisting of latitude and longitude). Despite the small data size, a 64-byte frame must be sent and processed by the receiving node.

After the computational system performs its analysis, its information must be sent over the network to the fire control system. The Fire Control system will use the information to direct the anti-missile fire at the oncoming projectile. The transfer of information between the Computational system and the Fire Control system must be completed as quickly as possible. Here again, using standard Ethernet for this network transfer, delays can occur due to collisions on the network, and possible protocol

processing associated with the send and receipt of Ethernet packets. Likewise, if we substitute an ARINC-664 network to take advantage of its deterministic behavior, it would perform better than standard Ethernet, but there are still delays associated with BAG latencies.

### **Shared Memory Network**

By contrast, a Shared Memory network is a better architecture for our Hardware in the Loop application. The reasons that the SMN will perform better is its ability to:

- Generate an immediate interrupt on another system
- Transmit data at full line speed (2.125 Gbps)
- Eliminate protocol processing, so interrupted applications can begin processing data in accessible memory.

In the example application, the Threat detection system can generate an interrupt on the SMN to the Computational system along with sending the threat data. There is no risk of collision on the network, since all node output data takes priority over anything being received. This is a key advantage and why the SMN meets event-driven application requirements. Upon receipt of the interrupt, the Computational system can act immediately upon the Threat data stored in its memory. Then after performing the computations, it can immediately interrupt the fire control system with data required for elimination of the incoming threat.

### **CONCLUSION**

Shared Memory Network (SMN) is a deterministic, event-driven network technology that is designed for real-time simulation applications. The SMN is based on reflective memory ring network architecture. The SMN defines a low latency protocol that defines the movement and mapping of data in shared memory among network systems. The SMN is optimized to move high-bandwidth, low latency data, reliability and with determinism. These are key requirements for most Avionics simulation systems.

Most simulation and hardware in the loop applications need event driven movement of data with event driven interrupts. The SMN meets these requirements and it is simple to implement and scalable. In addition, SMN will meet extremely tight timing requirements within the

network. The efficiency and availability of the data traversing the SMN network to applications is far better than standard Ethernet or Deterministic Ethernet. SMN is an ideal sub network for hardware in the loop applications.

## References

- [1] "Reflective Memory", From Wikipedia, the free encyclopedia  
[https://en.wikipedia.org/wiki/Reflective\\_memory](https://en.wikipedia.org/wiki/Reflective_memory).
- [2] Avionics Interface Technologies, "*Shared Memory Network Interfaces*",  
[www.aviftech.com](http://www.aviftech.com)
- [3] Avionics Interface Technologies, "*SMN Hardware Manual*", 2015
- [4] GE Fanuc., "Real-Time Networking with Reflective Memory," [www.geautomation.com](http://www.geautomation.com).
- [5] Curtiss Wright, "SCRAMNet+ Shared Memory – Speed, Determinism, Reliability, and Flexibility for Distributed Real-Time Systems," [www.cwcdefense.com](http://www.cwcdefense.com).

© 2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.