Extended Abstract for AIAA SPACE 2015 Conference, 8/31/2015 – 9/2/2015

TTEthernet for Integrated Spacecraft Networks

Andrew Loveless, NASA Johnson Space Center, Houston, TX 77058

Aerospace projects have traditionally employed *federated* avionics architectures, in which each computer system is designed to perform one specific function (e.g. navigation). There are obvious downsides to this approach, including excessive weight (from so much computing hardware), and inefficient processor utilization (since modern processors are capable of performing multiple tasks). There has therefore been a push for *integrated modular avionics* (IMA), in which common computing platforms can be leveraged for different purposes [1]. This consolidation of multiple vehicle functions to shared computing platforms can significantly reduce spacecraft cost, weight, and design complexity. However, the application of IMA principles introduces significant challenges, as the data network must accommodate traffic of mixed criticality and performance levels – potentially all related to the same shared computer hardware. Because individual network technologies are rarely so competent, the development of truly integrated network architectures often proves unreasonable. Several different types of networks are utilized – each suited to support a specific vehicle function.

Critical functions are typically driven by precise timing loops, requiring networks with strict guarantees regarding message latency (i.e. determinism) and fault-tolerance [2]. Alternatively, non-critical systems generally employ data networks prioritizing flexibility and high performance over reliable operation. Switched Ethernet has seen widespread success filling this role in terrestrial applications. Its high speed, flexibility, and the availability of inexpensive commercial off-the-shelf (COTS) components make it desirable for inclusion in spacecraft platforms. Basic Ethernet configurations have been incorporated into several preexisting aerospace projects, including both the Space Shuttle and International Space Station (ISS) [3]. However, classical switched Ethernet cannot provide the high level of network determinism required by real-time spacecraft applications. Even with modern advancements, the uncoordinated (i.e. event-driven) nature of Ethernet communication unavoidably leads to message contention within network switches. The arbitration process used to resolve such conflicts introduces variation in the time it takes for messages to be forwarded.

TTEthernet[1] introduces decentralized clock synchronization to switched Ethernet, enabling message transmission according to a time-triggered (TT) paradigm. A network planning tool is used to allocate each device a finite amount of time in which it may transmit a frame. Each time slot is repeated sequentially to form a periodic communication schedule that is then loaded onto each TTEthernet device (e.g. switches and end systems). Each network participant references the synchronized time in order to dispatch messages at predetermined instances. This schedule guarantees that no contention exists between time-triggered Ethernet frames in the network switches, therefore eliminating the need for arbitration (and the timing variation it causes) [4].

Besides time-triggered messaging, TTEthernet networks may provide two additional traffic classes to support communication of different criticality levels. In the rate-constrained (RC) traffic class, the frame payload size and rate of transmission along each communication channel are limited to predetermined maximums. The network switches can therefore be configured to accommodate the known worst-case traffic pattern, and buffer overflows can be eliminated [5]. The best-effort (BE) traffic class behaves akin to classical Ethernet. No guarantees are provided regarding transmission latency or successful message delivery [6]. TTEthernet coordinates transmission of all three traffic classes over the same physical connections, therefore accommodating the full spectrum of traffic criticality levels required in IMA

[1] TTEthernet is the further commercial development of the TT-Ethernet research jointly conducted between the Vienna University of Technology and TTTech Computertechnik AG [6].

architectures. Common computing platforms (e.g. LRUs) can share networking resources in such a way that failures in non-critical systems (using BE or RC communication modes) cannot impact flight-critical functions (using TT communication). Furthermore, TTEthernet hardware (e.g. switches, cabling) can be shared by both TTEthernet and classical Ethernet traffic.

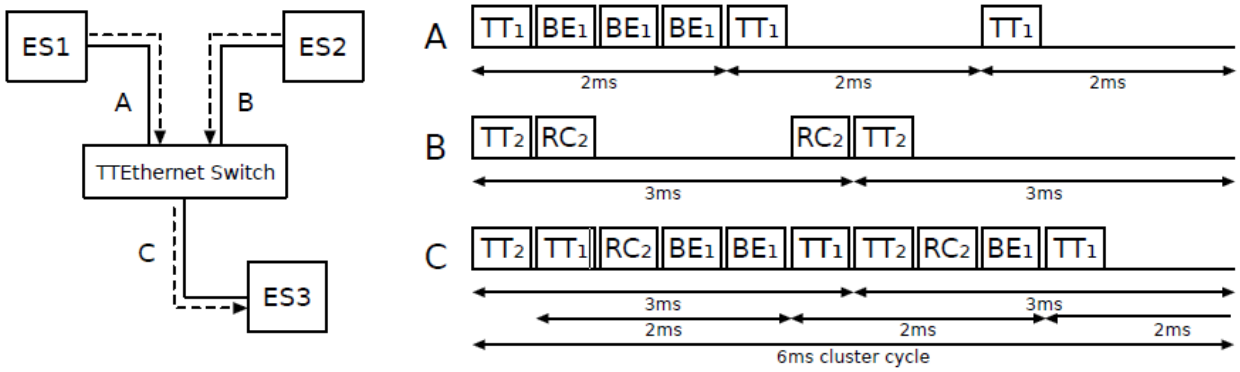

Figure 1. Integrated dataflow in a sample TTEthernet network

The Avionics and Software (A&S) project, chartered by NASA's Advanced Exploration Systems (AES) program, is exploring the application of the TTEthernet technology to future spacecraft architectures. TTEthernet is used in the second iteration of a network-based failover architecture showcased in the Integrated Power, Avionics, and Software (IPAS) facility at Johnson Space Center (JSC). Real-time failover occurs between two redundant flight computers running Core Flight Software (CFS) in a simulated Asteroid Redirect Mission (ARM). To show the versatility of the TTEthernet-based failover method, dissimilar processors are used in the redundant flight computer set. One flight computer is realized on a standard Linux PC running CentOS 6.5 64 bit. The second flight computer, a Proton400k-L, incorporates a Freescale dual-core PowerPC processor in a radiation hardened 3U single board computer (SBC) platform.

Several different software applications were developed to aid in the configuration of TTEthernet networks and in the visualization of their operation. A TTEthernet network interface status monitor was created to provide real-time feedback regarding the condition of any installed TTEthernet controllers. A second application was developed to let a user change a local network adaptor's schedule configuration from a straightforward graphical interface. This separates the management of a device's communication schedule from the software actually used to send and receive TTEthernet messages. Within the context of a redundant flight computer setup, it allows both computers to run identical flight software loads while still communicating according to different TTEthernet schedules. A third program was developed to monitor and graphically display the flow of all critical traffic (TT or RC) within a TTEthernet network.

The implementation of TTEthernet-based failover requires integration of the TTEthernet library/API with the flight software running on each flight computer. CFS provides a mission independent environment offering core services designed to insulate developers from underlying hardware and software dependencies. The long-term goal of CFS's integrated core flight executive (cFE) is to facilitate the growth of a reusable bank of flight software applications [7]. The integration of TTEthernet with CFS is therefore important not just for this failover application, but also so it may be leveraged in future spaceflight projects. A shared library was developed to provide compatibility between CFS and the TTEthernet network adapters. The library allows any CFS application to access core TTEthernet capabilities, including the ability to send and receive mixed criticality messages. The flight software

interfaces that had previously used UDP/IP messaging to communicate with the simulation were rewritten to use the new TTEthernet library.

Two methods were developed to enable failover between the redundant flight computers – one based on periodic heartbeat messages, and the second using coupled data pairings. Both approaches utilize a traditional failover methodology, in which only one flight computer controls the vehicle and drives the simulation at a time. In the first failover method, the primary computer sends regular heartbeat messages to the backup machine to indicate that it is still functioning. If the backup stops receiving heartbeats, it assumes that the primary machine has failed and takes control of the vehicle. Because both flight computers receive the same data from the simulation simultaneously, the exchange of no other state information is necessary. The second failover method better leverages TTEthernet's capabilities to increase performance and eliminate the need for a periodic heartbeat. Both flight computers still receive data from the simulation simultaneously, but are no longer divided into static primary and backup roles. Because no distinction exists between them, each machine processes information and sends effector commands back to the simulation as if it is the only computer flying the vehicle. Both messages are sent back such that the simulation always receives them in paired sets. The receipt of both commands composes one single command cycle. One computer is designated as primary in each cycle, and data sent by the backup is discarded. If no command from the primary computer is included in the received data set, the backup computer takes the primary designation, its command is processed, and the mission continues uninterrupted.

Figure 2 and Figure 3 depict the integrated failover architecture as showcased during the A&S project's end of FY14 Integrated Test. In addition to the capabilities described above, the flight computers also communicate with other vehicle subsystems using standard Ethernet interfaces. The active flight computer outputs mission status information encapsulated in UDP packets. This data is transmitted in classical Ethernet frames through the TTEthernet switches, into the Bay 1 network, and to a touchscreen crew display. Additionally, an ECLSS simulator in Bay 2 of the IPAS facility broadcasts data such as partial pressure and ambient temperature to both flight computers. This data is relayed as UDP traffic over a standard Ethernet network, through a router connecting the Bay 2 and Bay 1 networks, and into the TTEthernet switches. In both cases, the UDP packets travel through the same physical cabling as the time-triggered messages in the flight control loop. A separate machine running Engineering DOUG Graphics Environment (EDGE) software receives data from the simulation and displays a high fidelity 3D representation of the mission as it progresses.

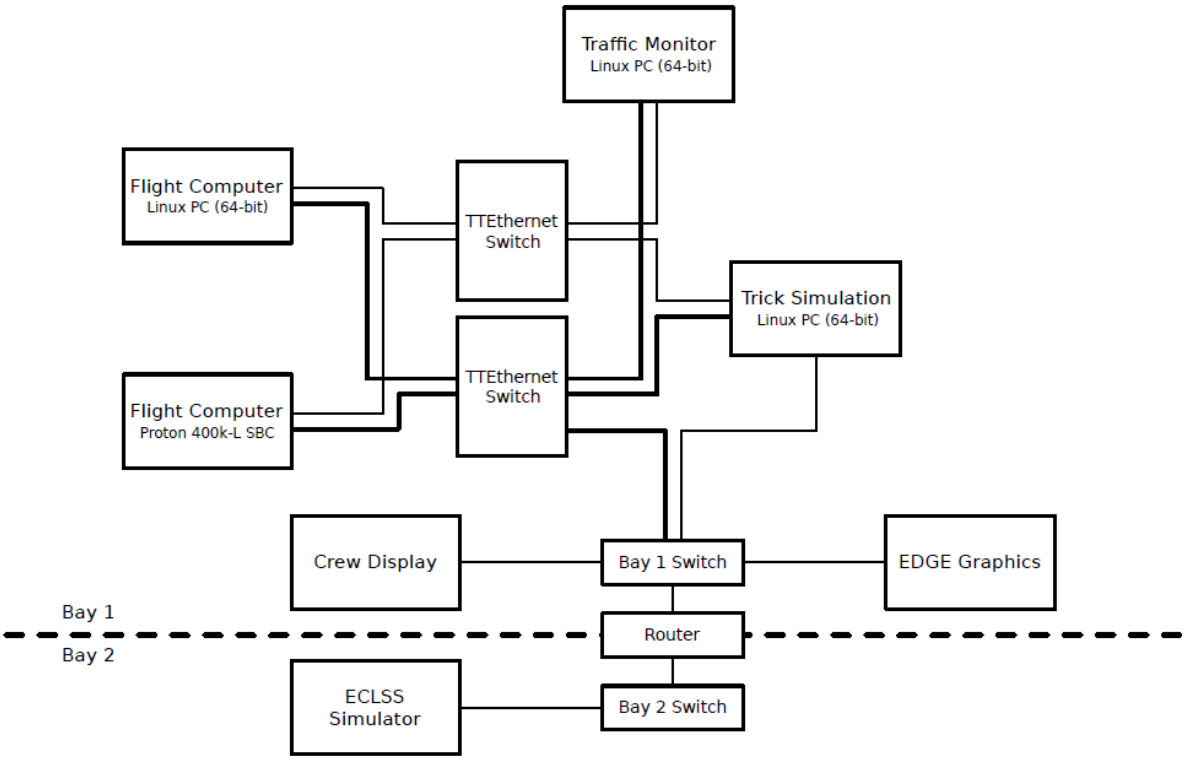

Figure 2. Final integrated failover architecture in IPAS facility (physical connections)

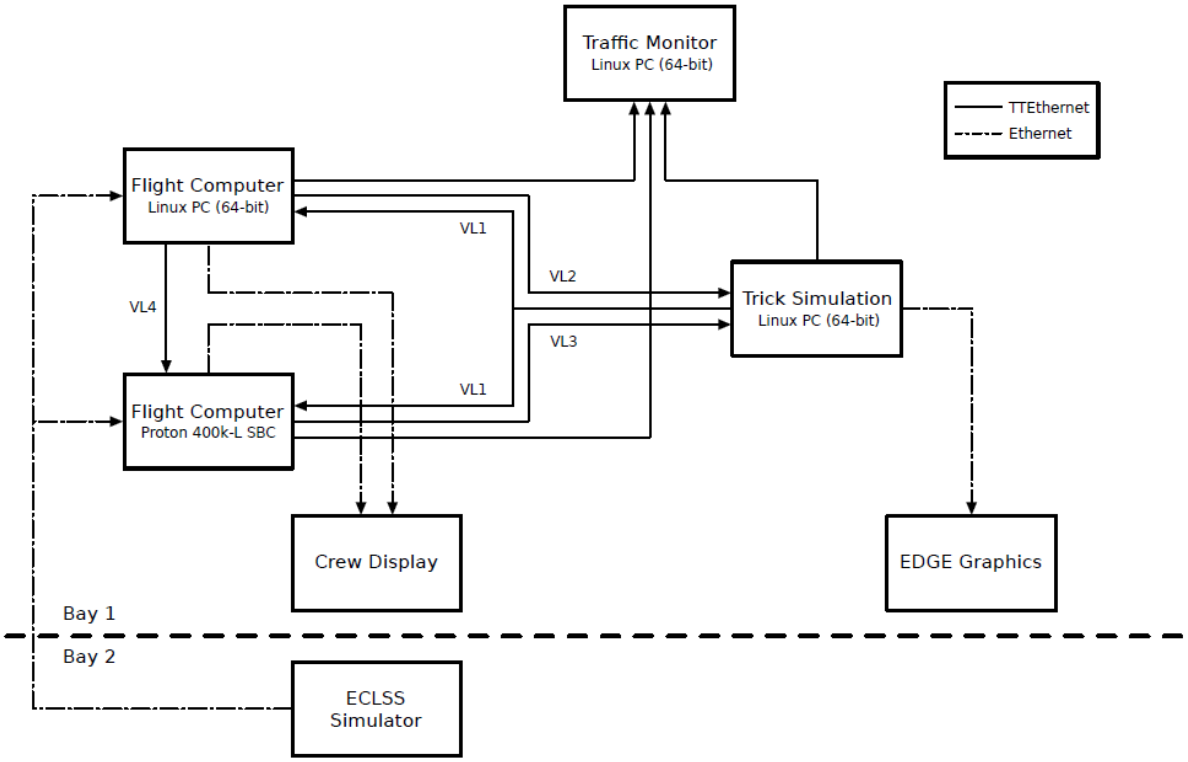

Figure 3. Final integrated failover architecture in IPAS facility (data flow)

References

[1] Rushby, J., "Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance," Tech. rep., NASA Langley Research Center, Hampton, VA, June 1999.

[2] Alena, R., Ossenfort, J., Laws, K., Goforth, A., and Figueroa, F., "Communications for the Integrated Modular Avionics," *Proc. Aerospace Conference, 2007 IEEE,* March 2007.

[3] Goforth, M., Ratliff, J., Hames, K., and Vitalpur, S., "Avionics Architectures for Exploration: Building a Better Approach for (Human) Spaceflight Avionics," *Proc. SpaceOps 2014 International Conference on Space Operations,* Pasadena, CA, May 2014.

[4] Steiner, W., Bauer, G., Hall, B., and Paulitsch, M., "Time-Triggered Ethernet," *Time-Triggered Communication,* Aug. 2015.

[5] Steiner, W., Bonomi, F., and Kopetz, H., "Towards synchronous deterministic channels for the Internet of Things," *Proc. Internet of Things (WI-IoT), 2014 IEEE,* Seoul, Republic of South Korea, March 2014.

[6] Steiner, W., and Paulitsch, M., "Time-Triggered Ethernet," *Industrial Communication Technology Handbook,* Boca Raton, FL 2nd ed. 2015.

[7] "Core Flight System (CFS): Development Standards Document," July 2011.