ACCELERATING STANDARDS COMPLIANT TMNS RADIO IMPLEMENTATIONS

Todd A. Newton, M. Wayne Timme

Southwest Research Institute®
San Antonio, Texas
todd.newton@swri.org, wayne.timme@swri.org

ABSTRACT

IRIG 106-17 defines interoperable two-way network telemetry interfaces for the wired as well as the dynamic TDMA air interface. While the air interface is based on the familiar SOQPSK-TG waveform, a TmNS-based radio contains a dynamic TDMA MAC regulated by Link Management through the use of RFNMs. This paper illustrates the TmNS-based radio aspects of the IRIG standard by describing our experience utilizing a two-track approach for accelerated TmNS compliant radio development. We have divided the architecture by engineering discipline lines (Communications vs. Computer Engineering). Doing so allowed us to accelerate the design, simulation, and test tasks while using a common code base across various transceiver implementations. Discussion includes a description of the software modules that provide TmNS interfaces for standards compliant radio functionality such as the TDMA MAC, RFNM processing, system management, and MDL configuration as well as system-level integration testing.

KEYWORDS

Telemetry Network Standard (TmNS), iNET, IRIG 106-17

INTRODUCTION

Standards are vital to interoperability amongst components, particularly when components come from multiple vendors.

The IRIG 106-17 standards [1] provide the framework for designing and deploying system of systems networked-based telemetry. However, if the standards are not precise enough, different interpretations can lead to system implementations that are not interoperable. Often, a de facto standard reference implementation arises to alleviate these types of challenges. This leads to a

logic question: what examples of standards do we have that have dealt well with these challenges.

Use of the IETF standards as the springboard and model for the IRIG 106-17 standards.

Since its inception, the integrated Network Enhanced Telemetry (iNET) program has leveraged existing, successful, standards. Faulstich, while providing a short history of the program quoted the mantra, "to boldly go where everyone else has gone before [2]." This indeed happened in IRIG 106-17. In fact, a large portion of this IRIG standard points directly at IETF standard sections. But, is this enough? What came first? The IETF standards or an implementation? What was used by programmers in order to build interoperable networking applications? The source code of Berkeley's OSI stack or the IETF's RFC that standardized implementations. A careful, reading between the lines (or timelines) chapter 2 of Raymond's famous text, "The Art of Unix Programming" [3] not only discloses the sordid history of the "TCP/IP Wars," but also makes it clear that some of the IETF's success in itself, was not only going where others had gone before, but also using their codebase as an example.

In the telemetry world, it is no different. The IRIG 106-17 standardizes two-way network telemetry as part of the Telemetry Network Standard (TmNS) contained in Chapters 21-28. Though standardized, no de facto implementation exists to serve as a reference for the required interfaces. If such an implementation did exist, it would pave the way for additional TmNS-compliant radios to enter the market easier and with a high level of confidence to be interoperable.

Like the IETF, IRIG 106-17 success needs example implementations.

This paper discusses an approach to providing a clear example that implements the standardized network interfaces and protocols up to the hardware transceiver of an IRIG 106-17 compliant TmNS radio.

DOES THE WORLD REALLY NEED MORE RADIOS?

There are several radio products on the market today, many of which operate in C-band. TmNS radios, however, are more than the average C-band transmitter. By design, the TmNS radios are two-way radios, allowing both sending and receiving of packetized transmission bursts. At their core, the TmNS radios operate as IP routers that maintain one or more RF interfaces in which the transmission bursts are associated with. Most other transmitters on the market today utilize a continuous pulse-code modulation (PCM) stream rather than the more asynchronous approach provided by the TmNS radios.

As with the release of any new standard, initial implementations tend to be limited in number. The same is true for TmNS-based radios. As part of the iNET project, a small number of hardware radio prototypes from a single vendor were built for use in the initial iNET flight tests. A few of these radios are located in the iNET System Integration Lab (iSIL) hosted at the

Southwest Research Institute® (SwRI®) in San Antonio, TX. In the iSIL they are used as part of the overall system-level testing of TmNS-based systems. While having real hardware in the lab to perform this testing is great, having only a single implementation of a TmNS radio does not allow for true interoperability testing with other implementations. Multiple interoperable implementations of TmNS radios are needed in order conclude that the IRIG 106-17 chapters pertaining to TmNS radios are complete and have the right level of detail.

TMNS RADIO EMULATOR

The initial need for a TmNS-compliant radio emulator became evident to us during the development of other TmNS-compliant components. SwRI is currently developing reference applications for the Link Manager and System Manager applications. One challenge faced during development of these tools is that their operation is highly dependent upon communication with other devices. The Link Manager provides bandwidth sharing of a single frequency by operating as a time-division multiple access (TDMA) scheduler for up to 10 pairs of TmNS-compliant radios concurrently, basing the number and duration of each TDMA transmission slot on each radio's current network traffic backlog and the priorities of the missions being supported by these radios. Without any TmNS-compliant radios to test with, the Link Manager application and its scheduling algorithms could not be fully tested. Furthermore, even when the hardware radio prototypes in the iSIL are updated to meet the IRIG 106-17 standard, there are not enough of them in order to test the system performance and scalability requirements. With the need established, we decided upon a course of action that would provide the management interfaces and protocols that a TmNS radio would support with respect to Link Manager and System Manager communication.

Other organizations have spent a great deal of effort providing radio simulators in order to test the scalability of the Link Manager to manage 40+ TmNS radios. While they had some merit, the scope of these efforts was too narrow. These simulators only exchanged messages with the Link Manager, but they did not support real interaction with any other components in the system. Thus, the System Manager application could not monitor the radio simulator for health and status information like a real TmNS radio. The data flows used by the simulators were generated by the radio simulator nodes themselves (or virtual machines running alongside the simulator node) rather than being received from other devices on the network and being routed through the simulator. They did not perform any IP routing function like the real TmNS radios do. Because of these limitations, these radio simulators had no value in terms of end-to-end system level testing within the iSIL and were not used.

What was really needed was a TmNS radio emulator that would mimic the behavior of a real TmNS radio. With no workable solution to emulate TmNS radios available, SwRI developed one on their own. Apart from a real RF interface, the emulator functions as a real TmNS radio is expected to function. The TmNS radio emulator has been built on a small platform running Linux. It contains an SNMP interface with support for the TMNS-MIB as used by the System Manager application. It also contains an RF Network Message (RFNM) Processing Engine for exchanging RFNMs with the Link Manager over a TCP connection. The emulator provides periodic queue status and link metrics to the Link Manager over the TCP connection, and it

receives from the Link Manager its transmission schedule in the form of Transmission Opportunity (TxOp) assignments. The emulator supports all Type-Length-Value (TLV) triplets defined in IRIG 106-17 Chapter 24 for RFNMs, most of which play critical roles in the dynamic scheduling algorithms utilized by the Link Manager.

The radio emulator provides the TmNS Source Selector (TSS) server capability. This allows for TSS connections to be established between the radio emulators and the Link Manager application. The TSS connections serve as secure tunnels as well as support seamless antenna-to-antenna handoff scenarios.

The radio emulator operates as an IP router. Using classic Linux tools, data being routed through the emulator is categorized and placed in appropriate TE Queues while waiting to be transmitted out the emulated RF interface. When the transmission schedule allows, data is pulled from the TE Queues and sent out the RF interface.

TECHNICAL APPROACH TO TMNS RADIO DEVELOPMENT

The different interfaces of the TmNS radio emulator were initially built in a modular approach in order to allow testing of specific interfaces without being dependent upon other unrelated interfaces. As various modules were added to the emulator's suite of interfaces and supported protocols, it became apparent that the suite could very quickly become the basic building blocks of a real TmNS-compliant radio. These modules can be seen in Figure 1 and are further discussed in the subsections below.

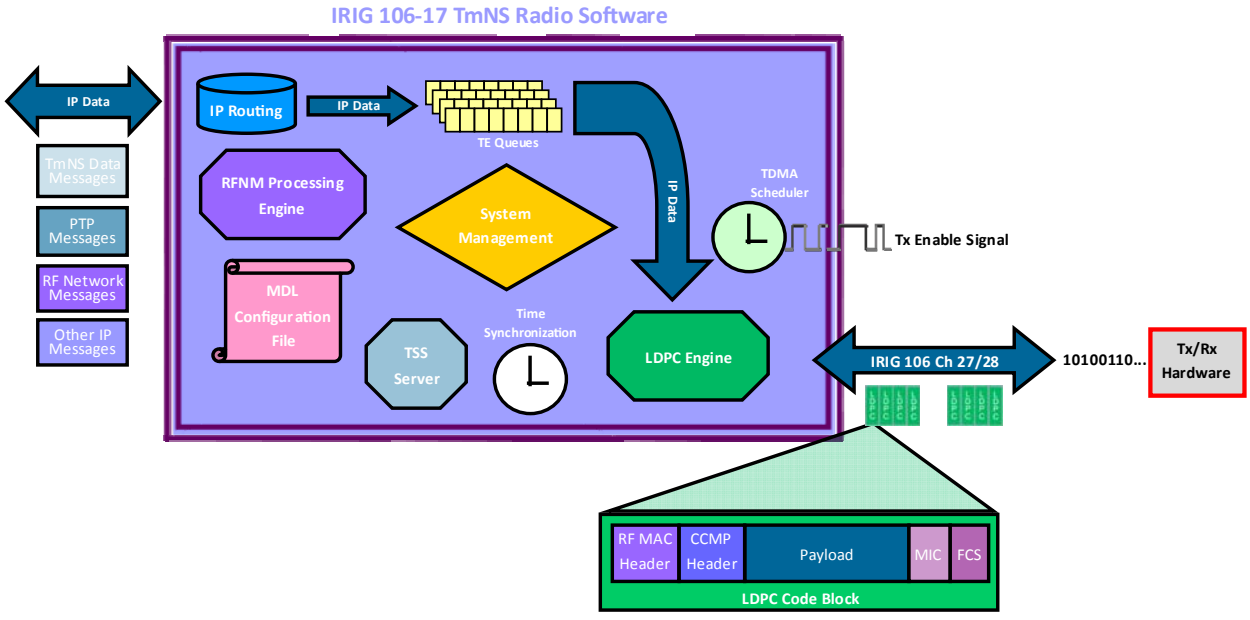

Figure 1. TmNS Radio Software

RFNM Processing Engine

The RFNM Processing Engine module manages a Link Agent connection by listening for incoming TCP connections over its Link Agent port. The module is configurable to be able to use TLS over its Link Agent connection; it can also be configured to only support TLS connections to its Link Agent. Once the connection is established, this module generates and receives RFNMs to and from a Link Manager. The module periodically generates RFNMs with TLVs that indicate its MAC queue status, TE queue status, link metrics for receiver statistics, link transmission statistics, and TxOp acknowledgements. Some of the TLVs are generated once per transmission opportunity; others are generated once per TDMA epoch; The TxOp acknowledgements are generated asynchronously in response to the receiving and applying of a new TxOp schedule that the radio receives. The module receives and parses TxOp assignment TLVs and Heartbeat TLVs that it receives from a Link Manager. These TLVs are used in the system to authorize and modify a radio's transmission schedule. This scheduling information is passed to the TDMA Scheduler module.

TDMA Scheduler

The TDMA Scheduler module maintains the radio's transmission schedule as provided either through MDL configuration or through RFNMs from a Link Manager. The module dictates when a radio is allowed to transmit data over its RF interfaces.

LDPC Engine

The Low-Density Parity Check (LDPC) Engine module performs the packing of IP data into LDPC code blocks in preparation for transmission over an RF interface. The output of this module is a serial stream of 1s and 0s that are sent to the transmitter hardware for modulation and transmission over the RF network. Time of transmission is based on the TDMA Scheduler module.

The module also performs the unpacking of LDPC code blocks received from the receiver hardware and reassembles their content back into IP packets that can then be routed as appropriate.

MDL Processing

Like all TmNS-compliant devices, a TmNS radio is configurable via a Metadata Description Language (MDL) configuration file. The MDL schema is defined in IRIG 106-17 Chapter 23. During the configuration process, the MDL Processing module parses an MDL configuration file, and from it the module applies the appropriate configuration, including its transmitting and receiving RF link addresses, queueing policies, RF frequency, RAN configuration parameters, and any initial transmission opportunities.

System Management

The System Management module provides an SNMP interface to the management resources required of TmNS radios. This includes TMNS-MIB as specified by IRIG 106-17 Chapter 25 as well as other publicly available RFC MIBs as required by IRIG 106-17 Chapter 22 for IP routing devices. The SNMP application utilizes version 3 of the protocol, adding authentication and encryption to the SNMP message exchange. This module provides the interface that the System Manager application utilizes for obtaining current device health and status as well as basic command and control of the radio.

Time Synchronization

Time synchronization is a key performance requirement within a TmNS-based system. Because the RF system is sharing the frequency amongst other radios, it is critical that all radios share a common time reference. The TmNS approach to time synchronization relies upon IEEE 1588-2008, the Precision Time Protocol (PTP) for synchronizing clocks over a network. The Time Synchronization module implements a PTP application that is capable of synchronizing its local clock to an IEEE 1588-2008 GrandMaster Clock in the network. If the host platform contains an external time source input, such as from GPS, the PTP application could operate as the GrandMaster Clock for other devices within its connected network.

IP Routing

At its core, a TmNS-compliant radio is an IP router. It maintains at least a single wired interface, and it maintains at least a single RF interface. If the host platform is Linux-based, this module is largely reduced to a platform configuration to utilize the IP routing functions of Linux. Though the MAC layers differ between the wired network interface, which is likely Ethernet, and the RF network interface, which is defined in IRIG 106-17 Chapter 27, traffic flows from these interfaces converge at the IP layer and can be routed out any available interface on the radio platform.

TE Queues

Traffic Engineering (TE) Queues are necessary in order to ensure proper levels of service are granted to the different data flows traversing through the TmNS radio. QoS policies on the radio service the TE Queues according to the queueing disciplines, structures, and rates as provided through MDL configuration of the radio.

TSS Server

The TSS Server module provides the capability of a TSS Server for a radio. Not all radio instances require the use of a TSS Server. It is currently only required when the radio is part of a group of radios that subscribe to the same RF link, such as in the case of multiple ground-based radios that are tracking the same Test Article (TA). The TSS Server configuration is provided through MDL configuration.

FROM EMULATION TO REALITY

With the end goal of seeing more TmNS-compliant radios enter the market, we see the TmNS radio emulator as a stepping stone on the accelerated implementation path. The radio emulator contains all the interfaces and network protocol support required by the standards. The only piece missing from the radio emulator is the actual RF hardware interface. Thus, by integrating the TmNS radio emulator software modules with a C-band transceiver, a TmNS-compliant radio implementation can be realized in a shorter amount of time.

The TmNS radio emulator's modular approach provides flexibility during integration of the software with transceiver hardware platform. The emulator provides a module for transforming IP packets into LDPC code blocks and can output a serial stream of bits to the transmitter hardware during transmission windows. However, if the target radio platform performs its own LDPC block generation and just needs IP packets delivered to it, the emulator software can easily be configured to deliver IP packets to the LDPC engine on the platform rather than the LDPC engine module provided by the emulator. Other emulator modules can be enabled or disabled as well.

The radio emulator framework is not dependent upon a particular RF waveform. Therefore, it is possible to add various TmNS interfaces to existing radio platforms today. Though only a single waveform, the SOQPSK-TG waveform, has been specified by the TmNS in IRIG 106-17, future revisions may incorporate other waveforms. The radio emulator framework would be unaffected by such changes. In fact, it may help pave the way for proving out and incorporating additional RF waveforms into future revisions of the standard.

CONCLUSION

The goal of the TmNS chapters in the IRIG 106-17 standard is to ensure interoperability amongst all TmNS components. Due to a lack of readily available TmNS-compliant radios or any alternative radio emulator solution, we developed our own radio emulator. This effort was initially done to support our development effort for the Link Manager application. The emulator implemented relevant TmNS interfaces in a modular fashion such that newer features and interfaces can be easily integrated into the emulated radio process, such as a System Management interface, Time Synchronization interface, TSS Server interface, etc. After adding support for a couple of TmNS interfaces, we realized that our radio emulator can be the foundation for building real TmNS-compliant radios. With the higher layer interfaces addressed by our software framework, the missing component to realizing a hardware TmNS radio is the output of the software that feeds into the transmitter/receiver system.

We believe that our TmNS radio emulator helps the IRIG 106-17 standards on telemetry networks by providing a reference implementation of the standards-defined interfaces for TmNS radios. Our TmNS radio emulator has been built according to IRIG 106-17 Chapters 21-28 and provides a path forward to accelerate future TmNS-compliant radio implementations. It has been tested and used as part of the iSIL for system level testing in support of both Link Manager and

System Manager application development. We believe that this software can be used as the de facto standard approach for new TmNS-compliant radio development efforts, ultimately reducing time-to-market for new radio implementations while providing a high level of confidence for interoperability of these new radios with other TmNS-compliant radio implementations.

REFERENCES

[1] *Telemetry Standards*, IRIG Standard 106-17, 2017.

[2] Faulstich, Raymond; Skelley, Daniel; Anderson, Brian, "iNET Deployment Process: A Case Study," in *International Telemetering Conference Proceedings*, 2009.

[3] " Raymond, Eric S., "The Art of Unix Programming," Addison-Wesley, 2003.