Data Flow and Remote Control in the Telemetry Network System

Daniel T. Laird, Air Force Flight Test Center
Edwards AFB CA

Jon Morgan, JT3, LLC
Edwards AFB CA

ABSTRACT

The Central Test and Evaluation Investment Program (CTEIP) Integrated Network Enhanced Telemetry (iNET) program is currently developing new standards for wired-wireless local area networking (LAN-WLAN) using the Internet Protocol (IP), for use in telemetry (TM) channels, under the umbrella of the Telemetry Network System (TmNS). Some advantages of TmNS are real-time command and control of instrumentation, quick-look acquisition, data retransmission and recovery ('gapless TM' or 'PCM backfill'), data segmentation, etc.. The iNET team is developing and evaluating prototypes, based on commercial 802.x and other technologies, in conjunction with Range Commander's Council (RCC) Inter-Range Instrumentation Group (IRIG) standards and standards developed under the iNET program.

KEY WORDS

Link, Frame, Transport, Serial Stream, Network, Data Flow, Control, Payload, SST, PDU, TmNS, UDP, TCP, IP, Unicast, Broadcast, Multicast, C^2, IRIG, iNET.

INTRODUCTION

To discuss data flow and control we designate our TmNS channel as both a serial streaming format, and 3rd layer and 4th layer internetwork protocols that carry data payload. The data channel is a unique abstraction, independent of the underlying *signal*

channel, which are usually referred to as 1st or physical (PHY) layer signal structure, and 2nd or data link (DL) layer framing and associated protocols[4][6].

To simulate data flow in the TmNS channel require serial streaming telemetry (SST) frame and network protocol data unit (PDU) payload models for the TmNS (Figure 1.). The SST frame and PDU payloads are expected to be similar, but not identical. In conjunction with the data flow, a command and control (C^2) completes a ground station (GS) to test article (TA) communication link (CL). The SST, PDU downlinks and C^2 uplink form a prototypical model of an TmNS channel.

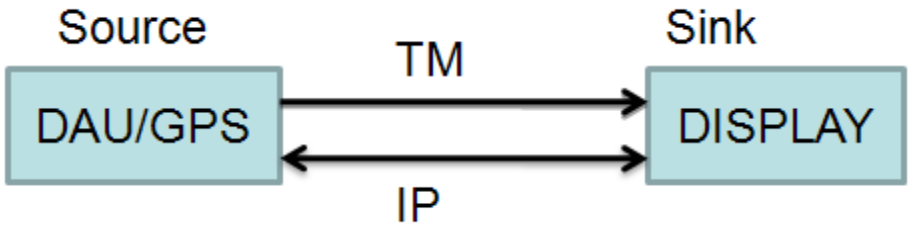

Figure 1
TmNS Channel

This paper (the fifth in a series of papers on simulating the TmNS channel[1][2][3][4]) will discuss prototype development and preliminary findings of full TmNS channel prototype testing. Our modeling approach employs specific aspects of IRIG-106[7] telemetry (TM) standards, the Internet Protocol (IP) with User Datagram Protocol (UDP) [5] transport. Our approach is also consistent with the Open Systems Interconnection (OSI)[6] model. All iNET standards developed by the standards working groups (SWG) imply commercial IP networking and IRIG TM standards.

DATA FLOW

A *downlink* refers to a communication link in the TmNS channel, i.e., from a TA to a GS. The downlink in our model is a composite of one TM and two IP channels with source and sink operating as a client-server link (Figure 2). Although independent of serial TM and network physical (PHY) and data link (DL) layers, the simulations imply legacy pulse-code modulation[1] (PCM) formats, and both wired and wireless media access control (MAC) and (LLC) protocols[5].

[1] PCM is a comprehensive legacy term for data formatting, bit coding, serial streaming and signal modulation. Data structure is independent of any imposed pulse stream modulation technique.

Source Model

The source model (Figure 2) employs for both IRIG standard (legacy PCM) *formats*, for the TM, and the UDP *transport* protocol operating over an IP *network* protocol.

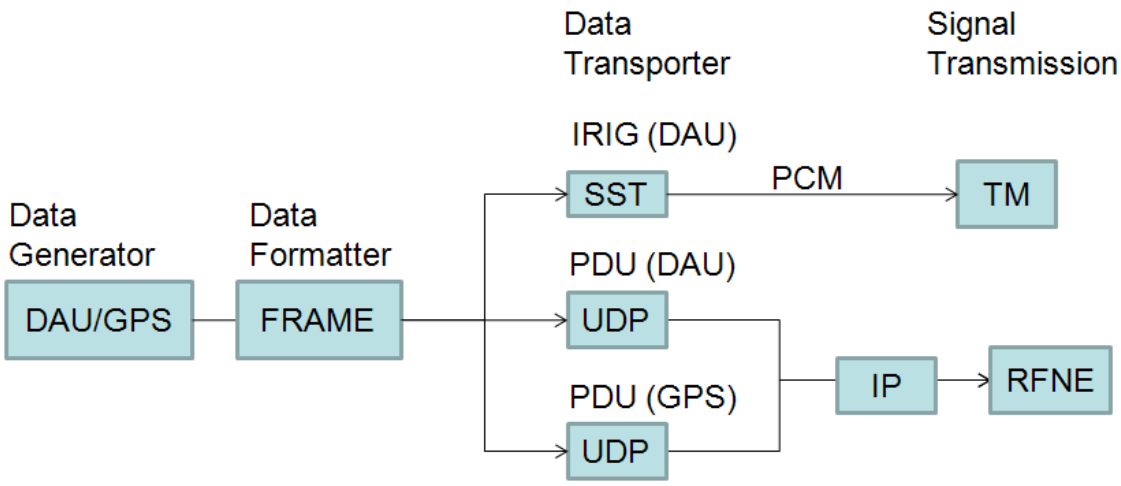

Figure 2
TmNS Channel Source Model

The PDU payload are 32-bit signed integer and 64-bit floating point *words*. Frame size is selectable. Each frame contains a SST and PDU headers. The PDU header is an additional six words, plus a status word, which are removed prior to SST subsystem transmission to comply with IRIG Chapter 4 SST format. The RFNE designates the RF network element: the wired and wireless network interface. RFNE standards are now being developed under iNET by the network element standards working group (NESWG).

The LabVIEW™ (LV) graphical user interface (GUI) used to generate the source data, frame it and transport it to the transmitters is shown in Figure 3. The controls for assigning IP addresses and for configuring the IRIG frame are on the left. Controls for four types of data sources are on the bottom right. The sources are a sine wave, a triangle wave, a gaussian noise and a random multiplier that scales random numbers generated over the real number interval [0,1]. The top right is a display of the GPS data prior to packaging into UDP/IP for multicast. The other controls are for frame timing, turning channels 'ON' or 'OFF' and saving source data. The yellow 'SyncClock' and white background 'ATMSocket' are Active-X indicators, of the controls that bind the LV code to the Monarch-E TM system and a Brandywine™ PCI-SYNCCLOCK32 GPS system.

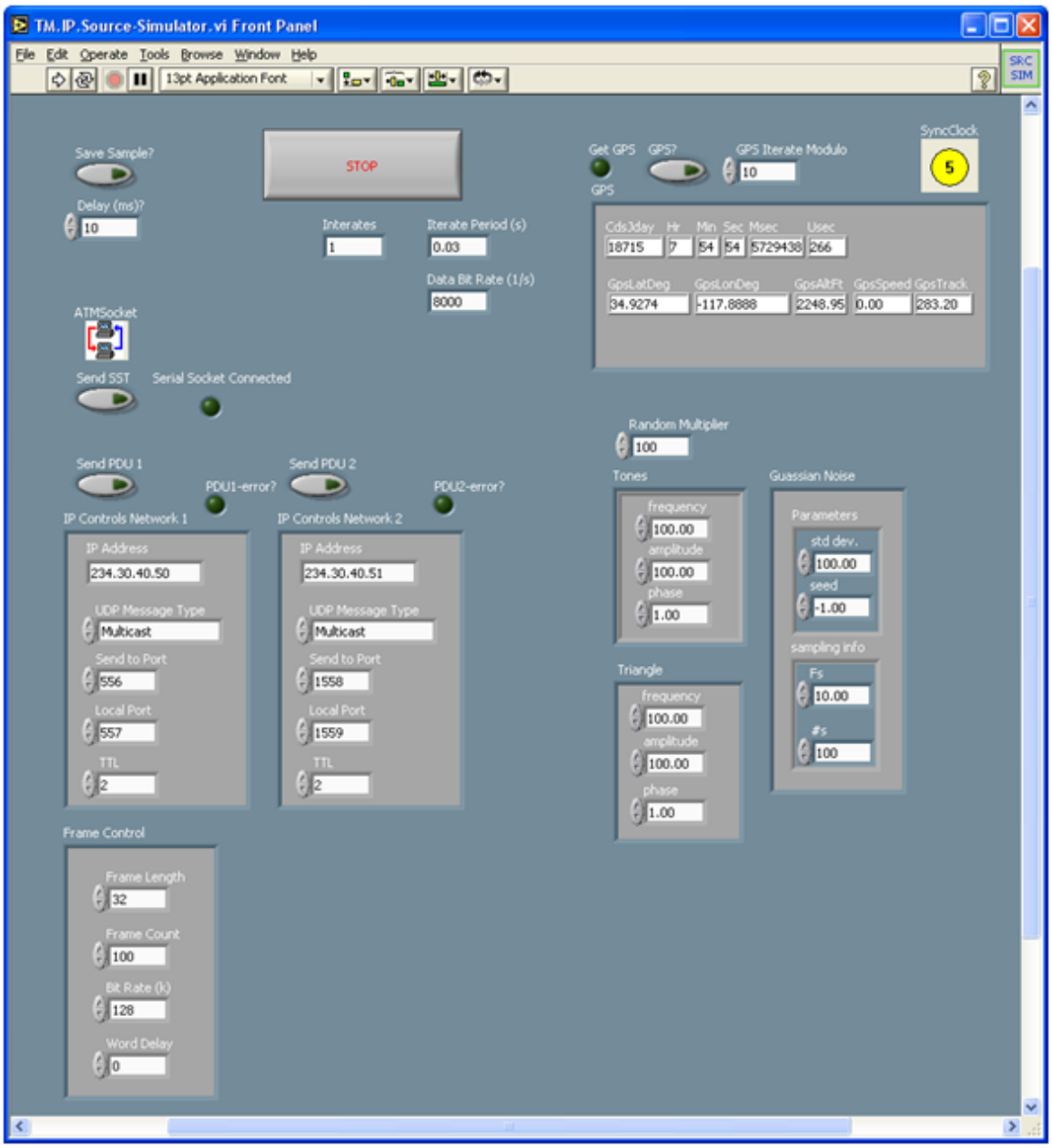

Figure 3
Data Generator GUI

The GUI controls (on the left hand side) shows three selectable transport layer *addressing schemes*[5]:

- **Unicast**: addressed to a *specific* host on a network.
- **Broadcast**: addressed to *all* hosts on an subnet.
- **Multicast**: addressed to a *some* hosts on a subnet.

To emulate TmNS data flows we use multicast addressing. In the LV application, SST frames are converted to serial streams via an ATM Socket interface to a Monarch-E TM system, which sources data and clock at a fixed rate to drive an IRIG-106 transmitter (Tx). The PDU payload is carried in a UDP transport over IP. All 'host addressing' is at the 3rd (IP) layer.

The source model generates a data frame of selectable size and transports this over a TM and an IP channel. A second IP channel carries GPS data acquired from the PCI-SYNCCLOCK32 system for delivery over another Ethernet link. Each IP source connects to GS sink via the transport and assigned ports. The interfaces that bind the GPS and TM subsystems to the data source are via Active-X controls. The serial stream module of the Monarch & LV data source simulator run independently, synchronized by local clocks jammed to GPS time.

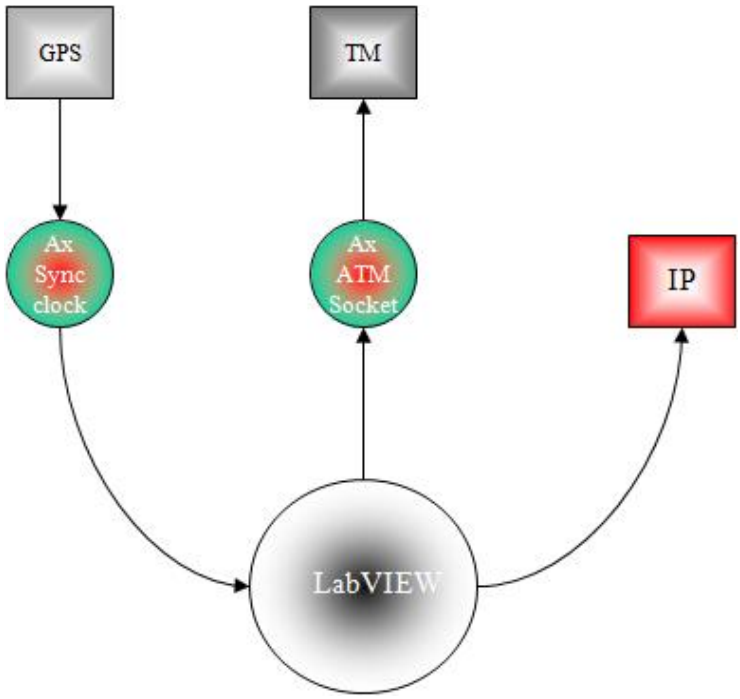

Figure 4
Active-X Application Connections

The source model is built as a LV application that binds to other applications via Active-X controls (Figure 4.). This model can run in either the Windows™ XP operating system (XPOS) or Linux operating system (LXOS).

Sink Model

The sink models are based on prototypes of data streaming processes presently under investigation and development by GSSWG for IP data routing, TM decommutation, stream segmentation and word presentation. A diagram is shown in Figure 5.

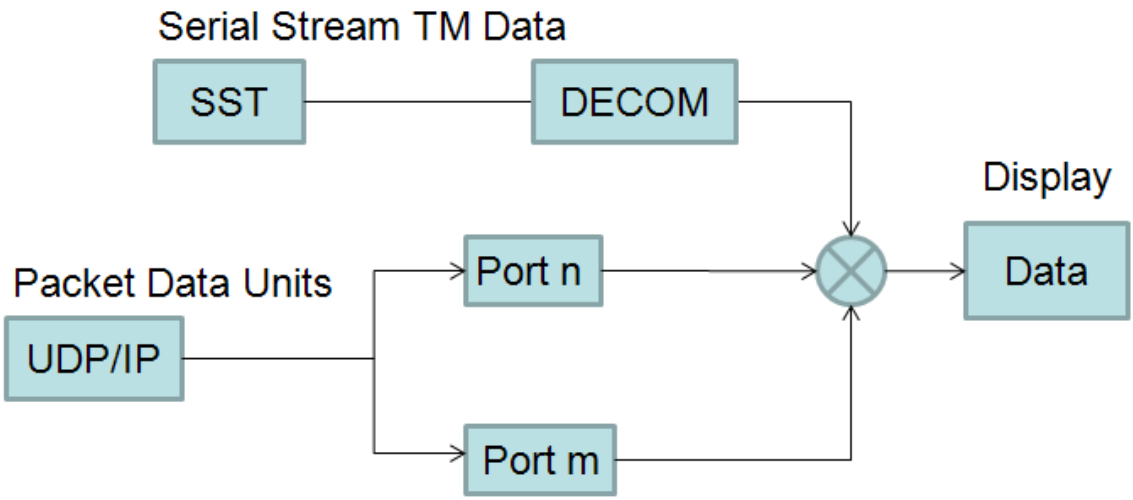

Figure 5
TmNS Channel Sink Model

A prototype GS application collects both the SST and PDU payloads and segments the data for display and side-by-side comparison of both the TM and IP segments. A screen shot of the Interactive Analysis and Display System (IADS) is shown in Figure 6.

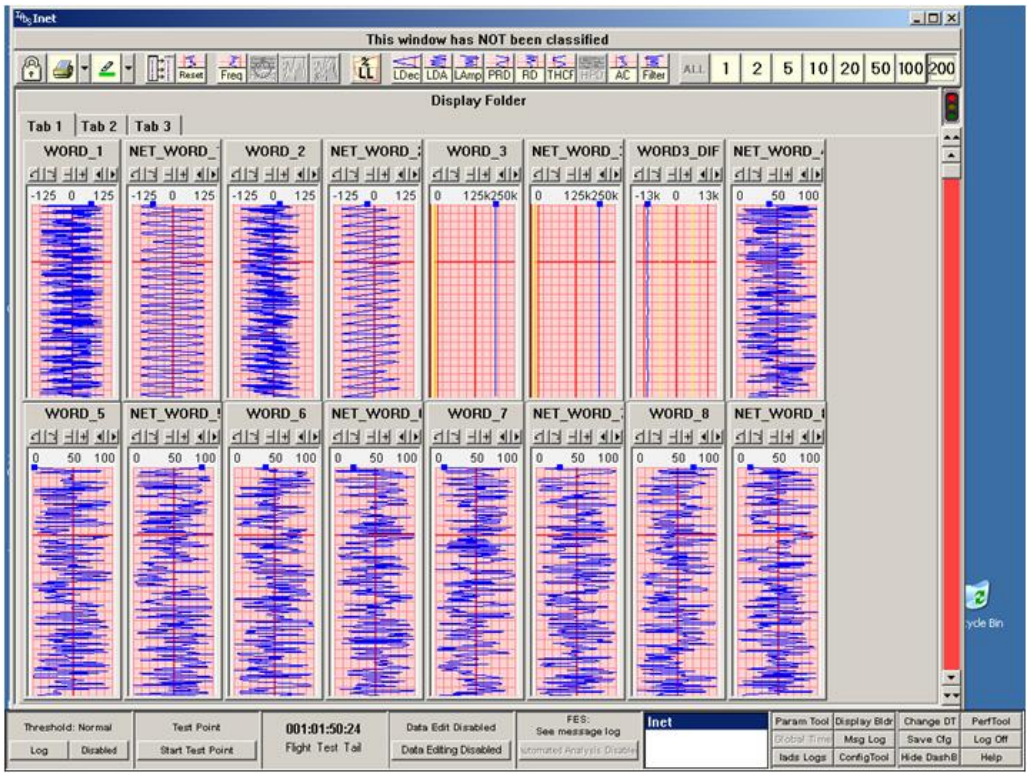

Figure 6
GSS IADS Application GUI

This shows TM and IP data segments side-by-side. You can see a sine wave and triangular wave in the second and third 'strip-charts'. The first window is the same data as the seconds window; the third is the same as the fourth. The difference in the views is due to time scaling.

Initially, to test the source delivery over the IP network, a rather simple sink model, built in LV/XPOS, linked with the source via UDP/IP sockets on local host. Tests were run using all addressing schemes to verify source structure and data delivery. We verified the data delivery via a display of real-time and stored payloads. A Monarch module was used to capture and store the SST payload for verification of the TM delivery via a third party data display application. We then established connections to remote workstations running developmental GS applications in LXOS. To segment payload and display data we employed existing decommutation and network hardware, and GS applications now under development by the GSSWG. The GS segments conform to payload standards developed by the TASWG. The GS applications used are built primarily in the C programming language, and run on a LXOS Fedora Core (FC9).

Transport Data Packets

Simulated TmNS data is transported via UDP in the TASWG payload format[9]. A message is show in Figure 7. The SST data payload contains 1 status word followed by 1 minor frame packaged as data words.

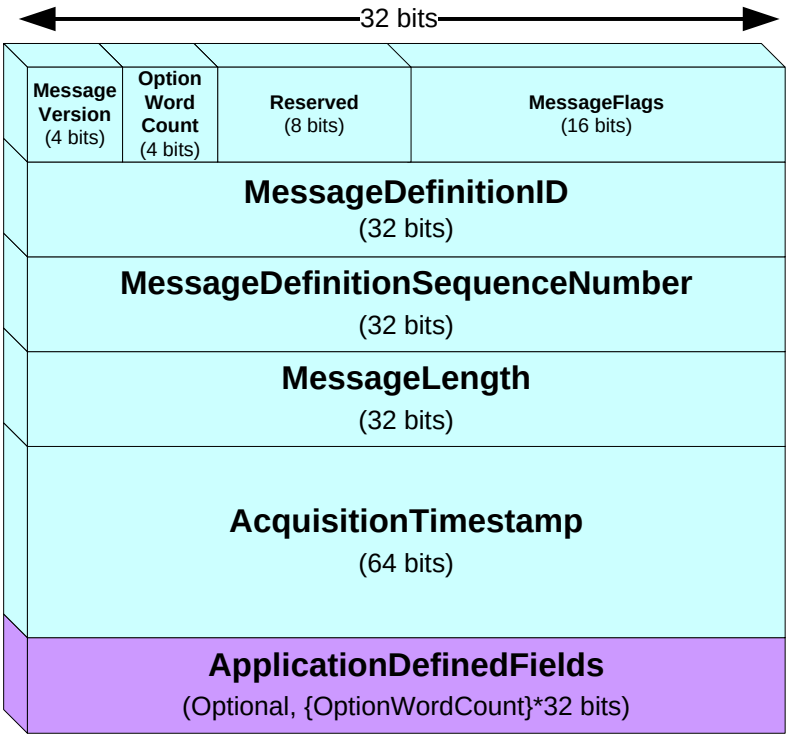

Figure 7
TmNS Message Header Format

To keep processing simple, only TmNS Data Message Headers append data i.e., no payload package headers were used. The format of a TmNS Data Message includes a 6 word header followed by payload. The header contains 8 fields. We foresee a similar structure for 'TmNS Control' and 'TmNS Status' Messages.

PCM Backfill

An advertised capability of iNET is the ability to provide what is known as 'gapless TM'. One approach we're experimenting with provides this capability by seamlessly replacing corrupted SST (PCM) frames with recorded frames retrieved from the on-board recorder or data acquisition unit (DAU). The GSSWG call this capability 'PCM backfill' (Figures 8. And 9.).

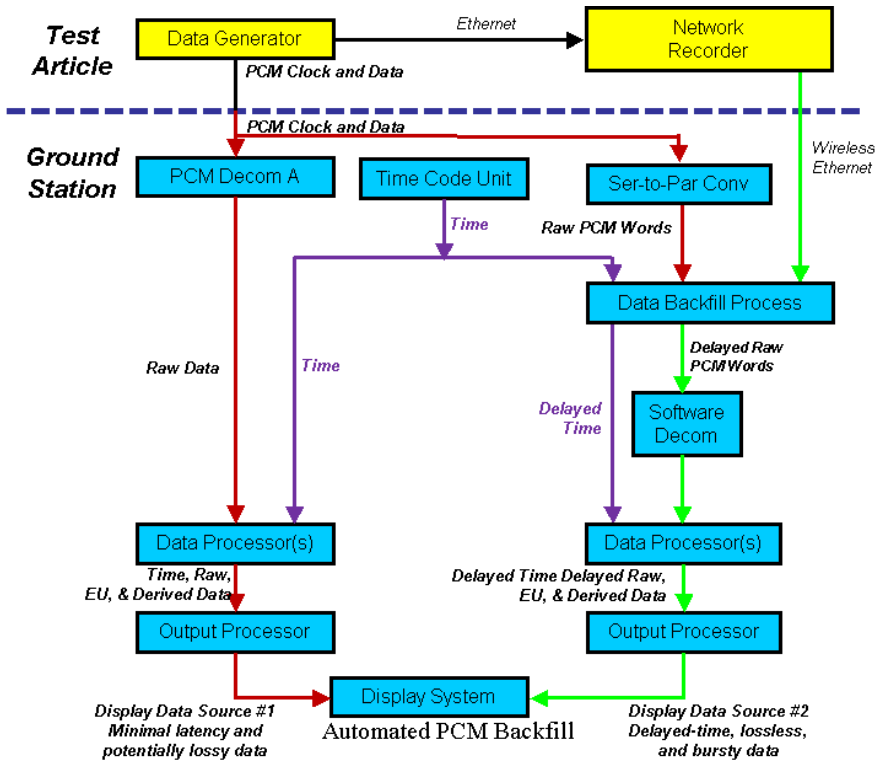

Figure 8
TmNS Message Flow

Backfill Data Flow

In our implementation, we will not replace corrupt data after it's been processed, but rather, we process backfill data when corrupt PCM data is detected. We queue PCM data while fetching replacement frames from the recorder or DAU. The replacement frames

are then inserted into the PCM stream. This non-corrupt data will be treated at the GS as a completely independent data source. Data transport for backfill is via TCP.

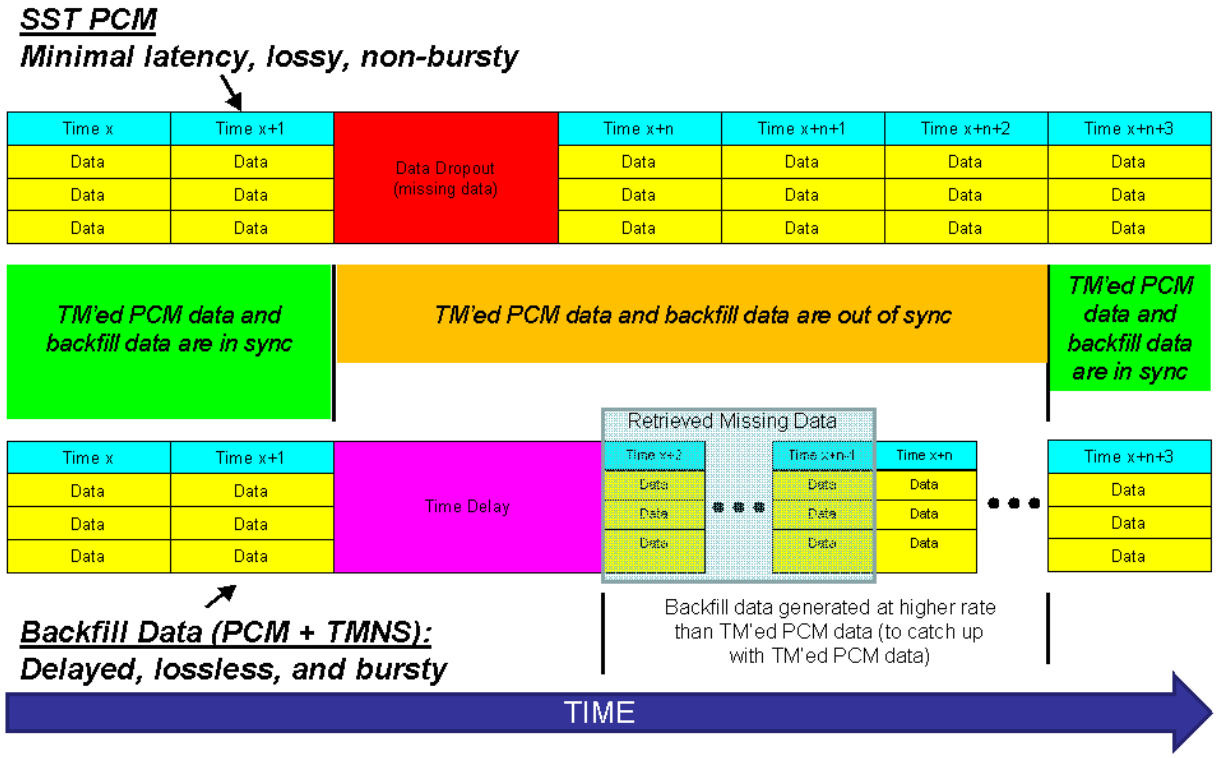

Figure 9
TmNS Message Backfill

Other Data Flows

Other data flows besides back fill of PCM will also be tested as we progress with our experiments and development. Several are presented in a companion paper: '*Augmenting Serial Telemetry with TmNS Data Delivery*'[8]. We will test a TCP connection to the recorder to get, in near real-time, a small data set in the TmNS data message format. The data will be compared to the real-time transmission of the UDP multicast data with the same content.

iNET Recorder

We've also built a prototype iNET Recorder prototype as an application over LXOS on a PC104 computer. The iNET Recorder employs COTS Dell servers and the PC104 computer. The recorder is configured via custom eXtensible Markup Language (XML) files. The recorder 'listens' for IP multicast packets that conform to the TASWG TmNS data message format, and records the specific message definition identifiers (MDID) listed in the configuration file.

Using the Dell server and an external USB hard drive, a recording rate of 17.1 MB/second (MB/s) was achieved. Using the same USB hard drive on the PC104 computer, a recording rate of 3.96 MB/s was achieved. This lower record rate is a result of the slower 10MB/s Ethernet link on the PC104 Computer. When running performance tests on the PC104 computer, via local-host, results comparable to the Dell server were achieved.

Metadata

To describe the individual measurements for processing in the GS, TmNS *metadata* was not used. Instead, the follow on to the Joint Test Data Management System (JTDMS)[9], called Telemetry Attributes Manager (already incorporated into the AFFTC Mission Control System (MCS) telemetry processor) configures all measurement parameters for decommutation, segmentation and data display.

Results

The actual payloads delivered over TM and IP channels were verified via a full TmNS channel simulation. We directly plumbed both *data* and *clock* from the Monarch-E TM system to a PCM decommutator; or *data* only to a bit-sync, via RF plumbing (simulated RF link). Although we could not get a deterministic synchronization between the LV/XPOS and Monarch-E applications via the UDP sockets, both the decommutator and bit-sync locked on the serial TM data streams. The IP data packets were delivered via a direct IP routing, and viewed with 'Wire Shark' (a.k.a. 'Ethereal' – a network standard third-party packet viewing application), as it was traversing the network. Wire Shark permits displays of the data as packaged at each layer of the IP stack, i.e., from link to transport layers.

Other custom LV tools and the IADS run in both XPOS and LXOS, and allow for direct access to the data flows for comparison between TM and IP source data, i.e., we can view SST and PDU payload parameters. There were notable delays between the TM and IP deliveries, about ½ second delay (see Figure 6) was encountered on the SST data due to slack synchronization of the LV/XPOS AX Monarch-E interface. Data loss using the TM interface was also encountered, due to the inability to synchronize the timing of the LV calls with the Monarch PCM transmitter clocking edges.

Future Development

In flight, the TM data will be transmitted to the GS using standard legacy means. The TmNS data stored onboard an 'iNET aware' recorder will allow queries to command backfill for TM signal dropouts. The reconstructed 'pristine' data will be displayed side by side with the legacy SST data in the IADS. A 'TCP like' protocol will provide transport between the iNET recorder and GS applications. The next phase of source data will employ an FPGA data frame generator, built on NI PCI eXtended for Instrumentation (PXI), to directly drive an IRIG-106 compliant TM Tx and fill PDU for transport, thus eliminating the slack synchronization of the TM and IP channels and the delay between the running data of the IADS displays. The PXI will house both digital and analog acquisition units, a GPS receiver, an ARINC 429 bus interface unit, as well as standard peripheral interfaces, e.g., Ethernet (10/100Mb and Gig-Ethernet), EIA-422,

USB, 1395, etc.. As with prior work on communication links, we will also simulate the signal channel using the channel emulator in the Telemetry Lab at Edwards AFB CA.

COMMAND and CONTROL

An *uplink* refers to a communication link from a GS station or site to a TA asset and/or network. As with the downlink, an IP network provides the GS to TA link. A C^2-interface connects a GS work station to a TA asset, in this case an IRIG-106 compliant TM Tx as a client-server implementation using UDP/IP, port mapping and an IP-to-HDLC converter. Figure 10 shows the command flow.

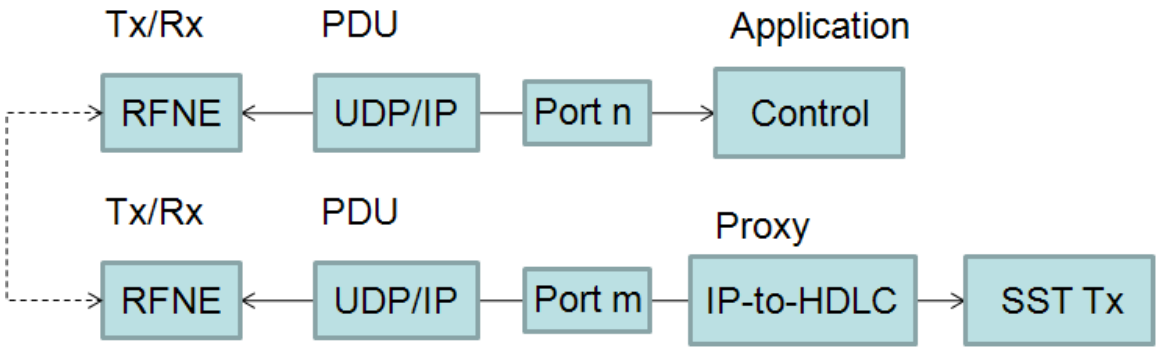

Figure 10
Command and Control Flow

C^2 Uplink

To uplink we use a C^2-interface built in LabVIEW™ and run on the XPOS. The GUI for a client-server proxy is shown in Figure 11. Intially[4], this interface was built on LV/XPOS on a PC104 computer, using two methods of transport over an 802.x[5][11] link: the first employed TCP/IP and the Hypertext Transfer Protocol (HTTP), so the operator uses either a client-server or a web-browser to proxy an interface to the IRIG-106 Tx; the other proxy mapped serial ports from a remote OS to a local OS using virtual instrument standard architecture (VISA) serial port mapping protocol.

In the latest incarnation we interface via an IP-to-HDLC converter. This converter allows for transport of serial command in TCP or UDP PDU transport payload over an IP network, without need for an application (e.g., an agent) interface proxy. The IP-HDLC converter bridges a serial device and network device (in our case, the network device is either a router or Ethernet switch). The converter unpacks the IP and TCP payloads and converts the payload (in our case, ASCII coded commands) into serial HDLC for message exchange with the TA serial asset (in our case, an IRIG-106 Tx). Conversely, asset responses are translated from HDLC as IP PDUs (TCP or UDP transport) payload,

preserving the ASCII data structure. The IP is then framed in the Ethernet DL layer for delivery to the remote workstation via our network downlink.

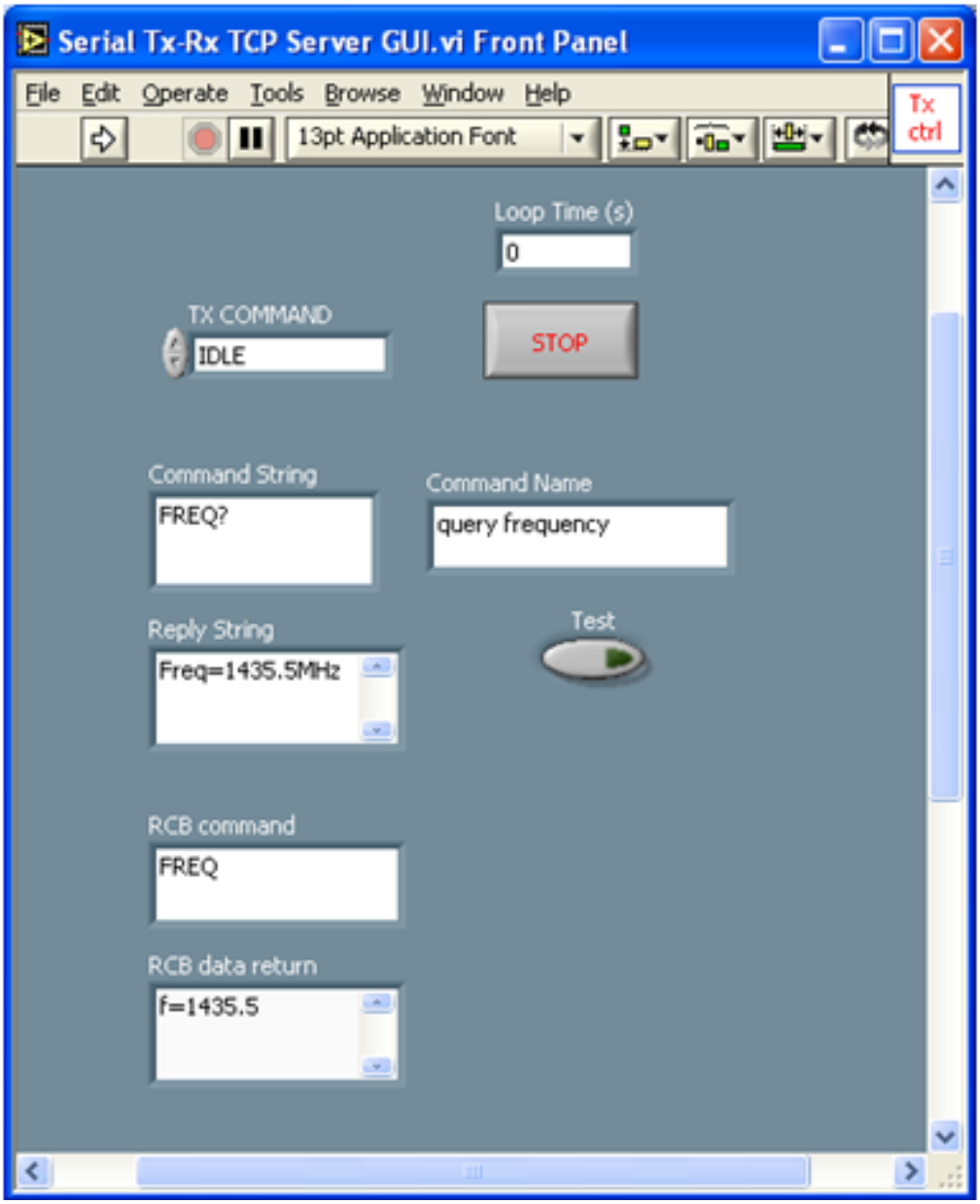

Figure 11
Local server Proxy for SST Remote Control

Results

As there are no rigid timing requirements for the delivery of commands from a local GS workstation to a remote TA asset, we can employ acknowledgements or a connection-oriented transport protocol for the uplink C^2 interface. Using TCP/IP, the C^2 uplink behaved as expected. We were able to reconfigure retrieve status messages from the TA ARTM Tx via the link. We previously interfaced with a remote video camera to retrieve streaming video over our standard GPS/TSPI link, both in the AFFTC Telemetry Lab and flight tests.[4]

SUMMARY and CONCLUSION

To simulate an TmNS channel, we have developed prototypes over the past three years that implement various subsystems of the TmNS channel. This is the fifth in a series[1][2][3][4] of papers that presents the channel development via prototype testing.

The initial prototypes were built around 802.x[5][11] standards, using commercial network, instrumentation, XPOS and LXOS applications and hardware products. Initially we tested the RF link, using a Harris Corporation SecNet11+® 802.x transceiver operating in a LXOS environment. We then incorporated C^2 interfaces using various proxies and 802.x Mikrotik routers to interface GS workstations with TA assets, e.g., data recorders and an ARTM Tx using LV/XPOS development application to create custom software application proxies. Now we've incorporated the 'final link' in the chain with a simulation of a full GS-to-TA communication using both IRIG TM and commercial IP standards. The next step is to implement a full wide area network (WAN) with a LAN-WLAN interface to existing range infrastructure to test network performance and limitations.

The test results show that telemetry and networks can operate in parallel as one TmNS channel to extend the operator interfacing with TA assets during flight with an TmNS channel. With both a data downlink and C^2 uplinks, the potential use of the channel to extend TM and instrumentation is only limited by developer and user imagination.

ACKNOWLEDGEMENTS

The authors would like to thank the OSD/CTEIP program office for the support and opportunity to develop these prototypes and evaluate performance under the iNET Program. Special thanks to Mark Ross and Glen Wolf for their technical assistance. As always, thanks to CLSWG lead, Kip Temple and GSSWG lead, Bruce Lipe for guidance and confidence in our work.

REFERENCES

1. 'An Airborne Telemetry Link for the iNET Technical Demonstration System', Kip Temple and Daniel T. Laird, ITC San Diego 2008.
2. 'An Airborne Network Telemetry Link', Kip Temple and Daniel T. Laird, ITC San Diego 2006.
3. 'Instrumenting an Airborne Network Telemetry Link', Daniel T. Laird and Kip Temple, ITC San Diego 2006.
4. 'Operator Interfaces for Controlling Serial Streaming Telemetry via a Network Command and Control Link', Daniel T. Laird, ITC Las Vegas, 2007.
5. 'TCP/IP Architecture, Protocols and Implementation', Sidnie Feit, McGraw-Hill, New York New York, 1993.
6. 'OSI Upper Layer Standards and Practices', Baha Hebrawi, McGraw-Hill, New York New York, 1992.
7. 'IRIG standard 106', Comprehensive Telemetry Standard, parts 1and 2, Range Commander's Council, 2007.
8. 'Augmenting Serial Streaming Telemetry with iNET Data Delivery', Carl Reinwald, ITC Las Vegas 2009.
9. 'Test Article Standard, version 0.6.1' (pending), TASWG 2009.
10. 'JTDMS: an Object Oriented Approach to Telemetry Attribute Management', Jon Morgan, Computer Sciences Corp., ITC San Diego, 2000.
11. '802.11 Wireless Networks, The Definitive Guide', Matthew S. Gast, O'Reilly, Paris, 2002.