INET STANDARDS VALIDATION:
END-TO-END PERFORMANCE ASSESSMENT

Myron L. Moodie¹, Maria S. Araujo¹, Thomas B. Grace²,
William A. Malatesta², Ben A. Abbott¹
¹Southwest Research Institute®  ²Naval Air Systems Command (NAVAIR)
San Antonio, Texas  Patuxent River, Maryland
myron.moodie@swri.org, maria.araujo@swri.org, thomas.grace@navy.mil
william.malatesta@navy.mil, ben.abbott@swri.org

ABSTRACT

The integrated Network-Enhanced Telemetry (iNET) project has developed standards for network-based telemetry systems. While these standards are based largely on the existing body of commercial networking protocols, the Telemetry Network System (TmNS) has more stringent performance requirements in the areas of latency, throughput, operation over constrained links, and quality of service (QoS) than typical networked applications. A variety of initial evaluations were undertaken to exercise the interfaces of the current standards and determine real-world performance.

The core end-to-end performance initial evaluations focus collectively on the movement of telemetry data through the TmNS. These initial evaluations addressed two areas: end-to-end data delivery and parametric data extraction. This paper presents the approach taken by these ongoing efforts and provides initial results. The latest results will be presented at ITC 2010.

KEYWORDS

IP networking, real-time, QoS, iNET

INTRODUCTION

The integrated Network-Enhanced Telemetry (iNET) project has developed standards for network-based telemetry systems. While these standards are based largely on the existing body of commercial networking protocols, the Telemetry Network System (TmNS) has more stringent performance requirements in the areas of latency, throughput, operation over constrained links, and quality of service (QoS) than typical networked applications.
The networks used in telemetry applications may use some of the same building blocks that are used in typical enterprise computer networking, but the performance requirements for telemetric networks require purposeful planning of architectures and interfaces to achieve success. In enterprise networks, most applications can live with best effort service delivery because the applications rarely have tight latency requirements. If it takes a few seconds or minutes longer to get your email or file transfer through when the network is heavily utilized, then usually that is deemed acceptable in enterprise networks as long as the data gets through in a “reasonable” amount of time. Even for more “time-critical” applications like voice or video, the network latency variations can be absorbed by sufficiently large application-level buffers. Enterprise networks also have the “luxury” of relatively low average network utilization and a small percentage of “high-priority” traffic in the network. Since most enterprise network users’ traffic patterns are highly bursty (i.e. low duty cycle of network utilization from a single node), the aggregation of the network can leverage these statistics to provide sufficient capacity to all users without planning capacity equal to the sum of the peak loads from each user.

Telemetric networks have a much more difficult problem than enterprise networks. Since a majority of the traffic in these networks is movement of acquired test data, a high percentage of the traffic has tight latency requirements (100s of milliseconds) due to safety of flight concerns. This same characteristic leads to a high percentage of the traffic in telemetric networks also being considered “high priority”. Since much of the acquired test data is either periodic sampled data or bus event driven by periodic message update rates in avionic systems, the data throughput from both individual data acquisition units (DAUs) and the aggregate of all DAUs is a fairly constant rate when observed in a one second window. Add in that some portion of the telemetric network will be on a test article with size, weight, power, and environmental constraints and the network will likely have some portion that has high constant network utilization with high priority and low latency requirements.

One of the key benefits of the new iNET program is the development of a two-way network telemetry link connecting the test article network to the ground network. Within iNET’s Telemetry Network System (TmNS) architecture, the Test Article Segment (TAS) connects to the Telemetry Ground Segment (TGS) by way of the Radio Access Network Segment (RANS) and Range Operations Segment (ROS). Like most wireless networks, practical constraints on available spectrum mandate that the available network throughput in the RANS is significantly smaller (in some cases two to three orders of magnitude smaller) than in the test article and ground networks it connects. Managing this constrained RANS link efficiently is critical to ensuring that as much high-priority traffic is able to pass with as little latency as possible, often with additional reliability constraints.

These constraints require acknowledging that more network throughput will be routinely requested by end applications than is available in the network. In typical enterprise networks with constrained links like digital subscriber line (DSL) or cable modem, all data fight equally for available throughput and thus all applications slow to the capability of the link. The use cases for TmNS require that the access to available throughput be favored and possibly guaranteed to the applications with the highest priority. Implementing approaches to both
quantify and mark the relative priority of data and then enforce these priorities in the network are generally categorized as Quality of Service (QoS) mechanisms.

**INET STANDARDS FOR END-TO-END PERFORMANCE**

Since achieving multiple levels of QoS within a network-based system requires an end-to-end system-wide approach, there are QoS-related elements in each of the iNET standards. The test article, RANS, radio, and ground standards focus on the data transport portions of QoS. The system management standard provides mechanisms for determining status of network utilization and application throughput. The metadata standard provides the ability to describe test setup that is used by the system management interfaces to configure the QoS-affecting settings within the system. All of these pieces must work together to provide end-to-end QoS.

A major function of a TmNS is to move acquired test data from test articles to ground processing with different needs for latency, throughput, and reliability that vary depending on the particular measurements and on the test operating conditions. For instance, during one phase of a particular test, the test operators may need samples of a particular set of measurements with as little latency as possible due to safety of flight issues even if it means losing some samples during telemetry dropouts. In another phase of the same test, the test operators may need reliable transport of the same measurements for analysis even if it raises latency due to resending data lost during telemetry dropouts. The Test Article Standard provides two protocols to accommodate these varying needs. The Latency/Throughput Critical (LTC) Delivery Protocol specifies how to deliver TmNSDataMessages when latency or throughput constraints are more important than reliability constraints. The Reliability Critical (RC) Delivery Protocol specifies how to deliver TmNSDataMessages when reliability constraints are more important than latency or throughput constraints.

LTC uses the User Datagram Protocol over the Internet Protocol (UDP/IP) to deliver sequences of TmNSDataMessages to multicast addresses. Delivery to unicast and broadcast addresses is also allowed. Since the data delivered by LTC is by its very nature latency critical, this protocol is focused on controlling latency first and data reliability second. Consequently, no guaranteed transport mechanism is used that would require acknowledgements and resends which would delay data outside of the latency performance bound. The MessageDefinitionSequenceNumber field in each received TmNSDataMessage can be used by the DataSink to determine what data (if any) was lost due to a telemetry dropout or other system problem. When used in the typical multicast variant, LTC also provides an efficient means of sending the same data to multiple DataSinks. Consequently, LTC is very well suited to sending live test data from DAUs to recorders, computational units, displays, etc. within the test article network where network reliability is virtually guaranteed due to the fully wired media.

The second protocol for the delivery of TmNSDataMessages is the RC delivery protocol. The RC delivery protocol includes detailed mechanisms (i.e. transport and application protocols) for implementing reliable data transfer between DataSources and DataSinks. Since additional session control is necessary to ensure highly reliable transfer, this type of delivery occurs between only two devices (unicast) with no guarantees for latency and throughput metrics. RC
Data delivery uses the Transmission Control Protocol over the Internet Protocol (TCP/IP), which provides error-free delivery of a data stream at the transport layer using a system of acknowledgements, timeouts, and retries. Although solid reliability is maintained for RC delivery, minimal latency is not guaranteed and competing flows reduce throughput when the network is congested.

The Test Article Standard specifies the widely used Differentiated Services (DiffServ) QoS protocol as an additional tool to meet QoS requirements when traffic engineering measures may not be sufficient to address momentary congestion within the TmNS. The QoS protocols implement a set of Per Hop Behaviors (PHBs) at the outputs of NetworkNode interfaces. The PHBs define a set of policies enforced at input and output queues of EndNodes and when forwarding (routing) data through the TAS network as a means of prioritizing data aggregates. The PHBs further define policies regarding traffic shaping, re-marking, packet discard, and latency. Unified implementation of QoS protocols across the TmNS can provide for fair delivery of multiple data aggregates having disparate QoS requirements over various parts of the TmNS (TAS, RANS, ROS, TGS). DiffServ Code Point (DSCP) values specify PHBs and are marked on each packet at the source based on priority information contained in the metadata instance document for the particular test.

Since the RANS portion of the TmNS will represent the most throughput constrained link in the system, achieving end-to-end QoS requires mechanisms to prioritize queuing of the packets passing through RANS. Fortunately, the choice of DiffServ as a QoS mechanism in the Test Article Standard makes implementing this priority queuing straightforward. DiffServ is commonly available in most modern routing hardware and software implementations. Consequently, the RANS (and the connected ground network infrastructure) can easily leverage this DiffServ capability to provide the needed priority queuing.

**AN ASSESSMENT FRAMEWORK FOR EVALUATING THE INET STANDARDS**

Now that the initial iNET standards are documented, the next step to maturing the standards is to demonstrate the functionality in as realistic a network as possible. To achieve this, a series of standards assessments are in progress that exercise and evaluate the completeness of the interfaces specified in the standards. These standards assessments use a common network test bed that was developed to emulate a real TmNS using currently available hardware and newly developed simulator software. The results from these standards assessments will determine which portions of the standards are complete, identify any needed modifications, and provide confidence to equipment vendors to begin investments to implement the standards in their hardware.

The validation test bed is comprised of several interconnected networks that represent test articles, the RANS, the TGS, and the Mission Control Center. The networks consist of DAUs, network recorders, network switches, system managers, a Serial Streaming Telemetry (SST) transmitter and receiver, data processing units, and routers to connect to other subnets. Ideally, all equipment would be actual flight test hardware. However, this is not a practical approach due to the immediate need for proving iNET technologies and the lack of availability of actual flight
test hardware that complies with the iNET standards. To cope with this, device emulators have been deployed in the networks, some as end devices and others as TmNS proxies. They may be used in the place of the actual flight test hardware until the real hardware becomes available. For the flight test hardware that is currently in the test bed but not fully iNET-compliant, the device emulator will serve as a TmNS proxy device.

From an end-to-end QoS perspective, the key capability needed of the network test bed is an emulation of the network capacity and latency of the RANS link. A combination of commercial off-the-shelf (COTS) hardware and COTS and custom software provide the emulation of the RANS link since no real RANS equipment that implements the iNET standards currently exists. Open-source routing software has been combined with custom software to both constrain the link capacity to expected RANS throughput capabilities and also delay packets to match the propagation delay expected over maximum distance RANS links. These throughput limits and delays are configurable to allow testing over a mix of operating conditions. More details on the configuration of the test bed can be found in a separate ITC 2009 paper [1].

The core end-to-end performance assessments focus collectively on the movement of telemetry data through the TmNS. These are the end-to-end data delivery, selective data retrieval, and TAS/RANS interface throughput assessments. The end-to-end data delivery assessment investigates the practical performance in a TmNS network of the QoS technologies and approaches. This assessment utilizes software EndNode emulators that generate and receive LTC and RC delivery protocol traffic. The DiffServ priority markings are configurable for each source of measurement data generated by these DataSources. A variety of test cases have been developed to test the performance of these data delivery and QoS mechanisms for a mix of small and large network setups. Initial testing focused on sending individual flows of LTC and RC data from the test article network to the ground network with best-effort DiffServ priority markings. Once these tests verified the basic connectivity of the network and proper functioning of the EndNode emulators, testing progressed to increased numbers of traffic flows and mixes of DiffServ markings. Additional test cases involve adding system management traffic and varying the constrained RANS link during tests. At each of these test cases, the throughput, latency, and data loss statistics are collected. This allows comparative analysis between different test runs to evaluate if the DiffServ provides sufficient QoS capability.

The selective data retrieval standards assessment focuses on ensuring that recorded data on the test article can be reliably and predictably retrieved by the ground station through the RANS. This is basically a specialization of the end-to-end performance assessment, but is important as a separate evaluation since the data retrieval scenario is expected to be the mainstream of the iNET use cases. Selective data retrieval involves a system on the ground network initiating an RC data delivery protocol connection to the recorder on the test article. The test cases in this assessment involve performing selective data retrieval while active live test data and system management data share the constrained RANS resource. Various DiffServ configurations are evaluated with measures of throughput, throughput variation, latency, and overhead percentage collected at each test case. Understanding not only the achievable throughput but also the throughput variation will help assess concerns about total transfer time predictability.
The TAS/RANS interface throughput assessment focuses specifically on whether the current standards provide sufficient mechanisms for maximizing utilization of the constrained RANS link. Test setups first exercise the RANS with a “realistic” mix of test data, system management data, and general purpose network traffic and measure the overhead associated with those transfers. Based on this, techniques for minimizing this overhead are evaluated. These include proxying requests in order to combine into larger packets, distributed system management, dynamic assignment of measurement groupings to allow only retrieval of needed data (instead of entire originally recorded TmNSDataMessages), point-to-point protocols, and compression. These tests will provide insight into whether the additional complexity required by each approach is justified by the associated gain in RANS network efficiency.

RESULTS OF THE INET STANDARDS ASSESSMENT

Emulators for the generation and consumption of TmNSDataMessages have been created as a basis for test bed EndNode simulation. Two general types of simulators have been developed – a DAU simulator and a recorder simulator. The DAU Simulator generates TmNSDataMessages as defined by the Test Article Standard according to the Latency/Throughput Critical (LTC) Delivery Protocol. The Recorder Simulator consumes TmNSDataMessages as defined by the Test Article Standard according to the LTC Delivery Protocol. In addition, the Recorder Simulator generates and consumes TmNSDataMessages as defined by the Test Article Standard according to the Reliability Critical (RC) Delivery Protocol. In addition, both the DAU and recorder simulators are manageable according to the System Management Standard (SNMP implementation) and can be configured by a Metadata Description Language (MDL) instance document.

Based on configuration files generated by an MDL Instance Document parser, the simulators can be configured with several parameters, including role identification (Role ID), device type, MessageDefinitionID (MDID), destination address, destination port number, and DSCP bits for the setup of TmNSDataMessage generation sequences. Send/receive statistics are collected at a configurable interval for system management purposes.

Additionally, the DAU simulator reads configuration files to aid in traffic shaping of TmNSDataMessage generation. These configuration parameters include maximum message size, maximum message latency, fixed message length constraints, package patterns (i.e. fixed order/fixed number/variable order/variable number), package length, package rate, package payload, among others. The simulators use input files with raw data (circular buffer) to generate the package payloads, thus providing the user a simple way of changing the package payload. Moreover, the DAU simulator is capable of reading a binary file of concatenated TmNSData Messages as a message simulation input file.

The recorder simulator is configured similarly to the DAU simulator, although not all of the traffic shaping parameters are used. LTC Data Messages are consumed based on the MDL Instance Document used to configure the Recorder Simulator. Similarly, the RC Data Source portion of the Recorder Simulator will start based on the MDL Instance Document used to configure the Recorder Simulator. The RC Data Source uses the DataDeliveryControlChannel to
exchange control commands with the RC Data Sink, using Real Time Streaming Protocol (RTSP). The RC Data Source uses a binary file as its input to service RC Data Sink requests. Future capabilities will include live data streaming. Specific requests to the RC Data Source are sent via the DataDeliveryControlChannel using the TmNS_URI by the RC Data Sink. The recorder simulator currently parses the complete TmNS_URI and is able to service the following requests: TmNSmdidlist, TmNSpdidlist, TmNSdestIP, TmNSdestport, TmNSdeliverymdid. Additionally, the recorder simulator is able to handle the following RTSP commands: SETUP, PLAY, TEARDOWN. It also supports the RTSP range header using the Precision Time Protocol (PTP) time range format. Further RTSP functionality will be added as simulator development continues.

Finally, both simulators are manageable via SNMP. Some of the available system management information includes bytes sent/received, messages sent/received, messages lost, rcSessionURI, record mode, record command, among others. In addition, the simulators currently support several SNMP control commands, including enable/disable, recorder command, media erase, to name a few. Further functionality will be added as simulator development continues.

Both the DAU and recorder simulators have been developed in ANSI C and C++ and have been tested on Ubuntu Linux. They can be compiled using the standard “gec” C++ compiler. An Alpha release of the DAU and recorder simulators is available as bootable “live” compact discs (CDs) to allow their use in any Intel machine with a network interface. Standard libraries will be released (source code) for LTC traffic generation and consumption so that other EndNode applications can be built upon them. Figure 1 displays a Wireshark capture of TmNSDataMessages. The TmNSDataMessageHeader, PackageHeader, and PackagePayloads are highlighted.

As previously mentioned, a series of standards initial evaluations are currently in progress using the test bed network described earlier in this paper. In this section, initial results of the end-to-end data delivery initial evaluation will be discussed. More specifically, initial results of the TAS to RANS interface throughput evaluations will be presented.

Tests were conducted to determine how a throughput constrained link would affect throughput of LTC data as well as RC data delivery. In the TmNS, the RANS Radio will introduce the largest amount of latency and throughput constraints, due to its lower bandwidth, when compared to the TAS. Since no real RANS Radio equipment that implements the iNET standards exists yet, the RANS Radio throughput link constraint was emulated (using a network emulator) during these initial evaluations. This was achieved by configuring the network emulator to reduce the available bandwidth in a segment of the network between the sending node and the receiving node to 10Mbps.

The primary goal of these tests was to determine how simultaneous flows of LTC and RC data would share the link capacity as the LTC data rate increased to near maximum link capacity. DAU and recorder simulators were used to generate LTC and RC traffic in the network. Using Wireshark, a capture of the data received at the receiving node was obtained and analyzed to determine the throughput for each of the different data flows. The results are shown in Figure 2, which depicts RC, LTC, and total link utilization for different LTC data rates.
Figure 1. Wireshark Capture of TmNS Data Message

Figure 2. Average Throughput versus LTC Traffic Load
Based on Figure 2, as the LTC data rates increase, the TCP protocol’s congestion control mechanisms reduce the rate of RC data entering the network so as to keep data flow below a rate that would lead to network packet drops. As a result, end-to-end RC data delivery is accomplished reliably, but with added latency, while not impacting the delivery of the latency-sensitive LTC data delivery both from a latency as well as a throughput perspective.

Figure 3 and Figure 4 show instantaneous RC data throughput as a function of time. Clearly, the RC throughput varies significantly over time. The significant variations in instantaneous throughput can be largely attributed to TCP’s flow and congestion controls that try to maximize instantaneous link availability. These time constants need to be considered in the design of any link optimization approach such as the RANS link manager.

![Figure 3. RC Instantaneous Throughput over a 65-second Period](image)

![Figure 4. RC Instantaneous Throughput over a 500-millisecond Period](image)
CONCLUSION

The overall success of iNET depends on proper functioning. But proper functioning not only means that the correct data gets collected and moved to the correct ends, but it also means that it gets there by the time it is needed. As such, the ongoing end-to-end performance assessment being performed will provide information crucial for understanding the current state of the architecture and some of the constraints on the realm of possible applications that are currently supported.

ACKNOWLEDGEMENTS

The work described in this paper would not have been possible if not for the contributions from the many members of the iNET Standards Working Groups. We gratefully acknowledge this effort and the funding and guidance provided by the iNET program.

REFERENCES