KEY COMPONENTS OF THE INET TEST ARTICLE STANDARD

Thomas B. Grace[1], Joshua D. Kenney[2], Myron L. Moodie[2], Ben A. Abbott[2]
[1]Naval Air Systems Command (NAVAIR) **[2]Southwest Research Institute®**
Patuxent River, Maryland **San Antonio, Texas**
thomas.grace@navy.mil, joshua.kenney@swri.org,
myron.moodie@swri.org, ben.abbott@swri.org

ABSTRACT

The integrated Network-Enhanced Telemetry (iNET) Test Article Standards Working Group (TASWG) has developed a standard for Telemetry Network System (TmNS). The introduction of Internet Protocol (IP) networks on test ranges has created the potential for greater flexibility in the telemetry environment. This paper discusses the rationale for particular decisions concerning key components mandated by the standard. Performance implications concerning the mandates of the standard are also described.

As an educational aid, examples of TAS-based processing philosophies and data structures have been constructed. These examples, including sending and receiving messages, are shown to reinforce understanding core concepts of the standard.

KEYWORDS

iNET, Test Article, Networking, Internet Protocol (IP)

INTRODUCTION

A set of standard technologies was required for the Test Article and its components that ensure broad acceptance by the telemetry community, interoperability between vendors of like components, and economical sources of products. The standard technologies were chosen to be well established, with proven performance, continued future market support, and mature technologies with multiple competitive sources of integrated system components and constituent electronic devices (e.g., integrated circuits, wiring, and software).

Internet Protocol (IP), Ethernet, and Base-T/FX are a suite of common protocols used by the Internet and most commercial networks. This set of core technologies can be viewed as a set of layers, with each layer specifying the transmission of data and providing well-defined services to

upper layer protocols. Each layer also uses services from lower layer protocols. Choosing IP, Ethernet, and Base-T/FX as the set of core technologies provides the general approach to data transmission on the bottom three layers of the Open Systems Interconnection (OSI) model: physical, data link, and network. The detailed use of these lower-layer protocols, as well as the use of other protocols in conjunction with and on top of these lower-layer protocols, is specified in the Test Article Standard.

In the Test Article Standard, all Test Article network interfaces use Internet Protocol version 4 (IPv4) as the core network layer protocol. IPv6 does not meet all the goals stated above (i.e. broad acceptance, interoperability, economical sources, well-established, proven, market support, etc.) and the main IPv6 features are not required by the Test Article networks. Figure 1 is a protocol map for the Test Article standards.

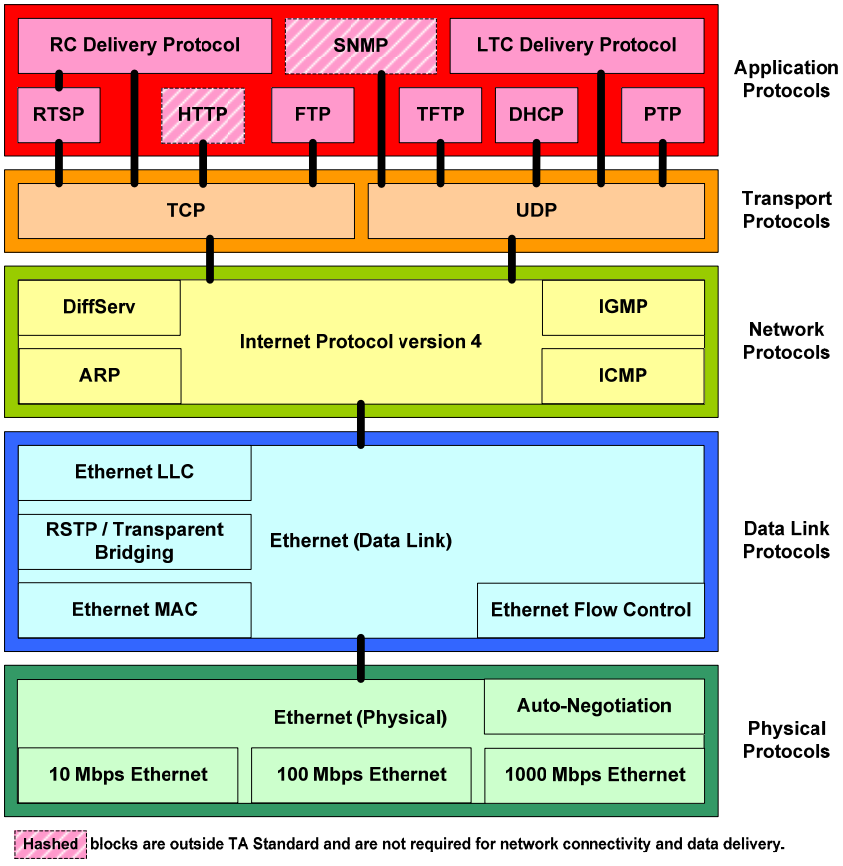

Figure 1. Test Article Standard Protocol Map

CORE NETWORK STANDARDS

The Test Article Standard specifies Ethernet (IEEE 802.3-2005) for the physical layer of the Test Article network. Ethernet defines a number of wiring and signaling standards, supporting 10BASE-T/100BASE-TX/1000BASE-T over copper Unshielded Twisted Pair (UTP) cabling

and 10BASE-F/100BASE-FX/1000BASE-LX over fiber optic cabling for physical layer transmission. The Test Article Standard requires that one or more of the 10 Mpbs, 100 Mbps, and 1000 Mbps standard speeds be implemented. The Test Article Standard also requires auto-negotiation for copper connections and recommends auto-negotiation for fiber connections. These copper and fiber-based standards, along with Ethernet's data link layer protocols, are widely used in commercial networks.

The current Test Article Standard states that connectors and cable media should meet the electrical or optical properties of the standards referenced (i.e. specific Clauses of IEEE 802.3-2005). It is understood that applicability to the operational environment will place additional constraints on the selection of the connectors and cable media. A list of flight-qualified cables that have been suggested by implementers of previous networked telemetry applications is included at the end of the Test Article Standard document.

The Test Article Standard also specifies Ethernet (IEEE 802.3-2005) for the data link layer of the Test Article network. The data link specifications of Ethernet provide common hardware addressing and Media Access Control (MAC) for the transmission of data frames between network devices. Test Article networks must support the frame structure, field definitions, MAC conventions, Logical Link Control (LLC), and flow control defined by Ethernet.

The Ethernet data link protocols in the Test Article Standard are sufficient for basic connectivity between one or more stations over a single Ethernet link, possibly extended in length and/or inter-connected by network repeaters and hubs. This half-duplex, contention-based approach is typically enhanced by bridging or switching multiple Ethernet links, resulting in only two stations per link communicating contention-free in full-duplex. Therefore, switching network devices is also required to conform to the requirements set forth in IEEE 802.1D-2004 for transparent bridging and Rapid Spanning Tree Protocol (RSTP) functionality.

The standards for Test Article network protocols rely upon the requirements for Internet hosts and routers as defined by the Internet Engineering Task Force (IETF). The IETF develops and promotes Internet standards, cooperating closely with other standard bodies and dealing in particular with standards of the Transmission Control Protocol (TCP)/IP suite. The Request for Comments (RFC) documents published by the IETF are intended to provide guidance for vendors, implementing organizations, and users of Internet communication software. They represent the consensus of a large body of technical experience and wisdom, contributed by the members of the Internet research and vendor communities.

IP is the network layer protocol used to logically address hosts and route data packets throughout the network. IP is encapsulated by Ethernet, the data link layer protocol. Test Article network transport protocols use IP to carry data from source host to destination host. IP is a connectionless or datagram internetwork service, providing no end-to-end delivery guarantees. Using Ethernet as the layer below IP enables error-free data transmission, and the layers above IP are responsible for reliable end-to-end delivery service when required. The IP protocol includes a provision for addressing, type-of-service specification, fragmentation and reassembly, and security information. The datagram or connectionless nature of the IP protocol is a fundamental and characteristic feature of the Internet architecture.

Looking to other network layer protocols specified in the Test Article Standard, the Internet Control Message Protocol (ICMP) is a control protocol that is considered to be an integral part of IP, although it is architecturally layered upon IP (i.e., it uses IP to carry its data end-to-end just as a transport protocol like TCP or User Datagram Protocol [UDP]). ICMP provides error reporting, congestion reporting, and first-hop gateway redirection. Another network layer protocol, Internet Group Management Protocol (IGMP), serves is an Internet protocol used for establishing dynamic host groups for IP multicasting. It is noted in the Test Article Standard that switching network devices can use IGMP "snooping" to aid in efficient network utilization.

For the transport layer, the Test Article Standard specifies the TCP and the UDP end-to-end data transmission protocols and options. Like the network layer protocols, TCP and UDP are based on widely accepted IETF RFCs. TCP is the primary virtual-circuit transport protocol for the Internet suite. TCP provides guaranteed, in-sequence delivery of a full-duplex stream of octets (8-bit bytes). TCP is used by those applications needing guaranteed, connection-oriented transport service. UDP, on the other hand, offers only a minimal transport service—non-guaranteed datagram delivery—and gives applications direct access to the datagram service of the IP layer. UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP.

The Test Article Standard also specifies several application layer network services for address resolution, address configuration, name services, and general file transfer. The Address Resolution Protocol (ARP) is the standard method for finding a host's hardware (MAC) address when only its network layer (IP) address is known. Other standard services can configure various parameters (including standard address subnetting) necessary for the clients to operate in an IP network. Although this configuration may be done manually, the Dynamic Host Configuration Protocol (DHCP) allows devices to be added to the network with minimal or no manual configurations. The Test Article network requires a DHCP service, but makes no mandate on which device acts as the DHCP server or which client devices must have DHCP enabled. This allows the choice of static or dynamic IP address assignment. The Domain Name System (DNS) associates various sorts of information with domain names; most importantly, translating computer hostnames into IP addresses needed for network layer delivery.

For file transfer service, the File Transfer Protocol (FTP) is a reliable (i.e. using TCP) and widely-used mechanism that enables the transfer of file-oriented data from one device to another through a network. As such, latency-tolerant bulk data transfers should use the FTP protocol for transferring files between Test Article network-connected devices. Conversely, the Trivial File Transfer Protocol (TFTP) is a simple file transfer protocol, with the functionality of a very basic form of FTP. Since UDP is used for transport, TFTP uses a stop-and-wait-protocol for reliability, where devices send a maximum packet size of 512 bytes and wait for an acknowledgement. This results in slower transfers than FTP since FTP uses TCP sliding windows and larger packets, while TFTP can send only one (smaller) packet at a time and must wait for the specific acknowledgement of each packet. Due to its simplicity (i.e. using UDP), TFTP is easier to implement in a very small amount of memory, making useful for booting computers (such as routers and switches) that do not have any data storage devices.

Along with the network, transport, and application layer protocols, the Test Article Standard also references several IETF RFCs that define the host (RFCs 1122 and 1123) and router (RFC 1812) functionality requirements for Internet-connected nodes. These documents incorporate by referencing, amending, correcting, and supplementing the core protocol standards documents relating to hosts and routers, and they provide options and supporting documentation for the implementation of Test Article network components.

Overall, test software and users need to be able to manipulate network functions (e.g., QoS, performance, configuration, discovery, etc.) defined in the Test Article Standard. The Test Article Standard references the System Management Standard to specify these requirements.

LEVERAGING STANDARDS FOR TMNS-SPECIFIC OPERATION

The Test Article Standard specifies IEEE 1588-2002 for the distributed clock synchronization between components connected to the Test Article network. IEEE 1588 enables sub-microsecond synchronization between network devices using Ethernet. This network time synchronization scheme allows high resolution time stamps at *DataSources* to enable applications of time-correlation of data at *DataSinks*. The Test Article Standard specifies that a master clock may be physically located on any device connected to the Test Article network. Furthermore, devices needing precise synchronization can implement IEEE 1588 slave clock interfaces to use the synchronization service provided by the Test Article network. Depending on application performance requirements, design choices can be made for slave clocks, trading higher accuracy, higher cost hardware implementations against lower accuracy, and lower cost software implementations. For routing/switching devices, the standard also specifies “boundary clock” techniques, or approaches that are interoperable with boundary clocks (e.g. transparency implementations used in some switches), which are intended to eliminate large packet transport latency variations that are typically induced in these types of devices. The Test Article standard also specifies clock “flywheeling” in the absence of masters, recommends 1-pulse-per-second (1 PPS) outputs on devices, and recommends using the Global Positioning System (GPS) interface for external synchronization of a time master.

Several existing network data message structures were examined, and an informal trade study was performed with the conclusion that no single existing message format completely meets the needs of the TmNS. The working group agreed that a single common data message header was needed for moving data messages across the network. Through several drafts and community feedback, the TmNS Data Message Format standard was created as a “best of breed” of existing packet formats while meeting additional requirements for the transport of TmNS data. This universal data message header format captures the strengths and experience of existing formats (Instrumentation d'Essais des Nouveaux Avions [IENA], Ch. 10, Data Acquisition Recorder [DAR]), transports and supports existing formats, protects existing data processing and Data Acquisition Unit (DAU) software, promotes a common military and commercial format, encourages lower DAU and data processing costs, and provides scalability for current and future uses.

The *TmNSDataMessage* is a structure that is independent of transport. If sent on a network, typical network and transport headers are pre-pended (e.g. UDP/IP). A *TmNSDataMessage* is composed of a *TmNSDataMessageHeader* and a *TmNSDataMessagePayload*, as illustrated in Figure 2.

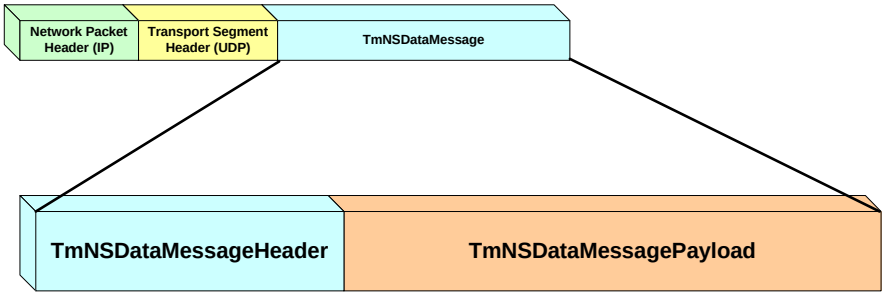

Figure 2. TmNSDataMessage

The *TmNSDataMessageHeader* assists with data transport and provides information about the *TmNSDataMessagePayload*. The Test Article Standard defines the *TmNSDataMessageHeader* containing the fields and associated bit-widths as outlined in Figure 3.

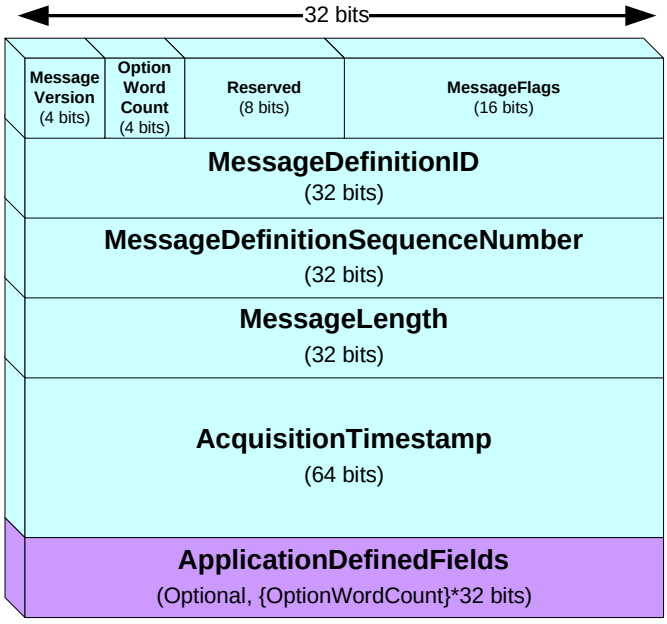

Figure 3. TmNSDataMessageHeader

The *TmNSDataMessagePayload* is made up of *Packages* (see Figure 4). The variable size of the *TmNSDataMessagePayload* allows control of the size and latency of *TmNSDataMessages*. *DataSinks* and other consumers of *TmNSDataMessages* can determine which *Packages* to expect in a particular *TmNSDataMessagePayload* by using the *MessageDefinitionID*. The *MessageDefinitionID* specified in the *TmNSDataMessageHeader* serves as a reference to

Package(s) format, content, and ordering in the *TmNSDataMessagePayload*. For details on how this information is defined, refer to the Metadata Standard Document.

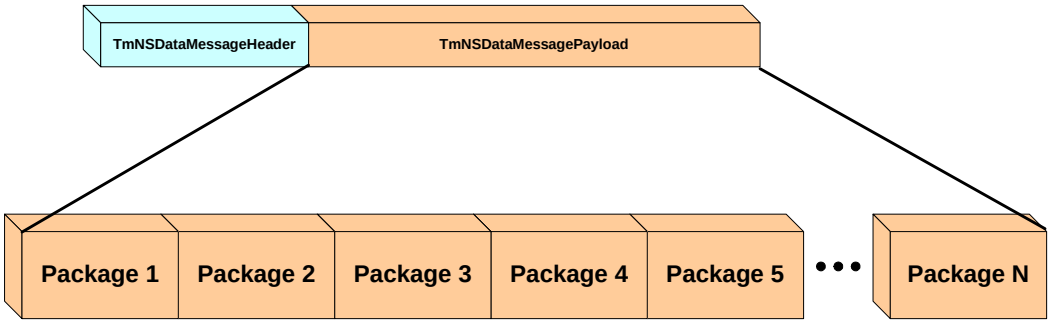

Figure 4. TmNSDataMessagePayload with Packages

A *Package* where a *PackageHeader* and *PackagePayload* are present is illustrated in Figure 5. To achieve *Package* structure interoperability, *TmNSDataMessages* must either use the standard *PackageHeader* defined in the Test Article Standard (Figure 6) or a *PackageHeader* completely described by metadata. All *EndNodes* that produce *TmNSDataMessages* must have, at the minimum, the ability to generate the standard *PackageHeaders*.

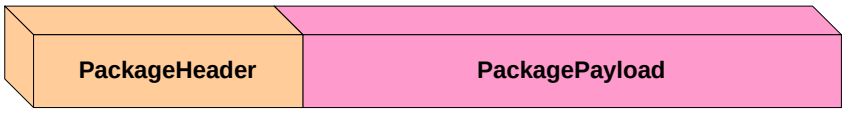

Figure 5. Package

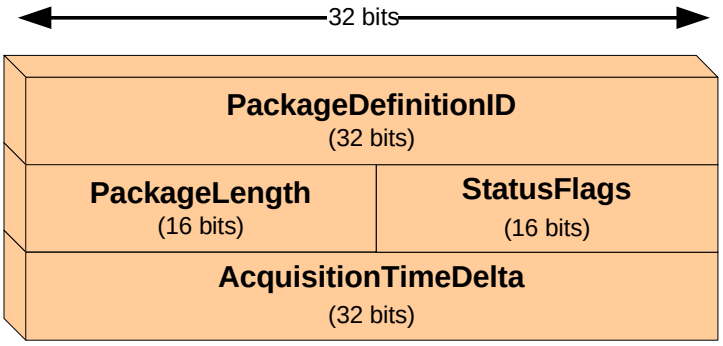

Figure 6. PackageHeader

Many possibilities exist for how *Packages* may be placed into *TmNSDataMessagePayloads*. Rather than over-specify, and possibly prevent users from being able to meet their mission needs, reasonable flexibility is allowed by Test Article standards. Responsibility is shifted to the users to impose local rules that meet the particular needs of their range. However, there are several rules that TAS networks shall follow to ensure a sufficient level of network transport

interoperability. First of all, *MessageDefinitionSequenceNumbers* shall not repeat or be generated out of order for a particular *MessageDefinitionID*, including when two or more *DataSources* generate *TmNSDataMessages* with the same *MessageDefinitionID*. This rule was defined since *DataSinks* might want to be able to detect lost *TmNSDataMessages*. So, a *DataSource* cannot repeat or generate *MessageDefinitionSequenceNumbers* out of order for any single *MessageDefinitionID*. Furthermore, more than one *MessageDefinitionID* may share the same multicast address for network transport. This rule was defined since there are a limited number of multicast addresses that can be practically subscribed to by any one IP stack instance running on a typical embedded processor. Finally, a *DataSource* (or group of *DataSources*) may send a sequence of *TmNSDataMessages* with the same *MessageDefinitionID* to multiple destination addresses. This rule was defined since modern Ethernet hardware can use multicast addresses to efficiently (and almost automatically) reduce the amount of load on *DataSinks* (and network switches) by only sending data to subscribed nodes.

The Test Article Standard specifically defines two protocols for the delivery of *TmNSDataMessages*. The first of these is the Latency/Throughput Critical (LTC) delivery protocol. LTC uses UDP/IP to deliver sequences of *TmNSDataMessages* to multicast addresses. Delivery to unicast and broadcast addresses is also allowed. Since UDP is inherently connectionless, considerations need to be made when using IP to support the latency and throughput guarantees. When planning synchronous-like data transfer over asynchronous network transport, some performance-enhanced networking mechanisms must be designed into the network fabric to ensure the timely transfer of latency-critical data and maintain sufficient bandwidth for throughput-critical data flows. To ensure optimal performance with IP multicast, Test Article network switches and/or routers should partition the network such that only two devices (i.e., the switch/router and the end device) are sharing a specific Ethernet link. In this setup, no Ethernet frame collisions occur and each pair of network devices can operate in full-duplex at full link speed without contention from other devices. Additionally, switches/routers with sufficiently large queuing buffers should be chosen to handle the “burstiness” of the routed traffic.

Even with this level of forwarding determinism, which is inherent to most modern implementations, composite throughputs on each link of telemetry networks should be planned to not exceed the available bandwidth. This planning should be realized using the anticipated send rates of each of the *MessageDefinitionID*, along with the pool of available multicast addresses, to assign multicast addresses in proportions that do not overflow available line rates and buffer queues. This planning should be automated in an application for producing setup files, but may be done manually based on the anticipated network topology. Metadata should be generated and shared between *DataSources* and *DataSinks* for associating *MessageDefinitionIDs* with their multicast groups.

The standard *TmNSDataMessage* structure includes *MessageDefinitionSequenceNumbers* to enable the detection and reporting of transport reliability through system management. Applications may also use *MessageDefinitionSequenceNumbers* to reorder flows of a particular *MessageDefinitionID* at *DataSinks*. The standard *TmNSDataMessage* structure also includes and *AquisitionTimestamp* field for correlation of LTC delivered data at *DataSinks*. 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 that would delay data outside of the latency performance bound. IP multicast packets will also include QoS markings in the IP header, as defined in the Test Article Standard, to enable the prioritization of multicast data across network devices.

The second protocol for the delivery of *TmNSDataMessages* is the Reliability Critical (RC) delivery protocol. This protocol is intended for the delivery of *TmNSDataMessages* when reliability constraints are more important than latency or throughput constraints. RC 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 a reliable protocol (i.e. TCP). The reliable protocol supports the error-free delivery of a data stream 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.

Stream-oriented transfer of data with RC consists of a data transfer between a *DataSource* and a *DataSink*. The data transfer is initiated, controlled, and concluded under the supervision of the *DataSink* via a "Control Channel." The *DataSink* specifies parameters defining the *DataSink* destination IP address and TCP port, transfer data set (*MessageDefinitionID* [MDIDs]), and other aspects of the data transfer via commands issued to the *DataSource* via the Control Channel. The *DataSink* establishes a "Data Channel" TCP connection to the specified *DataSink* address and port. The *DataSource* transmits the parameter-specified data set (often followed by an "end-of-data" indication) via the TCP connection to the *DataSink*.

Real Time Streaming Protocol (RTSP) was chosen for the RC control channel by the TASWG after studying Simple Network Management Protocol (SNMP) and Hypertext Transfer Protocol (HTTP) as alternatives. It was determined that RTSP provided the required control channel methods and parameters with the least customization of the protocol. TCP was chosen for the reliable transfer of data because TCP provides the required error correction, congestion control, and error recovery mechanisms for reliability critical data.

The goal of designing for quality of service (QoS) within the Test Article Standards context is the allocation of Test Article network bandwidth and technology sufficient to meet the latency bounds and packet loss requirements of network traffic with differing quality of service requirements. Proper network engineering and administration are a fundamental requirement for successful operation of a network-based telemetry system. The telemetry network designer must anticipate composite data rates across Test Article network interfaces and subnets. Adequate bandwidth, switching capacity, and packet processing policies must be sufficiently provisioned at each interface and across the network to meet the latency bounds and packet loss requirements of a particular implementation. The telemetry network designer should use a combination of telemetry traffic analysis, scheduling, and network over-provisioning as the primary tools for meeting quality of service requirements.

Network over-provisioning is the practice of allocating additional bandwidth capacity to a segment or interface beyond the amount required to carry the computed average composite data rate. The excess bandwidth provides dynamic range within the capacity of the given Ethernet segment or interface sufficient to handle the traffic variance of a composite traffic. Adequate over-provisioning can eliminate or greatly reduce the need for QoS protocols. The Test Article network engineer should employ over-provisioning to the maximum extent practical before investing in the complexity of other techniques. In fact, sufficient capacity must be present for QoS protocols to provide substantial benefit during periods of network congestion. Application of QoS protocols is appropriate when infrequent and random composite data rate on a subnet or interface may cause undesirable latency, latency jitter, or packet loss. QoS protocols can not compensate for a persistent lack of adequate bandwidth.

The Test Article Standard specifies the widely used DiffServ QoS protocols as an additional tool to meet QoS requirements when traffic engineering measures may not be sufficient to address momentary congestion within the Test Article network. The QoS protocols implement a set of per hop behaviors (PHBs) at the outputs of peripheral interfaces. The PHBs define a set of policies enforced at input and output queues of peripherals 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 (vNET, radio frequency network [rfNET], ground station [GS]).

CONCLUSION

Now that the initial Test Article Standard has been documented, we have been able to leverage the content to create a general library for the creation and consumption of *TmNSDataMessages*. This library has been created as a basis for test bed *EndNode* emulators. These *EndNode* emulators take command-line options for *DataSource*, *DataSink*, destination address, MDID, and DiffServ Codepoint (DSCP) bits for the setup of a *TmNSDataMessage* sending/receiving sequence. Statistics are printed to the screen periodically. These statistics include *TmNSDataMessages* sent, bytes sent, time-windowed (on configured printing rate) *TmNSDataMessages* per second, time-windowed (on configured printing rate) bits per second, MDID value, DSCP value for sent *TmNSDataMessages*, and *MessageDefinitionSequenceNumber* drops for received *TmNSDataMessages*. It is envisioned that test bed assessments using these *EndNode* emulators, as well as real hardware from vendors, will contribute to the process of the further maturation of the Test Article standards.

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 Group. We gratefully acknowledge this effort and the funding and guidance provided by the iNET program.