IRIG 106 Chapter 10 vs. iNET Packetization: Data Storage and Retrieval

Charles H. Jones, PhD
Edwards Air Force Base, California
Charles.jones@edwards.af.mil

ABSTRACT

The approach to recording data during Test & Evaluation has evolved dramatically over the decades. A simple, traditional approach is to pull all data into a PCM format and record that. A common current approach is to record data in an IRIG 106 Chapter 10 compliant format that records different forms of data (bus, discrete, video, etc.) in different channels of the recorder or exported data file. With network telemetry on the horizon, in the form of the integrated Network Enhanced Telemetry (iNET) standards, much of the data will be transported in iNET messages via Ethernet frames. These messages can potentially carry any type of data from any source. How do we record this data? Ultimately, no matter how the data is stored, it must be translated into a form that can be used for data analysis. Data storage forms that are conducive to this analysis are not necessarily the same that are conducive to real time recording. This paper discusses options and tradeoffs of different approaches to incorporating iNET data structures into the existing T&E architecture.

KEYWORDS

Recording standards, data analysis, IRIG 106 Chapter 10, iNET

INTRODUCTION

The IRIG 106 Chapter 10 [1] standard (to be referred to as Chapter 10) is usually referred to as a "recording" standard, but part of it can also be considered a standard for a self-describing data file format. The intent of the standard is to allow reading the data by anyone without regard to what vendor's equipment it was recorded on. A quick scenario is that the data is recorded, a file is extracted or exported from the recorder, the file is put on a computer, and analysis is done using this data with third party software. The conceptual layout of a Chapter 10 file is to store each data source in its native format in separate channels. Channel 0 stores the metadata, using IRIG 106 Chapter 9 Telemetry Attributes Standard (TMATS) [1], which describes the data in the other channels; this is the "self-describing" part of the file.

The integrated Network Enhanced Telemetry (iNET) program has established an Ethernet based instrumentation system that packetizes data. These packets can be very complicated and it is possible to store any data stored in a Chapter 10 file in these packets. The data is described via metadata using the iNET Metadata Description Language (MDL) [2]. Since it is fairly clear that packetized networked data acquisition, most likely via iNET, is the way of the future, some questions arise. How do you store data in packets for later data retrieval? Do you incorporate the packet structure into the Chapter 10 approach or develop some other mechanism? More generally, does packetizing data change the work flow of data collection and analysis significantly enough to justify replacing Chapter 10 with a different recording standard?

There are several issues to look at to answer this question, but ultimately the answer requires resolving the classic complexity conundrum of time-space tradeoff, which is always application dependent.

DATA STRUCTURES

The basic application is to collect data from multiple sensors (in the most generic meaning of the term “sensor”) and transforming that into a data structure that can be used for analysis. A major complication is that the data from the sensors is coming across multiple physical interfaces. Further, the data is encoded in myriad ways – everything from individual discrete bits, to standard floating point numbers, to video streams. Because of hardware and logistics, a practical approach is to record data in a form that is conducive to real time testing and then convert to a form conducive for analysis.

Chapter 10 defines a format that satisfies this real-time recording requirement. The basic design of Chapter 10, as illustrated in Figure 1, is to use multiple channels. Channel 0 contains information compliant with TMATS which describes the data in all the other channels. The other channels contain information in a single data format, e.g., PCM[1], MIL-STD 1553, Video, etc. (Chapter 10 identifies 16 different data formats – including Ethernet.)

For illustrative purposes, measurements of two parameters, M1 and M2, are indicated in different channels of Figure 1. These illustrate the most basic form of data. That is, data collected periodically at so many samples per second. The problem with this data structure is that an analysis is likely to require data, say M1 and M2, from different data sources. It is thus necessary to extract these measurements into a common data structure for analysis. Usually this requires time correlating the data points. Figure 2 illustrates the conceptual data structure for analysis.

At their core, iNET messages [3] use a structure to facilitate transport over Ethernet – similar, from a functional point of view, to MIL-STD 1553 or other bus messages. Since they are encapsulated in Ethernet/IP frames, they are wrapped with standard Ethernet and IP headers and footers. Then each iNET message has a header. Each iNET message can have one or more packets which contain the actual data. Each packet has an (optional) header as well. Each packet can contain its own type of data – to include,

[1] Because of its common usage, we will refer to “PCM” or “PCM Formats”. The iNET program is using the more generic term “Serial Streaming Telemetry (SST)”.

theoretically, any data type even beyond those identified in Chapter 10. Thus, you might have PCM in one packet and MIL-STD 1553 in another packet all in the same iNET message. The actual data points are thus buried inside the packets. This is illustrated in Figure 3, including a continuation of our illustration of measurements of parameters M1 and M2. The definition of what is in each packet is contained in an MDL XML instance which serves the same functional purpose as the Channel 0 TMATS file of Chapter 10.

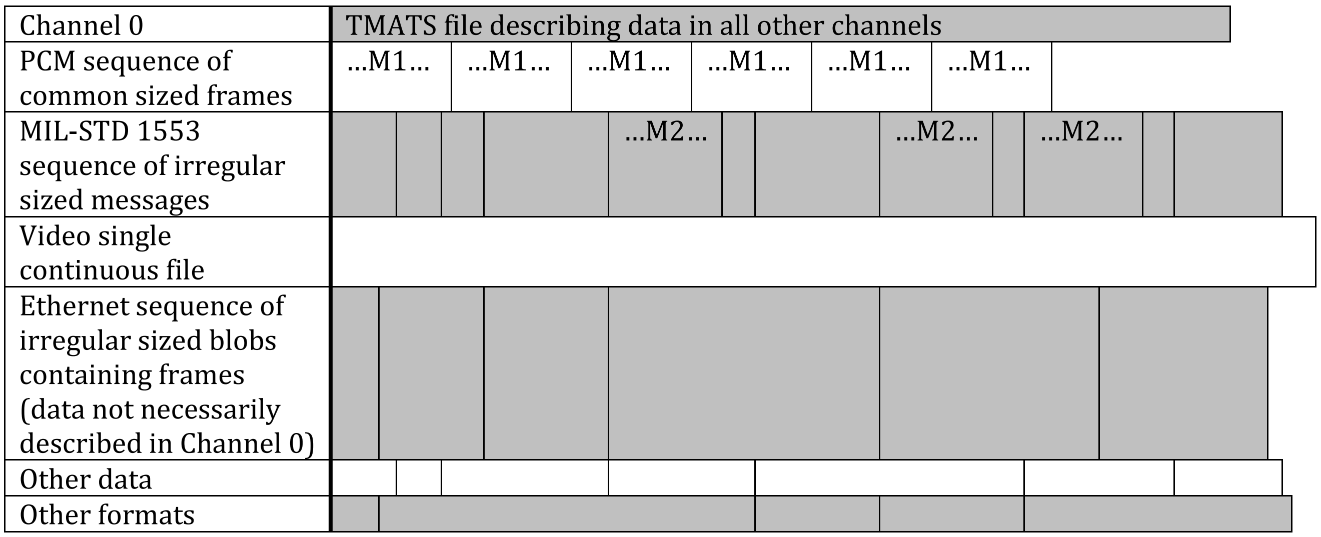
Channel 0
TMATS file describing data in all other channels
PCM sequence of common sized frames
...M1...
...M1...
...M1...
...M1...
...M1...
...M1...
MIL-STD 1553 sequence of irregular sized messages
...M2...
...M2...
...M2...
Video single continuous file
Ethernet sequence of irregular sized blobs containing frames (data not necessarily described in Channel 0)
Other data
Other formats

Figure 1 Conceptual Layout of a Chapter 10 File

Time	M1	M2
T1	Data Point	Data Point
T2	Data Point	Interpolated Data Point
T3	Data Point	Data Point
T4	Data Point	Interpolated Data Point
T5	Data Point	Data Point
T6	Data Point	Interpolated Data Point

Figure 2 Conceptual Layout of Analysis Data

Ethernet/IP Header	iNET Message Header	iNET Packet Header	Packet (Data) …M1… …M1…	iNET Packet Header	Packet (Data) …M2… …M2...	. . .	Ethernet/IP Footer

Figure 3 Conceptual Layout of iNET Message

Let us consider some of the differences between Chapter 10 and iNET messages.

Chapter 10 channels are intended to store large data structures for delivery to (after test) an analysis system. For example, a single channel might store a total bus capture of a MIL-STD 1553 bus for an entire test. Certainly, a channel containing video would be expected to include an entire video clip.

Because of size constraints or because of logical sectioning of a test flight, you might split a full bus capture into more than one channel, but that's not really the design intent and would be done for specific logistical reasons. In contrast, iNET messages are intended to transport data over Ethernet in real time to whatever device needs to export or import data. This means that you would not expect a single iNET message to contain a full bus capture for an entire test. (For efficiency reasons, you also would not expect an iNET message to contain a single data point or MIL-STD 1553 message; lots of small Ethernet frames increase overhead and decrease throughput.)

A significant design requirement for Chapter 10 was to make the data file "self-describing". This is why channel 0 contains a TMATS file. Channel 0 describes all other data in every other channel. In contrast, the very nature and purpose of messages does not allow for this type self-description – too much overhead for real time transport. Thus, an MDL file is maintained somewhere else and has to be accessed in order to decode an iNET message. This increases configuration management.

To be fair, there is an aspect of both structures that is not functionally different – the use of headers. Although Figure 1 does not stress header structure for Chapter 10 the way Figure 3Figure 2 does for iNET messages, there are plenty of headers to be found in Chapter 10. Each channel is likely to have a header and, for example, MIL-STD 1553 messages have headers. Headers are just a necessary part of data storage architectures to allow access of data in the structures.

An aspect of iNET messages is that, in order to obtain, say, a full bus capture, you have to reconstruct it from multiple iNET messages. On the other hand, Chapter 10 channels are constructed piecemeal throughout a test as well.

HARDWARE ARCHITECTURE

To help focus on the real-time functional differences between Chapter 10 and iNET messages, let us consider the progression of hardware architectures for real-time data acquisition systems. The following diagrams illustrate this progression. (Although the assumption is that any PCM Formatter is providing a PCM stream to a telemetry transmitter, for the sack of simplicity, transmitters are not included in the figures. Further, these are *notional* architectures and do not necessarily reflect anything that ever was or ever shall be implemented.)

The traditional approach, as illustrated in Figure 5, was very PCM centric. Everything – bus data, discretes, individual digitized analog measurements, etc. – went into a PCM format and this format was recorded. As recorder technology matured, especially via the electronics explosion, it became possible for a recorder to take in data directly rather than through a PCM formatter (Figure 4). Thus it could do a full bus capture while the PCM formatter stripped off just what it needed. In an extreme case, a recorder could take in data sources not telemetered. This required a robust recorder standard – Chapter 10. The future, as conceived by iNET and illustrated in Figure 6, is for all devices to communication over a network. Thus both PCM formatters and recorders receive only the data they need; although it is still envisioned that the PCM stream will be recorded.

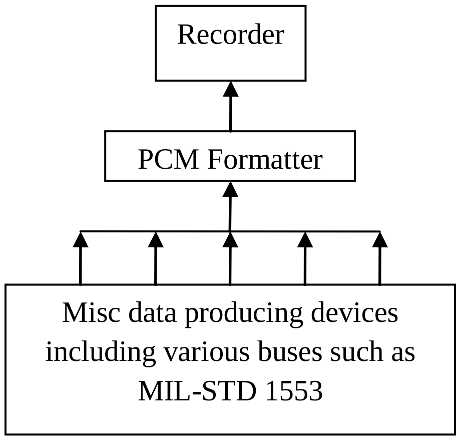
Recorder
PCM Formatter
Misc data producing devices including various buses such as MIL-STD 1553

Figure 5 Notional Traditional PCM Centric Architecture

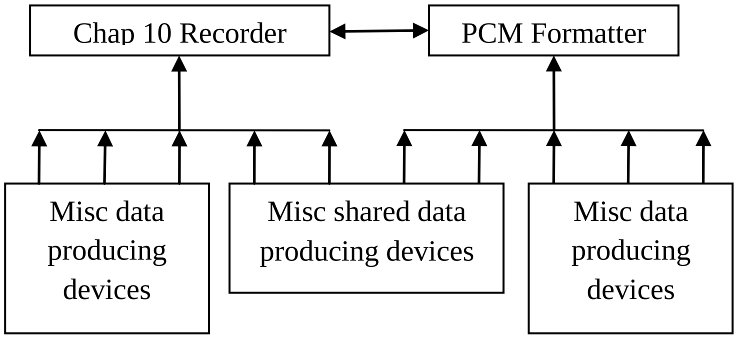
Chap 10 Recorder
PCM Formatter
Misc data producing devices
Misc shared data producing devices
Misc data producing devices

Figure 4 Notional Chapter 10 Centric Architecture

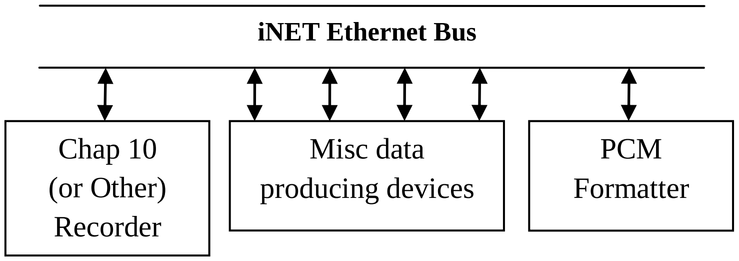

Figure 6 Notional iNET Centric Architecture

STORAGE AND RETRIEVAL ISSUES

The transition from a PCM centric architecture to a Chapter 10 centric architecture increased the complexity of data storage. The further transition to a network centric architecture opens the possibility for increased functionality via the ability to retrieve data off the test article during the test beyond the standard telemetry stream. (It also can facilitate more onboard data processing, but that's not the focus of this paper.) The term "parametric data retrieval" is being used to describe the retrieval of individual measurements off the recorder in real-time (or at least during the mission rather than after the mission). A particular application of this is PCM backfill. As is well known, PCM tends to have drop outs. By retrieving the recorded version of the data over the network, it is envisioned that a user can be presented with a pristine image of a measurement over time without dropouts. This changes the nature of the recording so that the complexity of retrieving data (partly as a consequence of how the data is stored) becomes an issue.

Consider, for a moment, the difference between on-board parametric retrieval and post flight extraction, including time correlation, of measurements for post-test analysis. They are very similar and the difference is mostly due to application. Post-test analysis requires many parameters to be extracted covering relatively long time periods. On-board parametric retrieval is likely to only require one or two parameters to be extracted over a short time period. But both require the basic ability to delve into the recorded data structure down to the individual measurements.

As we start to analyze performance we need to reflect on time-space tradeoffs. PCM represents a simple approach to data storage that requires very little overhead (minimal space cost). However, retrieving data is slow because the data must be searched sequentially (high time cost). An indexed database structure (which is common for final data analysis) is the complete opposite. A database uses much more space for the indexing information but allows random (e.g., fast) access to the data once you have traversed the indexing structure. Chapter 10 is, perhaps, in the middle. There is some indexing to allow for storage by channel, but once the channel has been identified, retrieval is still sequential and thus slow. From a storage point of view, an indexed database structure has a high time cost since the indexing must be created and traversed during storage in contrast to PCM or Chapter 10 which simply append data as they receive it. Note that it would be theoretically possible to provide a detailed indexing structure for Chapter 10 but someone designing an indexed data structure from scratch would take an approach much more like a relational database than Chapter 10. A Chapter 10 indexing scheme would be cumbersome at best.

There is another data retrieval application that we will refer to as "mission reconstruction". On occasion it is convenient to be able to reconstruct the entire mission to really analyze what happened or to troubleshoot the data acquisition process itself. This can be at two levels: reconstructing the activity on a single bus or subsystem or reconstructing all data acquisition activity. This is something that Chapter 10 excels at. By recording each data stream in its native format, it is relatively trivial to reconstruct this activity. Reconstructing this from an indexed data structure is not as trivial.

STORING iNET PACKETIZED DATA

The following seem to be the primary approaches that could be taken to store data transmitted over an iNET network:

1. Store iNET Entities in a Chapter 10 Channel providing MDL in Channel 0
    a. Ethernet frames containing iNET messages
    b. iNET messages containing packets
    c. iNET packets
2. Remove iNET packet and message structure, reconstruct original data structures and store in standard Chapter 10 channels.

This third option, Indexed Centric Storage, can be generalized to all data from any source.

3. Totally divorce the data from hardware or format origin and store indexed data.

The approaches are listed in order of complexity of decomposition prior to storage. Storing Ethernet frames requires no decomposition whereas choice 3 requires complete decomposition down to individual measurements. These thus represent trade spaces in terms of where the processing takes place – during storage or during retrieval.

Some may ask why you would even consider storing the Ethernet frames. This would be necessary to perform a complete mission reconstruction. In particular, this could be useful for analyzing network performance or security breaches.

PERFORMANCE ANALYSIS AND TRADEOFFS

Our approach to tradeoff analysis is to define data transformation functions in Table 1for different steps in the process and for different storage and retrieval techniques. Thus, for example, S(B) represents the process of segmenting bus data into iNET packets. Similarly, $S^{-1}(B)$ represents the inverse process of decomposing the iNET packets and retrieving the original bus data B. In order to not introduce unnecessary symbols some low level functions are not formally defined. Thus, for example, we do not introduce B(M) to indicate encoding a measurement into a bus data structure. However, we do introduce $\mathcal{R}_{10}(M)$ to represent a direct reconstruction of a measurement M from a Chapter 10 recorder. B(M) will happen regardless of which approach we take, whereas $\mathcal{R}_{10}(M)$ makes a difference in our analysis.

A key part of the analysis revolves around S(B) and its inverse. It is necessarily the case that data streams will be segmented when transported over the network; an hour long PCM stream will not be sent as a single Ethernet frame. On the other extreme, Ethernet throughput suffers significantly if every packet is very small; a single Ethernet frame will not contain a single PCM subframe. Within some level of tolerance, there are optimal Ethernet frame sizes. This and time synchronization concerns are reasons why the iNET message structure was designed to include multiple packets. A simple approach might be to determine the optimal packet size and then fill a single iNET message with that amount of data from a single source. However, this is likely to cause data from multiple sources to arrive at the recorder (or other destination) at very disparate and irregular times – this is not desirable.

This is the long winded explanation for the note associated with $S^{-1}(B)$. If an iNET message contains a single packet of a single data source, then the decomposition is not significantly more complex than deconstructing PCM, (i.e., $P^{-1}(B)$). However, this isn't going to happen. Packetized data is going to be more complex because a single iNET message will contain multiple packets from different data sources.

There is also some focus on the distinction between $R^{-1}_{10}(B)$ and $\mathcal{R}_{10}(M)$. It is possible to retrieve a measurement from a Chapter 10 recorder by first reconstructing the bus data. But it is also conceivable to retrieve that measurement from a Chapter 10 recorder directly without first reconstructing the entire bus data. This is introduced to discuss parametric retrieval and to contrast Chapter 10 with the same functional difference represented by the indexed versions $R^{-1}_{I}(B)$ and $\mathcal{R}_{I}(M)$.

Table 1 Function Definition for Complexity Analysis

Function	Definition	Complexity	Notes
M	Sequence of Measurements	N/A	M1 (or M2) as in Figure 1 would be an element of M
B	Original bus format	N/A	Any work constructing this is outside the scope of this analysis. M in B
P	PCM format	N/A	Any work constructing this is outside the scope of this analysis. M in P.

S(B)
S(P)
	B or P Segmented – e.g. iNET packets	Easy to hard	Easier if single bus going into single message structure

P(B)	Bus data in PCM	Easy	Well defined word-frame insertion

S^{-1}(B)	B	Easy to Complex	If iNET messages contain many packets with multiple data types, this is complex
P^{-1}(B)	B	Easy	Well defined word-frame retrieval
R_{10}(B)	Recorded version of B in a Chap 10 channel	Medium	Some configuration management of channels
R_I(B)	Recorded and indexed version of B	Very Complex	(not chap 10) for real-time parametric retrieval
R(P)	PCM recorded directly on traditional recorder	Easy	(not chap 10)
$R^{-1}{}_{10}$(B)	Reconstructed B off chap 10 recorder	Easy	Just read a channel
$R^{-1}{}_I$(B)	Reconstructed B off indexed recorder	Complex	Requires extra metadata to maintain relation of all M in B.
R^{-1} (P)	Reconstructed P off non chap 10 recorder	Easy	Standard read
$\mathcal{R}_{10}$(M)	Read M off Chap 10 recorder	Slow	Do not reconstruct B. Difficulty is sequential read.
$\mathcal{R}_I$(M)	Read M off indexed recorder	Easy	Do not reconstruct B. Whole point of indexing.

Table 2 captures the main analysis. The functions introduced above allow for a concise representation of the transformations. A different conceptualization is to think of the processes through which the data go from generation to storage to reconstruction. For example, the function $P^{-1}(R^{-1}(R(P(B))))$ simply captures the process of encapsulating bus data into a PCM stream [P(B)], recording the PCM stream [R(P(B))], reading the PCM stream from the recorder [R^{-1}], and extracting the original bus data from the PCM stream [P^{-1}].

It would have been possible to include a fourth column in
Table 2 and to make a distinction between extracting data for real time purposes and extracting data for post-test analysis. But, from the point of view of our analysis, there is no real distinction between them and they are both represented by the Parametric Retrieval column.

Here is a simple summary of the analysis. The traditional PCM centric approach is relatively simple, but measurement retrieval is slow and, in general, does not facilitate more advanced applications. Chapter 10 is slightly more complex and certainly facilitates mission reconstruction. However, Chapter 10 is pretty lousy at real-time parametric retrieval. An indexed storage approach certainly facilitates real-time parametric retrieval, but is lousy at mission reconstruction. An indexed approach also significantly increases the real-time storage complexity.

Table 2 Complexity Analysis by Application

Application Approach	Mission Reconstruction (Retrieve P or B)	Parametric Retrieval (Retrieve M)
PCM Centric	$P^{-1}(R^{-1}(R(P(B))))$ Easy Storage Easy retrieval	$P^{-1}(R^{-1}(R(P(B))))$ Easy Storage Slow retrieval
Chap10 Centric	$R^{-1}{}_{10}(R_{10}((B)))$ Medium storage Easy retrieval	$\mathcal{R}_{10}(R_{10}((B)))$ Medium storage Slow retrieval
iNET Centric: Store iNET entities	$S^{-1}(R^{-1}{}_{10}((R_{10}(S(B)))))$ Medium storage Complex reconstruction	$S^{-1}(R^{-1}{}_{10}((R_{10}(S(B)))))$ Medium storage Complex retrieval
iNET Centric: Deconstruct iNET messages	$R^{-1}{}_{10}\ (R_{10}(S^{-1}(S(B))))$ Complex storage Easy Reconstruction	$\mathcal{R}_{10}(R_{10}(S^{-1}(S(B))))$ Complex storage Slow retrieval
Indexed Centric	$R^{-1}{}_{I}(R_I(S^{-1}(S(B))))$ Complex Storage Complex reconstruction	$\mathcal{R}_I(R_I(S^{-1}(S(B))))$ Complex storage Easy retrieval

EXISTING INFRASTRUCTURE AND STANDARDS

As we move towards a networked instrumentation system and make decisions about how to store data, there is a very real and practical issue that must be considered. The development of the IRIG 106 Chapter 10 standard has involved a great deal of effort by many people. Chapter 10 has also been adopted around the world. A great deal of money has been spent developing not only the recorders themselves, but the analysis tools as well. It is a successful standard with a relatively mature support infrastructure. As useful as an indexed storage approach is for real-time parametric, do we just throw away this investment?

A key component of evolving towards indexed storage would be a standard for the indexing structure. One of the crucial design requirements of the Chapter 10 standard was that the delivered file must be readable by anybody without any proprietary constraints. Developing such a standard is likely to be as, or more, difficult than the development of Chapter 10. (An historic note is that such a standard, the Instrumentation Data Exchange (IDX) was proposed in [4].)

DISCUSSION

There are distinct tradeoffs between storage methods dependent on the application. Certainly for real-time parametric retrieval an indexed storage method provides a superior response. Such a mechanism would also enable greater onboard processing with many applications yet to be thought of. On the other hand, there are very valid reasons why the Chapter 10 standard was developed the way it was and there is an investment in Chapter 10 technology that will not be readily discarded.

One solution to this conundrum is to allow recorder manufacturers to deal with the real-time parametric retrieval problem however they want. Perhaps processing power is enough to retrieve data from a Chapter 10 structure in a timely fashion. Or, if they want to implement an indexed storage structure they can do so. These different methods could be accommodated by an interface standard that simply requests the data parametrically without specifying how the data is stored. However, the manufacturers would still be on the hook to provide the final, post-test, data in a Chapter 10 format.

In terms of how to store data transferred in Ethernet based iNET messages, it is not really clear what the answer is. But perhaps the solution is better stated in terms of reconstruction requirements. Do we want to establish a standard requirement of being able to reconstruct iNET messages or to reconstruct the Ethernet frames? We could thus, again, allow manufacturers to record data as they want as long as they can satisfy our reconstruction requirements. In terms of the final Chapter 10 compliant delivered file, it might be possible to store the Ethernet frames along with the original bus based channels. This would potentially double the storage space, but ground based storage may very well accommodate this.

One of the biggest conflicts with the suggested approach is physical retrieval of the data storage medium. If the data is stored onboard in a Chapter 10 file, then it is possible to simply remove the physical storage medium and transfer it to a ground based computer. Without a standard for an indexed approach, this is not possible.

Just for the sake of a reality check, as iNET is deployed, most likely data acquisition systems will be hybrids. It is unlikely that test articles will be completely upgraded to the architecture in Figure 6. New test articles might have such a system installed but even then, there are potentially good reasons for maintaining direct bus connections to the (Chap 10) recorder. Part of the point here being that it will be a very long time, if ever, that the entire system on every test article will completely transition to Ethernet; buses like MIL-STD 1553 and other operational buses are not going away any time soon. Perhaps this, in and of itself, is enough to maintain Chapter 10 recorders. It is very much designed for multibus systems.

The intent of this paper was to analyze tradeoffs. Unfortunately, these do not readily lend themselves to a clear and obvious universal solution. So, as iNET becomes a reality, the community needs to start looking at how recorders are going to integrate with iNET.

REFERENCES

[1] Telemetry Group, Range Commanders Council, IRIG Standard 106-11, Telemetry Standards, Secretariat, Range Commanders Council, U.S. Army White Sands Missile Range, NM, (2011).

[2] integrated Network Enhanced Telemetry (iNET), Metadata Description Language (MDL) Standard (Draft)

[3] integrated Network Enhanced Telemetry (iNET), Test Article Standard (Draft)

[4] C. Jones, *Instrumentation Data Exchange (IDX)*, International Test & Evaluation Association Workshop, Lancaster, CA, (2002) (Paper available upon request.)