NEW MONITORING PARADIGMS FOR MODERN AVIONICS BUSES

DAVE BUCKLEY

ACRA BUSINESS UNIT, CURTISS-WRIGHT CONTROLS AVIONICS & ELECTRONICS

ABSTRACT

In modern aircraft there is a proliferation of avionics buses. Some of these buses use industry wide standards such as ARINC 429 or AFDX while others are based on proprietary protocols. For many of the newer bus types there can be thousands of parameters on each bus. In a distributed data acquisition system the flight test engineer needs to record all of the data from each bus and monitor selected parameters in real time. There are numerous different approaches to acquiring, transmitting and recording data from avionics buses. In modern FTI there is also a proliferation of standards for recording and transmission including IRIG 106 Chapter 10, iNET and IENA. In this paper some common approaches to bus monitoring are compared and contrasted for popular buses such as ARINC 429, AFDX and Time Triggered Protocol. For each bus type the best approach is selected for reliable acquisition, speed of configuration, low latency telemetry and compact recording which is optimized for playback.

KEYWORDS

Bus monitoring, AFDX, ARINC 429, TTP

INTRODUCTION

This paper explores three popular avionics buses and considers the optimal method to acquire data from these buses during the flight test campaign. There are many ways to acquire data from an avionics bus and many choices to be made when designing data acquisition cards. In this paper, such important aspects as hardware platform, packet structure, flexibility, configuration and user experience are discussed. Furthermore the conclusions drawn for monitoring the three selected buses are extrapolated upon to generate a template strategy for monitoring all types of avionics buses.

AVIONICS FULL DUPLEX SWITCHED ETHERNET

AFDX (Avionics Full Duplex Switched Ethernet) [1] is an avionics bus built on top of commercial Ethernet. The extra features that it brings are provisions for deterministic behaviour and redundancy handling. ADFX runs at 10 or 100 Mbps using either copper or fibre. An AFDX network consists of two types of devices, an end system and a switch. The end system provides access to the network for external applications. The switch handles forwarding, filtering and traffic management. Redundancy is achieved by duplication of the switches and the connections to the switches such that each end system is connected to two independent networks as shown in Figure 1.

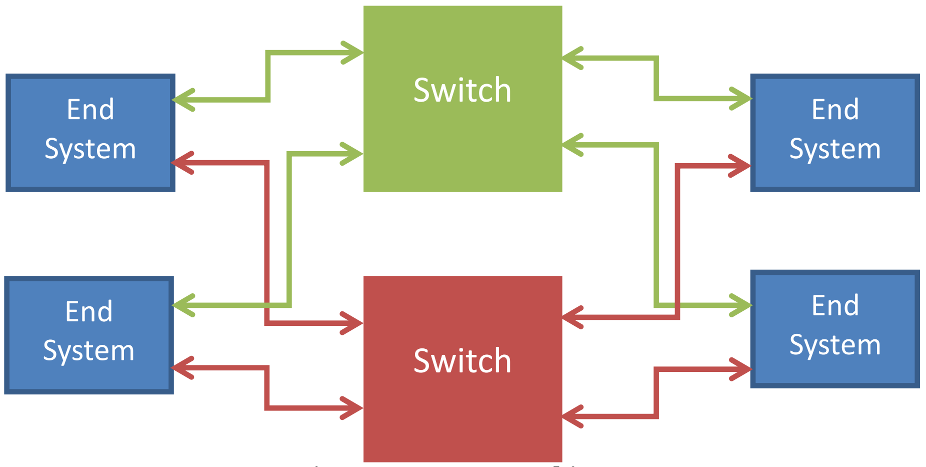

Figure 1: AFDX Architecture

Data is passed through the network via virtual links. Each physical connection between an end system and a switch can have multiple virtual links. The virtual link identifier is stored in the MAC address of the Ethernet frame. Determinism is achieved by limiting the amount of traffic that can be sent on any virtual link. Each virtual link has a bandwidth allocation gap (BAG) which dictates how often a frame can be sent on that virtual link. If the bandwidth allocation gap is 1 millisecond then the maximum rate that an AFDX frame can be sent on that virtual link is once per millisecond. Using this BAG the system designer can set up the AFDX network to guarantee that there is no data loss and that no frame is delayed excessively.

For test the job of the bus monitor is to listen to traffic on the AFDX bus and extract the relevant parameters required for analysis. Typically a subset of the parameters are required for analysis on board or on the ground (via telemetry) during flight and the entire content of the bus is recorded for post flight analysis.

The first task of the AFDX bus monitor is to remove redundancy. To send both copies of each parameter to telemetry, on board analysis or recording would put severe strain on RF bandwidth, network bandwidth and recorder capacity. The second task of the bus monitor is to extract (or parse) the subset of the parameters required for in flight analysis. AFDX is a transport layer protocol. This means that the application layer may be different depending on the platform. The same bus monitor that monitored AFDX on one platform may not be able to monitor AFDX on a different platform. So for bus monitors knowing that the bus is AFDX is only half the story.

An elegant approach to AFDX bus monitoring splits the monitoring into two parts. Firstly there is a card that removes the redundancy. Redundancy removal will ensure that only one copy of the data is retransmitted and that if any frame is lost or corrupted on the primary bus the valid frame from the secondary bus is used instead. The output of the redundancy remover is standard Ethernet and is sent to two different sinks. The first sink is a network recorder which records Ethernet frames. The second sink is a bus monitor to parse the parameters for in flight analysis.

Since AFDX is a transport layer protocol, performing the redundancy removal in one card and the parsing in a second card provides the flexibility to cater for different application layer protocols. One popular protocol defines the parameter based on the location in the frame. This means that if “left wing temperature” is the first parameter in the payload of a frame for a given virtual link it will always be in the first parameter in the payload for all frames on that virtual link. This application layer protocol can be parsed using a standard Ethernet bus monitor. However there are other application layer protocols whereby the parameter is not identifiable by its location in the frame but by a preceding identifier. For example when sending ARINC 429 over AFDX it can be common to add an identifier before each ARINC 429 message to uniquely identify each message and hence each parameter. To parse parameters sent using this protocol requires a different type of bus monitor which searches via packet content rather than by means of location in the packet.

As with all electronic designs, there are many ways to implement a bus monitor. The protocol handling and parsing can be implemented in hardware using an ASIC (Application Specific Integrated Circuit) or FPGA (Field Programmable Gate Array) or using software in a microcontroller. Notwithstanding the extra reliability and robustness that an FPGA architecture provides there are other reasons to choose an FPGA based architecture for AFDX bus monitoring. Dedicated hardware will always outstrip embedded software for performance. Embedded software can offer extra flexibility to the designer but dedicated hardware will always do the job faster. This is particularly relevant for AFDX bus monitoring where for some application layer protocols the bus monitor is required to analyse every byte of every packet in order to establish which parameters are in the packet. In fact micro-processor based architectures have been seen to fail to keep up with the traffic when used to monitor AFDX buses.

Another challenge with AFDX bus monitoring is the sheer numbers of parameters that can be carried in any one AFDX bus. Potentially tens of thousands of parameters may need to be defined to allow the flight test instrumentation (FTI) engineer to select whichever parameters are needed during a flight test campaign. The manual task of defining these parameters could be very time consuming. This is a particular problem when the parameter list changes and needs to be continually redefined. In this context modern setup software will significantly simplify the task. Modern setup software contain bus importers which read in the parameter definition in the format defined by the avionics designer and integrator, and translate the parameter definitions into the FTI metadata language such as XidML [2] with one click of a button. This allows all parameters to become instantaneously available to the FTI engineer to place in his telemetry stream or his on board analysis stream.

TIME TRIGGERED PROTOCOL

TTP (Time Triggered Protocol) [3] is a bus protocol designed for industrial applications. Several nodes share the bus and access to the bus is controlled using TDMA (Time Division Multiple Access). Data transfer consists of TDMA rounds. Each round is divided into slots and each node gets one slot in which it can send multiple messages. The bus has two physical channels. The same data can be sent on both channels to provide redundancy or different data can be sent on each channel to increase the throughput.

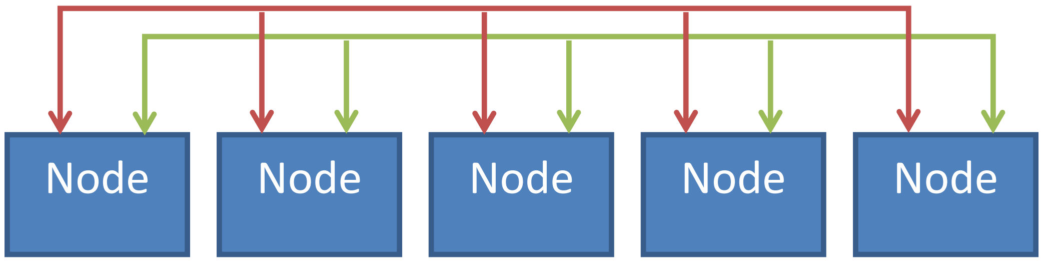

Figure 2: TTP Architecture

As with AFDX, TTP bus monitoring requires two tasks. First redundancy needs to be removed (if it exists) and second parameters need to be parsed and packetized. Given that the redundancy is optional, the bus monitor must be configurable to remove redundancy or provide a copy of data from both buses.

In terms of acquiring parameters the flight test engineer will want to be able to parse a subset of the parameters for in flight analysis and to be able to record all parameters on the bus. Recording all parameters for later analysis is referred to as packetization. Given that the backbone of most flight test systems is now Ethernet based the optimal place to record all the parameters on the bus is a network recorder. The data acquisition chassis will send the data from the chassis through the network to the recorder in Ethernet format. It is therefore the job of the bus monitor to packetize the data in such a way as to make most efficient use of the network and recording resources using a format that allows easy replay via analysis software.

There are several standards that can be used for transmitting and recording acquired data from avionics buses. IRIG 106 chapter 10 [4] defines how avionics buses are recorded but it does not define how bus parameters should be transmitted over Ethernet networks for recording in a distributed acquisition architecture. The iNET [5] working group is focused on providing a fully networked flight test architecture and has proposed the TMNS header for transporting parameters via Ethernet. However iNET is not yet a published standard and is subject to change. On top of this several of the major aircraft manufacturers have developed their own standards for data transmission and recording. At this point iNET TMNS is the closest to an industry wide standard so a transmission protocol based on iNET gives the best options for interoperability in terms of transmission, recording and replay. However maintaining hooks to support the other existing formats is also a practical approach.

There is also the question of how best to package the parameters within the Ethernet frames. The iNET TMNS header is 28 bytes. When added to the Ethernet MAC, IP and UDP headers the total overhead is 70 bytes. In order to use network bandwidth and recording capacity efficiently it is therefore necessary to send frames that are as close the maximum size frames of 1514 bytes as possible. This is done by building a frame with multiple messages from a bus. Typically for each bus the bus monitor will gather messages from one bus and build the payload of an Ethernet message. Once enough messages have been received so that the payload is at, or close to the maximum size, that frame is transmitted and the bus monitor begins generating a new frame payload. This is a particularly efficient method for asynchronous buses where the data does not come in a continuous flow. For these buses a policy of sending a frame after a given period of time could result in small or empty frames being transmitted through the network, wasting valuable bandwidth.

However using this aperiodic strategy to send frames of bus data only when the frame is full could lead to large latencies for some parameters as if the bus was quiet for long spells it could potentially take hundreds of milliseconds to fill a frame. To counter this issue the FTI engineer has two options to tailor the behaviour of his bus monitor card. The first option is the ability to set the frame size. Using this method he can reduce the frame size from say 1500 bytes to 500 bytes to ensure that the latency from a given bus is not excessive. He can also set a timeout. This timeout is a counter which starts ticking after the first message in the frame is received and after the counter has reached the timeout value the frame will be sent regardless of whether it has reached the target frame size. This allows the FTI engineer to set the worst case latency for all messages in the system.

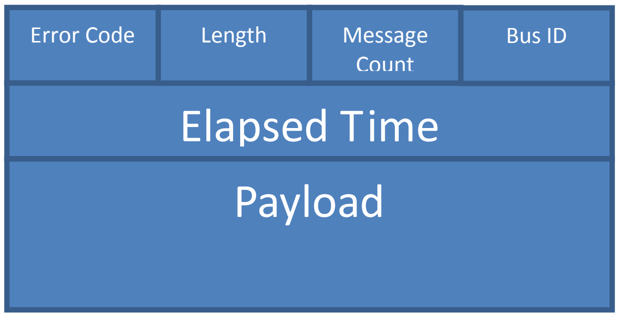

Figure 3: Message Header

When packetizing the messages, extra metadata is required to allow the analysis software to easily replay the parameters. TTP messages have variable length so a message header is added which indicates the length of the message as shown in figure 3. Also included in the message header is an elapsed time stamp. This time stamp allows the analysis software extrapolate when the message was acquired relative to the timestamp in the iNET packet header. The timestamp in the iNET packet header is the IEEE 1588 [6] time stamp of the first message in the frame. This is particularly useful for asynchronous buses where the messages do not arrive at regular intervals.

One advantage of the building the frame payload in the hardware is that the FTI engineer does not need to build any of the Ethernet frames in setup software. He simply needs to select the stream ID he wants to identify the frame and the bus monitor builds the frame under the hood. Similar to AFDX, a large number of parameters may be present on each TTP bus. Again the manual task of defining these parameters could be very time consuming. As with the AFDX bus monitor one of the key functions of modern setup software is to enable quick setup of the system. Bus importers which take parameter definition in native format and import them into the FTI metadata file turns a long laborious job into single click operation.

ARINC 429

ARINC 429 is the most widely used avionics data bus. An ARINC 429 bus consists of one transmitter and up to 20 receivers. Data is transferred at either 100Kbps or 12.5Kbps. The protocol is slowly being replaced by higher speed buses but is still very common with modern aircraft which often have a large number of ARINC 429 buses. Given one of the key requirements of an ARINC 429 bus monitor is high channel count. A modern bus monitor can monitor up to 24 buses simultaneously.

Monitoring a large number of channels is made possible by the relatively slow speed of the ARINC 429 bus. However the slow speed of the bus also poses some problems. As was mentioned above it is important for network efficiency to build frames as close to the maximum frame size as possible. However due to the slow bus speed of ARINC 429, waiting to fill the maximum size frame will cause large latency even if the bus is fully utilised. On the other hand setting a worst case latency of say 50 milliseconds will always result in small frames wasting bandwidth. This problem can be solved by using data from multiple buses to build frames. In this paradigm every message that is received by the bus monitor regardless of which bus it was on will be put into the Ethernet frame payload. The message header as shown in figure 3 contains a bus ID such that the analysis software knows which bus the message came from. Using this strategy an ARINC 429 bus monitor can generate network efficient low latency frames.

While discussing TTP bus monitoring the method by which the hardware packetizes frames for recording was explored. However for all bus monitors including ARINC 429 there is also the question of how to parse a subset of the data for in flight analysis. In fact to some extent packetization is relatively transparent to the FTI engineer whereas he needs to define every parameter that needs to be parsed. This is because the hardware cannot know in advance which parameters are required to be analysed in each flight.

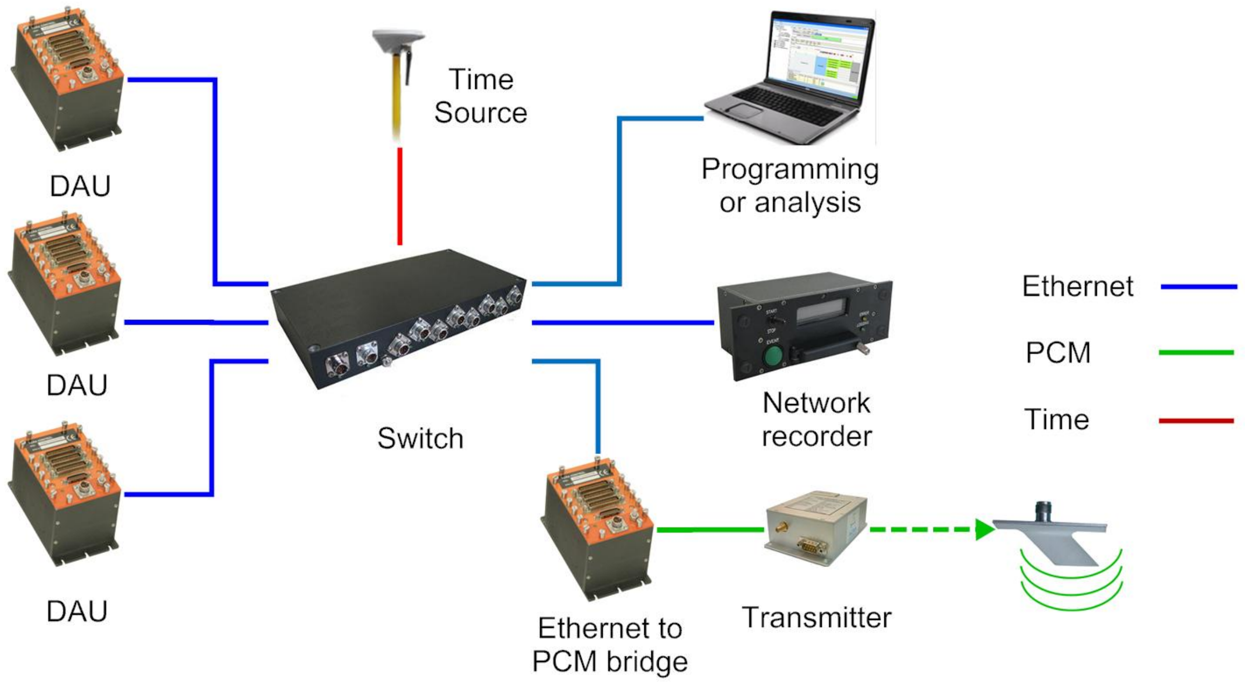

Figure 4 Typical Network System

To explore this topic a typical networked system with an Ethernet to IRIG 106 Chapter 4 PCM bridge is considered. Looking at figure 4, the bus monitors will be located in the chassis on the left of the diagram. The FTI engineer will want to place parameters from every chassis in the outgoing PCM frame. To do this, packets need to be built to route the parameters from every other chassis to the chassis which houses the Ethernet to PCM bridge. Traditionally this was a laborious task where first the outgoing PCM frame is created before packets from each chassis are defined to ensure that each parameter reached the Ethernet to PCM bridge chassis in time for transmission in the PCM frame. This would often have to repeated many times to get the timing correct.

Modern setup software can help with this problem. First, an FTI engineer can quickly define bus parameters using bus builders and importers. Then the PCM frame can be built from the Ethernet to PCM bridge chassis where all parameters in the system (every parameter from every chassis in the system) are available for placement. Once the parameters are placed in a PCM frame, the software will under the hood ensure the safe transport of all the parameters from the other chassis. The software will choose the optimal frame size and will control the traffic through the network such that the PCM frame is guaranteed to be coherent. This simplifies the FTI engineer's task to building a PCM frame as if all parameters came from the same chassis.

GENERIC BUS TYPES

AFDX, TTP and ARINC 429 are three of the common bus types considered in this paper. There are other common bus types such as CAN bus, MIL-STD-1553 and RS 422. On top of this there are a large number of proprietary formats that are currently being used. In essence the job of the bus monitor can be broken down into three separate tasks. Number one is protocol tracking, number two is selecting particular parameters for in flight analysis or parsing and number three is gathering up all parameters on the bus for post flight analysis (also known as packetizing). Once the strategy, architecture and IP blocks are established for parsing and packetizing ,the only real difference between each bus from a hardware point of view is the protocol tracker. For each new bus type the job is simply one of designing a new protocol tracker and integrating the existing parser and packetizer IP blocks.

Common implementation across all bus monitors also leads to common software setup and to a consistent look and feel of the user interface. This is important as once the user has learned how to setup one bus monitor there is no additional learning curve to setup another bus type. This added to other features – such as bus importers and builders, and packetizers which automatically build frames – turns the potentially complex and lengthy task of configuring a system with a large number of avionics buses into a quick and straightforward exercise.

CONCLUSION

While AFDX, TTP and ARINC 429 are quite different bus protocols, similar conclusions can be drawn on how best to monitor these buses for flight test. These conclusions can also be applied to many other bus types. AFDX in particular is a transport layer protocol, so flexibility is important. Sometimes different bus monitors are required depending on the application layer protocol built on top of AFDX.

In general terms FPGA based architectures are a better choice for bus monitoring due to their reliability, robustness and superior performance when compared with microprocessors based architectures. For the purposes of gathering all data on a bus it is important to build frames that are bandwidth efficient and have acceptable latency. This can be achieved using the automatic packetization strategies discussed in this paper. However, the user still has the ability to tailor these strategies by adjusting the size and latency of frames coming from his bus monitor.

For parsing subsets of the data it is important to simplify the bus definition, inter chassis communication and the output frame definition. This is achieved by having setup software which instantaneously imports thousands of parameters and makes all parameters in the system available to the FTI engineer at the output. Such a policy means that the FTI engineer does not need to be concerned with inter chassis transfers or spend a large amount of time defining the parameters for each bus.

These approaches will become even more important as aircraft are equipped with new and exotic buses with more complexity and higher speed and flight test campaigns gather ever more parameters.

REFERENCES

[1] ARINC Specification 664 Part 7, AEEC, September 2004.
[2] Alan Cooke, Diarmuid Corry: "*XML: A Global Standard for the Flight Test Community*", ETTC Proceedings, 2004
[3] Kopetz, Herman; Grunsteidl, Gunter (1993-06-22 - 1993-06-24), "*TTP - A time-triggered protocol for fault-tolerant real-timesystems*", FTCS-23. The Twenty-Third International Symposium on Fault-Tolerant Computing, Digest of Papers, Toulouse, France:
[4] Inter Range Instrumentation Group "*Telemetry Standard RCC Document 106-09*", Chapter 4, April 2009
[5] T Grace, J Kenny, M Moodie, B Abbott"*KEY COMPONENTS OF THE INET TEST ARTICLE STANDARD*", ITC 2009
[6] IEEE Standards Committee, "*Precision clock synchronization protocol for networked measurement and control systems*", IEEE Std. 1588, 2004