Challenges in Monitoring Modern Instrumentation Networks

Michaela Blott

ACRA CONTROL LTD, Landscape House, Landscape Road, Dublin 14 Ireland

Abstract: The adoption of commercial off the shelf networks, such as Ethernet, FireWire and FibreChannel, within the avionics community has dramatically changed the architecture of avionics busses and instrumentation networks. Higher bandwidth links and unified interconnects simplify existing infrastructure and wiring. But due to their point-to-point nature, networking topologies are fundamentally different from systems built on legacy bus technologies such as CAIS and MIL-STD-1553. Switched networks and ring topologies pose various challenges for the implementation of network monitoring hardware, and affect the design of bus monitors and distributed data acquisition systems.

This paper discusses some of these issues. In particular we address deployment issues, architectural choices such as pass-through versus tap approach, as well as handling of bandwidth requirements and complex communication protocols. We illustrate on the basis of a FireWire monitor system how these obstacles have been overcome for one given application.

Keywords: IEEE-1394, Ethernet, COTS, Networking, Data Acquisition Systems

1. Introduction

Monitoring of avionics and instrumentation busses has various purposes: busses are observed for error conditions to trigger alarm systems; busses are recorded for post-flight analysis; but also for extraction of parameters for real-time cockpit displays. Ten years ago, designers of monitoring hardware typically encountered data links that operated at less than 10Mbps, address spaces that could be measured in megabytes, and plain bus topologies where any traffic was seen by all nodes on the entire bus. Deployment was uncomplicated in that a bus monitor could be connected anywhere to the system. Furthermore, due to their passive configuration, risk of interference with mission critical data was negligible.

Nowadays, avionics interconnects and instrumentation networks are fundamentally different. Commercial off the shelf networks (COTS) increasingly penetrate the flight test environment, and although there is dramatic potential offered by these new technologies, engineers are faced with a new range of obstacles. Designers encounter data links at gigabit speeds, and address spaces that have increased by orders of magnitudes. Sophisticated communication models are present. Multi-layered protocols such as suggested in the ISO OSI reference model [12] need to be decoded. Redundancy management might need to be handled as for example within an AFDX/ARINC664 network. Large amounts of data need to be processed, filtered, aggregated, forwarded, and recorded. Finally, due to the point-to-point nature of these high-speed connections, switching elements and bridges are introduced which dramatically change the landscape of network topologies. These new flavours of interconnects constrain the deployment of the monitoring devices and affect the design and architecture of distributed data acquisition systems.

This paper addresses some of these issues. In section 2, we investigate the more prominent challenges that designers are faced with nowadays. Section 3 presents a case study of a FireWire monitoring system that demonstrates how for a given application these challenges were overcome. The paper concludes with section 4.

2. The Challenges

In recent years various different networking technologies have been adopted within the flight test community such as FC-AE and ARINC-664, which are based on FibreChannel and Ethernet respectively [6,7,8]. These technologies differ from each other in many ways, such as application protocols, communication models, and physical link implementations. In that way each of these standards poses individual problems, question marks and challenges. However, in this paper we focus on some of the more prominent issues that are common to most of them. More explicitly, we address the following problems, which are explained individually in the following subsections.

- Point-to-point links
- Deployment
- Increased processing requirements

2.1 A Quick Look at the Basics of Monitoring Point-to-Point Links

High-speed networks are inherently based on point-to-point connections to meet signal integrity requirements of transmission at high data rates. Figure 1 illustrates a simple example of a link segment from node A to B. There are few basic approaches that can be taken to monitor this link. Roughly speaking, we can distinguish between active and passive approaches.

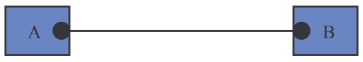

Figure 1: Trivial point-to-point link

The first approach we discuss is an active monitor and based on the implementation of a two-ported bus monitor that bridges all traffic and monitors while the packets fly through. This is shown in figure 2 and will be referred to as "pass-through" or "bridge" approach in the following discussions.

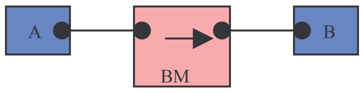

Figure 2: Pass-through bus monitor

All other options are classified as "passive approaches" in that the forwarding of traffic to the bus monitor becomes a responsibility of the network itself. For example, an external tap, such as an optical splitter, could be inserted into the link. The tap would then copy all traffic to the bus monitor as shown in figure 3. Similarly, a bus monitor could be connected to a so-called mirror or monitor or SPAN [13] port on a switch, which mirrors all incoming traffic on this port.

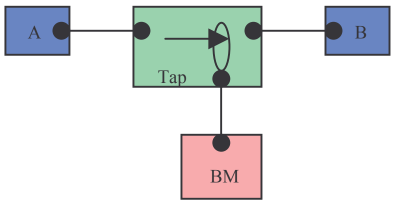

Figure 3: Monitoring via external tap

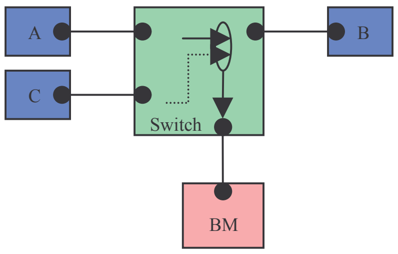

Figure 4: Monitoring via mirror port on a switch

The external tap and mirror port scenario are common approaches taken in the telecommunication industry for monitoring health and performance of Ethernet networks.

A variant of the passive approach relies on existing end-nodes within the system to forward the traffic explicitly to the bus monitor. For example, within an AFDX network, the bus monitor could be included within the target multicast group. However, this inevitably increases the load on the overall network and consumes system resources as, for example, link bandwidth and processing resources on the switches.

Pass-through and end-node designs are fundamentally different. Obviously, pass-through devices impose the implementation of at least two ports, whereas an end-node design requires only one port. Also, pass-through devices require the implementation of packet transmission whereas end-node designs only sink data. Implementation of transmission circuits is significantly more complex for ARINC-664 devices than for a MIL-STD-1553 or CAIS device. It involves the implementation of a full IP-stack including CRC and checksum computations, as well as additional buffer requirements and scheduled transmission as the standard defines Quality of Service [11]. All of this considerably affects the complexity of the design. Further, a pass-through device must comply with much stricter fail-safety requirements because it can potentially disrupt whole areas of the network. Extra precautions, such as relay bypasses and redundant power-supplies, should be considered. Furthermore, the bridging monitors must operate at full line speed in contrast to end-nodes which do not necessarily need to fulfill this requirement. Multiple monitors can monitor one bus concurrently to meet full bandwidth requirements as is shown in section 3. Finally, the choice of architecture has immediate implications on the deployment of the monitoring devices, as will be shown in the next subsection.

2.2 Deployment Issues

As topologies change from busses to switched networks and rings, one fundamental issue arises in conjunction with monitoring devices. This is illustrated in figures 5 and 6 which show an example of a simplified 100BaseTx switched Ethernet network and a MIL-STD-1553 bus. On a bus, deployment is unconstrained in that all data is visible independent of the location of the bus monitor. Within switched networks monitoring of any individual link segment only captures part of the traffic as is illustrated in figure 6. Some of the data captured with device #1, #2 or #3 is transmitted out from the switch onto segment D and some on segment E.

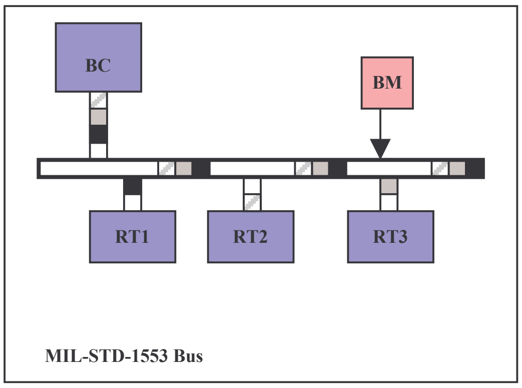

Figure 5: Deployment of monitoring devices on a MIL-STD-1553 bus

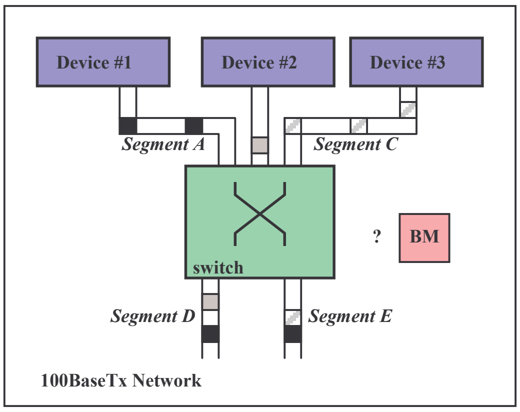

Figure 6: Deployment problem within switched networks

The system integrator is left with a number of choices. When using bridging monitors multiple devices need to be deployed, for example one in segment D and one in segment E. Figure 7 illustrates an approach that takes advantage of a mirror port on a switch. This solution requires only one monitoring device, which could be a single ported end-node design. This approach is attractive in that the monitoring hardware is completely isolated from the avionics or instrumentation bus and cannot interfere with the operation of the overall system. However, it requires additional processing resources on the switch.

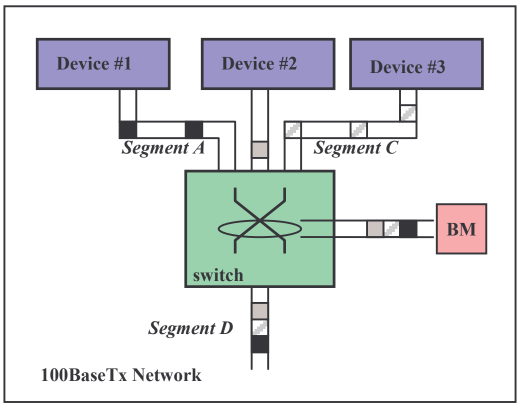

Figure 7: Monitoring via mirror port

FireWire is a special case since it actually behaves as a bus although it is implemented on point-to-point links [1,2,3]. This is achieved by using multi-ported nodes that broadcast all incoming traffic. With that, all traffic is visible on any individual segment within the system and monitors can be connected to any available port within the bus topology as is illustrated in figure 8.

Within ring topologies, designers encounter similar issues. However the discussion of these would lead beyond the scope of this paper.

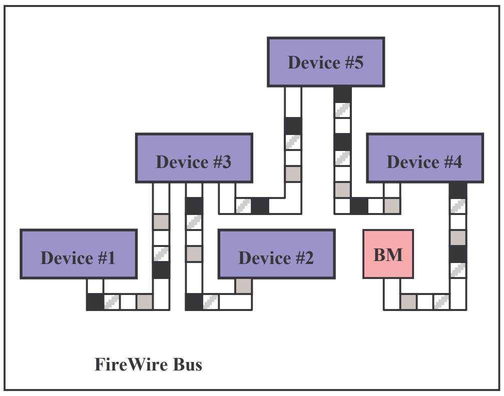

Figure 8: Deployment of a bus monitor within a FireWire bus

2.3 The Processing Challenge

Naturally, monitoring of a higher bandwidth link requires more processing power than the monitoring of its lower speed equivalent. Basic decoding and parsing have to operate at link bandwidth. However, there are additional factors that contribute to extra processing overhead associated with these new technologies. Some of these are listed below, and described in more detail in the following paragraphs. Implementation of this functionality is critical for real-time applications or bridging monitors. When data is recorded, post-processing software could potentially handle error checking, deduplication of data captured on redundant links, and handling of ordinal integrity.

- Introduction of CRCs and checksums
- Redundant backup networks
- Layered communication protocols
- Numerous application protocols
- Lack of ordinal integrity
- Potential packet transmission

<u>CRCs and checksums</u>
To compensate for increased bit error ratios on the physical links, Ethernet, FireWire and FibreChannel introduce additional CRCs and checksums in comparison to the under CAIS and MIL-STD-1553 defined parity bits. These CRCs and checksums need to be generated and checked which requires additional circuits.

<u>Redundant backup networks</u>
As in the case of AFDX, redundant backup networks are defined. For monitoring hardware this implies that both links need to be observed and the hardware needs to facilitate for two passive ports. If captured data is used by real-time applications such as alarm systems, error detection and data deduplication must be implemented.

<u>Layered communication protocols</u>
Layered protocols are commonplace with new networking technologies. A typical example is UDP over IP over Ethernet. For monitoring hardware this has a number of consequences. Even restricted decoding of a single application protocol adds overhead to a standard parser. In

general terms we can identify three functionalities that are affected by the more sophisticated communication protocols: classification of the type of message, flow identification, and flow state maintenance. Classification implies, for example, decoding of Ethernet frame type, IP header fields, such as protocol type and type of service fields, and UDP port numbers. Flow identification (parser slot mapping) and the state maintenance of a flow involve analysis of MAC addresses and/or IP addresses at minimum. In the case of a MIL-STD-1553 monitor, flow identification and classification is defined as a function of a maximum of two 16-bit command words. The state of a flow was immediately determined as there are never interleaved information exchanges and a remote terminal either replies directly or not at all. For UDP this is more complicated. A flow identification would in theory be a function of at minimum two 32-bit IP addresses, assuming a strict and static correlation between MAC and IP addresses. These massive address spaces can require hash lookup algorithms with collision handling. Flow states need to be maintained when acknowledgement protocols are in place and multiple unacknowledged flows can coexist. Typical examples for this would be TCP/IP or asynchronous transactions under FireWire. For these flows state information must be stored within the flow table, and the maintenance of these flows requires additional read-modify-write accesses.

Numerous application protocols

Another aspect to layered protocol stacks is the multitude of application protocols that can coexist. Designers are faced with decoding of different application protocols. Restricting the recognition to only selected protocols can have an impact on the usability of the device within other scenarios. Alternatively, the devices can be designed to monitor on a lower layer such as the link layer. With that any application protocol can be captured but obviously the ability to differentiate between them is lost. Further, this limits the monitor in its functionality in that error checking can only cover the layers that are decoded. Protocol layers that are not decoded are treated as payload.

Lack of ordinal integrity

When capturing traffic on a switched network, a new phenomenon arises: packets are buffered within switching devices and can take different paths through a network. One consequence of this is that packets may arrive out of sequence. Implications for the monitoring hardware is that sequence numbers must be at minimum recorded along with the parameters such that analysis software can rearrange parameters with correct ordinal integrity. Potentially desired behavior on behalf of a bus monitor would be detection of out-of-sequence packets and consequent dumping of such packets.

Potential packet transmission for bridge designs

Finally, for dual-ported pass-through monitors transmit functionality must be implemented. For UDP this now requires the generation of CRCs and checksums, as well as transfer of data to medium access controllers or similar interface devices. Additionally, these new technologies might have Quality of Service requirements as is the case for AFDX. This implies that there is some form of admission control on each link and packets need to be scheduled for transmission.

3. A Case Study of a FireWire Monitoring System

In the previous sections we presented a number of issues that are encountered when implementing monitoring hardware for modern instrumentation networks. In this section we analyse an example of a FireWire bus monitoring system and examine on its basis how these obstacles were overcome. In the first part we depict the design choices made for a single bus monitor, whereas the second part shows how the overall system was affected. The application background required the monitoring of twelve FireWire busses that operated at 200Mbps. The traffic to be monitored was transferred only in global asynchronous stream packets.

3.1 The FireWire Bus Monitor

The FireWire bus monitor itself was intended to be a user-module that can be inserted into one of the slots within a digital data acquisition chassis with a custom backplane (KAM500 [14]).

The first design choice that had to be made was whether the bus monitor should be implemented as a multi-ported device or a single-ported end-node, in FireWire terms referred to as branch or leaf devices. Standard design practice suggests that bus monitors should not be placed within critical parts of the system. Therefore to minimize potential interference with the system, a leaf node implementation was more attractive. Furthermore, an end-node design does not require implementation of transmit functionality, and only part of the transaction and link layer need to be supported which significantly simplifies the design. As discussed in section 2, deployment possibilities vary depending on this design choice. However, constraints are less of an issue in the special case of FireWire, where all traffic can be captured even as a leaf node. Therefore, a leaf node design offered considerable advantages over the bridge design in many respects. The design is simpler since packet transmission, full line-speed forwarding, is not a requirement. It is safer in that a failure by the monitor is not disrupting communication channels between other nodes. And finally it is very little constrained in terms of deployment options.

The second choice involved component selection. Typical FireWire devices are built on standard chipsets as are offered for example by Texas Instruments [15]. The Texas Instruments portfolio consists of PHY devices and a so-called link layer controllers. The PHY device handles all functionality of the physical layer whereas the link layer controller provides services associated with request and responses of individual transactions. Deployment of these chipsets offer much functionality out of the box with little development and integration effort. By the same token, they make hardware inflexible. The designer is locked into the behaviour of the chipset. Because a leaf node design does not require the implementation of a full link layer with packet transmission, the initial advantage of using a link layer controller was fairly slim. Similarly, the set-up and control of the link layer controller poses approximately the same amount of development effort as interfacing to the PHY directly. Therefore, we decided against the use of a link layer controller and implemented the interface to the PHY device (a TSB43BA1) within a high-density field programmable gate array (FPGA). This way we achieved utmost flexibility and greater control over the bus. The FPGA sits within the heart of the design and contains 99% of the functionality.

The third challenge was bandwidth and processing requirements. The selected PHY device offers link speeds of up to 400Mbps whereas the backplane operates at lower bandwidth. Obviously, there was no need to implement more than one physical interface per module and yet we still had to deal with potential overflows. The only way to overcome this was to approach this problem on a module and system level. The impacts on the architecture of the overall distributed data acquisition are discussed in the next section. For the design of the module, this implied that we needed to drop packets in a controlled fashion, which was achieved in the following manner: The first ingredient is a filter that can be programmed into the monitor. This filter informs the hardware about which channels to parse and which ones to drop. Obviously, this way we can program one module to filter channels such that it cannot generate aggregate traffic in excess of the backplane's capability. Other modules and backplanes can then be used to capture the remaining amount of traffic. The correlation of traffic and tags that were captured within separate units can be achieved via time-stamping.

Figure 10 illustrates the bottlenecks on the module. At the top of the diagram, there is a potential flow of 400Mbps worth of FireWire traffic coming into the module. At the bottom, traffic and associated tags are read out of the module at slower rates, which forms one of the bottlenecks. A second bottleneck is located on the SRAM interface which can be operated at an average of 256Mbps concurrent reads and writes. The module provides full 400Mbps speeds until packets have been passed through the filter which is very important to ensure controlled loss of traffic only. This is the first stage where dropping of traffic takes place. The transactions that were correctly received, classified and parsed, will then be written into the current value table together with associated tags at maximum SRAM access speed. The backplane reads these transactions out of SRAM, however at a slower rate than they're written into. Consequently, this provides a second level of filtering. The reason for faster write access than read access is that typically not all information captured is of interest.

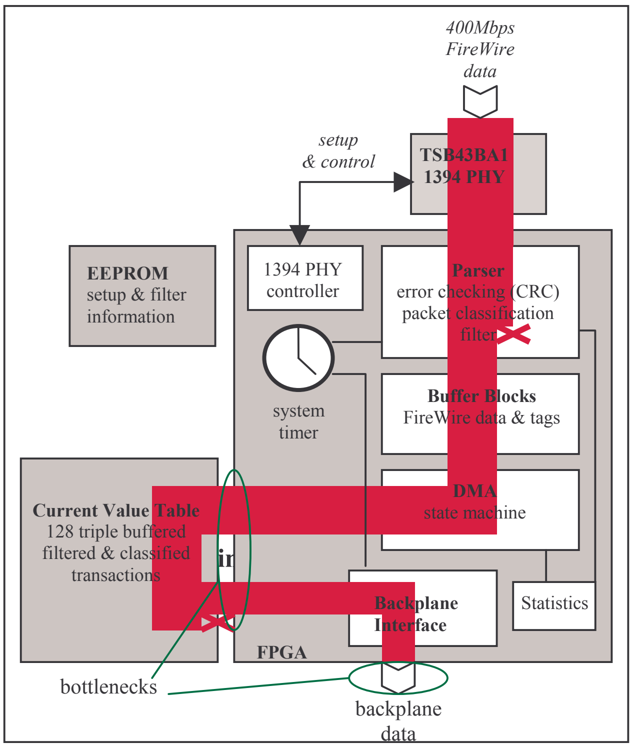
400Mbps FireWire data
setup & control
TSB43BA1 1394 PHY
EEPROM setup & filter information
1394 PHY controller
Parser error checking (CRC) packet classification filter
system timer
Buffer Blocks FireWire data & tags
Current Value Table 128 triple buffered filtered & classified transactions
DMA state machine
Backplane Interface
Statistics
FPGA
bottlenecks
backplane data

Figure 10: Bandwidth bottlenecks

3.2 The Distributed Data Acquisition System Design

As mentioned in the previous subsection, bandwidth and processing requirements can exceed the capabilities of individual digital data acquisition units and must be addressed on a system level. The key idea is that multiple units can monitor the same bus, where each unit only captures a user-controlled part of the relevant traffic. An important implication of this is that all units must be synchronized to the same time sources, such that later correlation of data is possible. The second key is that not all data must flow through the entire data acquisition system and congest it that way. Rather, the data can be transmitted out of each unit and concentrated externally as for example via a switch.

Figure 11 depicts an example application that monitors twelve independent FireWire busses via twelve bus monitors. These monitors are deployed in groups of four per slave data acquisition unit. One of these slave chassis is displayed in greater detail in the left box. It illustrates four busses with branch nodes. Each of these branch nodes is connected to one of the bus monitors. These are located as user-modules within a chassis. The chassis also contains an Ethernet transmitter module that can forward any selection of parameters that are available within the chassis. Additionally, parameters can also be transmitted and received via PCM to and from any other chassis within the topology. Finally, the diagram indicates the reception of time and synchronization signalling from the master chassis.

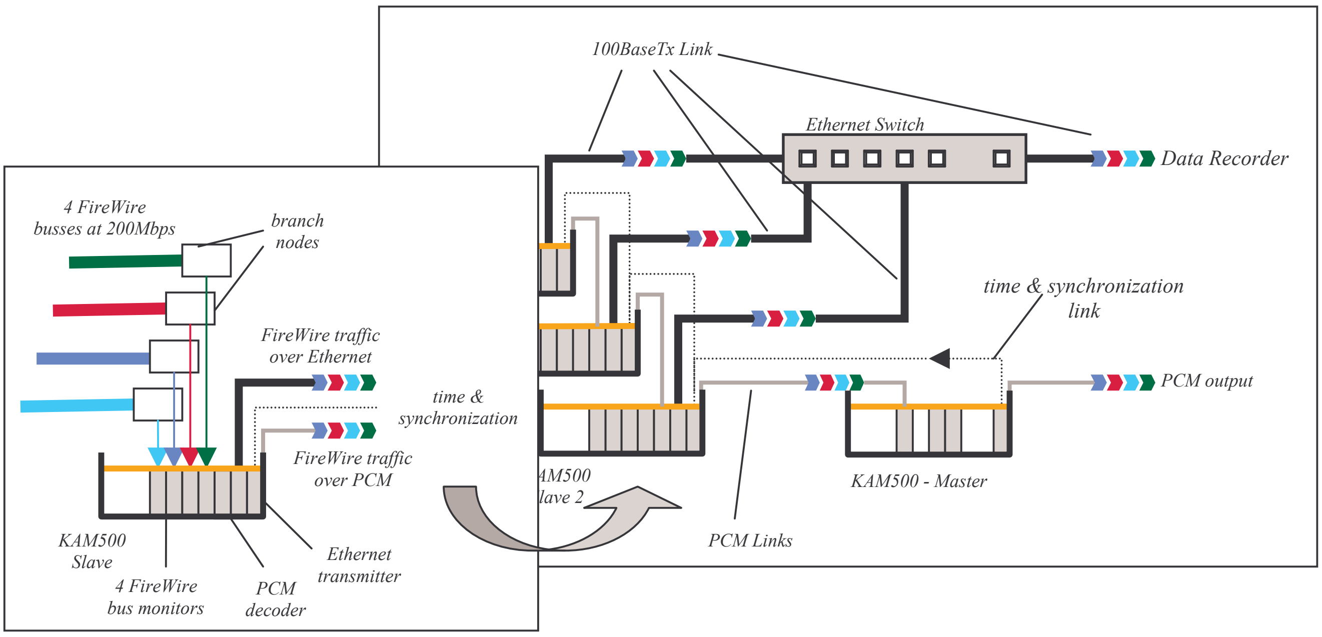

Figure 11: A distributed FireWire monitoring system

The box on the right depicts the system architecture, which comprises three slave chassis and one master chassis. The latter is connected to an external time source and synchronizes the entire distributed system. The main part of captured FireWire data is intended for recording. These parameters are transmitted from each slave DAU directly via Ethernet to an external switch and from there to a standard solid state recorder. This way congestion and bottlenecks within the chassis can be avoided. Some data that is intended for PCM transmission is routed via standard telemetry links from one DAU to the next and finally to the master chassis. From there the data is then forwarded to the outside world.

This system demonstrates how distributed systems can be scaled to meet increased capturing requirements and fulfill different demands such as given by real-time and recording applications at the same time.

4. Conclusions

This paper discussed some of the more general challenges encountered when monitoring modern instrumentation networks. Typical issues that arise within these new networks are related to topology, as these interconnects are typically switched and not all traffic might be visible on any individual segment of the network. Tightly linked to the deployment issue is the choice of architecture for monitoring devices, whether pass-through or end-node design is suitable within the given circumstances. Conservative design practice would always prefer a passive end-node design that can be situated outside critical system paths. Processing requirements can also be a challenge as not only link speeds increase significantly, but also protocols become much more complex. We showed on the basis of a distributed Firewire monitoring system how to overcome these issues. In particular, we demonstrated how bandwidth bottlenecks can be overcome on a system level and how distributed systems can scale to meet ever increasing requirements.

5. References

[1] *IEEE Standard for a High Performance Serial Bus,* 1996,
IEEE Std 1394-1995, Institute of Electrical Engineers, New York

[2] *IEEE Standard for a High Performance Serial Bus 2nd Amendment,* 2002,
IEEE Std 1394b, Institute of Electrical Engineers, New York

[3] *FireWire System Architecture*
MindShare Inc, Don Anderson, Addison-Wesley, 2nd Edition

[4] *MIL-STD 1553, Designer's Guide Handbook*
ILC Data Device Corporation, 5th Edition

[5] *High-Speed Networks, TCP/IP and ATM Design Principles*
Prentice Hall, William Stallings

[6] *IEEE Std 802.3 - 2000*
Institute of Electrical Engineers, New York

[7] www.fibrechannel.org

[8] *Technical Report for Information Technology – Fibre Channel-Avionics Environment (FC-AE)*
American National Standards Committee, INCITS, Information Technology,
TR-31-2002

[9] *IRIG-STANDARD 106-93 TELEMETRY STANDARDS*
Secretariat Range Commanders Council,
U.S. Army White Sands Missile Range, New Mexico 88002-5110

[10] INET

[11] *Draft 2 of ARINC Project Paper 664 Part 7 Deterministic Networks*
ARINC; Airlines Electronic Engineering Committee, 2551 Riva Road, Annapolis, Maryland 21401-7465 USA

[12] *Computer Networks*
Prentice Hall, Andrew S. Tanenbaum, 2nd Edition

[13] www.cisco.com/warp/public/473/41.html
Configuring the Catalyst Switched Port Analyzer Feature

[14] www.acracontrol.com

[15] www.ti.com/sc/1394

6. Glossary

AFDX:	Avionics Full Duplex (ARINC 664)
CAIS:	Common Airborne Instrumentation System
COTS:	Commercial off the shelf
CRC:	Cyclic Redundancy check
DAU:	Data Acquisition Unit
FPGA:	High density field programmable gate array
Mbps:	Mega bits per second
OSI:	Open Systems Interconnection
SPAN:	Switched Port Analyzer