DOT/FAA/AR-09/24

Air Traffic Organization
NextGen & Operations Planning
Office of Research and
Technology Development
Washington, DC 20591

Data Network Evaluation Criteria Handbook

June 2009

Final Report

U.S. Department of Transportation
Federal Aviation Administration

NOTICE

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.act.faa.gov in Adobe Acrobat portable document format (PDF).

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-09/24		

4. Title and Subtitle	5. Report Date
DATA NETWORK EVALUATION CRITERIA HANDBOOK	June 2009
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Kevin Driscoll, Brendan Hall, Phil Koopman, Justin Ray, and Mike DeWalt	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Honeywell International, Inc. 3660 Technology Drive Minneapolis, MN 55418	11. Contract or Grant No. DTFACT-05-C-00002

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
U.S. Department of Transportation Federal Aviation Administration Air Traffic Organization NextGen & Operations Planning Office of Research and Technology Development Washington, DC 20591	14. Sponsoring Agency Code AIR-120

15. Supplementary Notes

The Federal Aviation Administration Airport and Aircraft Safety R&D Division COTR was Charles Kilgore.

16. Abstract

The purpose of this Handbook is to provide evaluation criteria to be used in the development, selection, modification, adaptation, or approval of data network technologies and components to be deployed in safety-critical aviation systems. The expected readership for this Handbook primarily includes designers of digital electronics systems that may use data networks and those who are concerned with the certification of aircraft or aircraft engines containing such systems.

This Handbook's objective for providing these evaluation criteria is to facilitate the process by which data networks are employed in aviation digital electronics systems that may ultimately be certified as part of an overall aircraft or aircraft engine certification process. It focuses on identifying aspects of the technologies and component implementations that ultimately may have an adverse impact on the approval within the certification of an aircraft. Particular attention is given to issues that are generally overlooked or underappreciated in the industry.

This Handbook does not constitute Federal Aviation Administration certification policy or guidance, but may be used as input to future policy and guidance.

17. Key Words	18. Distribution Statement
Databus, Network, Certification, Criteria, COTS	This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	103	

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AC	Advisory Circular
ARINC	Aeronautical Radio, Incorporated
BER	Bit error rate
BGP	Byzantine generals' problem
CAN	Controller area network
CAST	Certification Authorities Software Team
CCA	Common cause analysis
CFR	Code of Federal Regulations
COTS	Commercial off-the-shelf
CRC	Cyclic redundancy code
CSMA/CD	Carrier Sense Multiple Access/Collision Detect
dc	direct current
DMA	Direct memory access
EDAC	Error detection and correction
FAA	Federal Aviation Administration
FCS	Frame check sequence
FIFO	First-in, first-out
FMEA	Failure modes and effects analysis
FMECA	Failure modes, effects, and criticality analysis
FTA	Fault tree analysis
HD	Hamming distance
HIRF	High-intensity radio frequency
IC	Integrated circuit
IEEE	Institute of Electrical and Electronic Engineers
ISI	Intersymbol interference
ISO	International Standards Organization
LLC	Logical link control
MAC	Media access control
MIL-STD	Military Standard
NACK	Negative acknowledgements
OSI	Open Systems Interconnect
PLL	Phase-locked loop
PSSA	Preliminary system safety analysis
RAM	Random access memory
RF	Radio frequency
SAE	Society of Automotive Engineering
SERDES	Serializer/deserializer
SEU	Single-event upset
SNR	Signal-to-noise ratio
SOS	Slightly-out-of-specification
TCP/IP	Transmission Control Protocol/Internet Protocol
TDMA	Time division multiple access
TMR	Triple modular redundancy
TTP/C	Time Trigger Protocol/SAE Class C

EXECUTIVE SUMMARY

Databus and data network technology continues to play an ever-increasing role in aviation digital electronics architectures throughout the range of aviation markets. The evolution of integrated modular aviation digital electronics architectures comprising multiple subsystems integrated into single and redundant data networks is increasing the influence of data networking. The criticality of data networks has previously led avionics manufacturers and aircraft original equipment manufacturers to design specific aerospace solutions to meet their requirements. In recent years, cost challenges have led to the adoption of commercial off-the-shelf (COTS) communication solutions in avionics. Although attractive from a cost perspective, the adoption of COTS presents certification issues, particularly as the complexity and increased leverage of technology continues to evolve. Subtleties may escape the system designer and leave dependability holes. An example is the interference of the Controller Area Network bit-error stuffing mechanism with message cyclic redundancy code coverage. COTS can be adopted as-is, or with fixes added so it is a better fit for dependable avionics requirements, i.e., the adaptation of Ethernet to Aeronautical Radio, Incorporated (ARINC®) Part 7. Helping this trend is the arrival of "safety-critical COTS" in the marketplace, particularly in automobile and process-control areas. However, even with designed-for-purpose technology, it is necessary to ensure that the technology has dependability consistent with real-world requirements and redundancy management schemes.

Development and evaluation of aviation digital electronics data networks that are suitable for safety-critical aviation digital electronics is a complex subject area. It requires detailed knowledge of communications systems, aviation communication and application requirements, mechanisms for creating dependable architectures, certification expectations, and assurance strategies. It is also important to note that, with correct architectural mitigation, almost any data network may be used in a certified system. For example, a layer of fault tolerance can be placed above the network to fix any of its shortcomings.

The objective of this Handbook is to provide criteria for evaluating data network technology for use in safety-critical applications. However, this should not be taken to mean that these criteria can be used to rank data networks in a scale of absolute goodness, independent of the avionics systems in which they are employed. Because the operation of a data network is so entangled with the avionics system it supports, it is not possible to make an evaluation of a data network on its own. The goal is to create a sufficient breadth of criteria that can be used to evaluate the widest range of data networks with respect to the avionics systems in which they may be employed.

This Handbook builds on previous documents in this area, particularly the Certification Authorities Software Team (CAST-16) position paper, "Databus Evaluation Criteria," "Handbook for Ethernet-Based Aviation Databuses: Certification and Design Considerations," and Advisory Circular 20-156, "Aviation Databus Assurance."

This Handbook includes a structured list of issues and criteria related to evaluating data network technologies for digital electronics applications. Many of these issues and criteria are overlooked or are underappreciated by many of today's digital electronics designers.

The material contained in this Handbook is part of the work done for the Databus Evaluation Criteria research project. This project was carried out in collaboration with Honeywell Laboratories, Minneapolis, MN; Carnegie Mellon University, Pittsburgh, PA; and Certification Services Inc., Eastsound, WA. The funding was provided by the Federal Aviation Administration.

1. INTRODUCTION.

The goal of this Handbook was to document objective evaluation criteria for data networks to be used in aviation products. Of particular interest are digital electronics applications that are safety critical. The evaluation of databus and networking technology is not a simple matter. It requires a detailed review and analysis of the lowest-level implementation characteristics of the selected technology together with the ability to map significant behaviors and failures to their architectural relevance. Avionics data networks are becoming more complex. They are evolving from half-duplex links Aeronautical Radio, Incorporated (ARINC®) 429 to full-duplex buses with multiple transmitters. Single master buses (ARINC 429, Military Standard (MIL-STD) 1553) are becoming networks with multiple masters or peer-based networks where effectively all nodes are masters (ARINC 664). These data networks are increasing their complexity by offering more features than in the past, for example, multiple classes of service. In the drive to reduce cost and weight, more integrated networks are shouldering a larger responsibility for correct system operation and avionics system fault containment.

For the purposes of this Handbook, the term "evaluation criteria" means the standards on which a judgment can be made regarding the suitability of a data network for use in digital electronics systems, given the characteristics or features of the data network that may have an impact on system safety. One cannot definitively say that a particular characteristic or feature would have a safety impact, because the architecture in which the network is used may be insensitive to (e.g., may not need) the particular characteristic or feature that would be a problem for other architectures. Thus, this Handbook will describe all the evaluation criteria that need to be considered, regardless of any particular architecture. The system designer and evaluators then must determine whether a particular evaluation criterion is applicable to the data network being evaluated and the system being designed.

This Handbook is not intended to provide a "go/no-go" checklist for justifying any particular data network technology since such a decision is very dependent on how a particular technology is used within an application. Therefore, if the evaluation of a network's suitability for a particular avionics system is unsure or unsatisfactory, the system designer has three options (only the first of which could be considered a go/no-go type of decision):

- Select a different data network
- Alter the data network design or implementation to overcome shortcomings
- Change the system design to accommodate the shortcoming(s)

Instead of a simple checklist, this Handbook provides a set of criteria with ancillary questions that form a framework for a data network technology, e.g., examination of conscience, that can be used to bridge the gaps between a data network technology's behaviors and the system-safety assumptions that underpin the top-level safety case. This is in contrast to a simplistic go/no-go judgment of the data network technology evaluated outside of any context of a digital electronics architecture in which it may be used.

1.1 ORGANIZATION.

Section 1 gives an introduction that provides a rationale for creating this Handbook.

Section 2 describes some aspects of the environment surrounding the evaluation of a data network.

Sections 3 through 9 of this Handbook present a discussion of data network technology attributes that must be considered when evaluating the technology within aviation digital electronics systems. This information is organized in relation to the hierarchies of communication stack models (e.g., the International Standards Organization (ISO) Open Systems Interconnect (OSI) model) for evaluation criteria that fit well with these models (sections 3 through 6) and by themes of special interest that require attention in the system design and deployment (sections 7 through 9).

Organizing the criteria along a communication stack model should make them easier to find and to correlate against communication network description documents, which are often organized in this way. However, a pure communication stack model approach misses essential attributes of data network design. Therefore, it is these areas of special interest (sections 7 through 9) that are most likely to be missed.

Sections 3 through 9 include criteria paragraphs that are numbered sequentially according to their relation to the protocol stack hierarchy. These paragraphs are inserted at the end of sections where the introduction of the criteria is appropriate. Each criterion is formatted in bold font beginning with a criterion number followed by a short title, and the main criterion question or statement. An optional paragraph of ancillary questions may immediately follow a criteria paragraph. These questions are intended to help the reader evaluate the criterion by calling attention to various aspects of the network design.

Note that criterion 2 is an exception to the standard for numbering criteria paragraphs sequentially. While the criterion 2 is numbered appropriately for its relationship to the protocol stack hierarchy, for the purpose of this document, it is placed where it will be best understood (section 4.2) after line-level encoding, has been explained.

Section 10 suggests an evaluation process using this Handbook.

Section 11 provides a summary.

Section 12 lists the references.

Section 13 provides a list of the technical terms used throughout this Handbook.

Appendix A provides additional information on data network technology and its issues.

1.2 BACKGROUND.

From a list of data network technology behaviors and beginning with the Certification Authorities Software Team (CAST-16) position paper, “Databus Evaluation Criteria” [1]; “Handbook for Ethernet-Based Aviation Databuses: Certification and Design Considerations” [2]; and Advisory Circular (AC) 20-156, “Aviation Databus Assurance” [3] as departure points, an examination was made of how communications primitives and services can be leveraged at

the application level, and what impacts the behaviors may introduce with respect to certification. The "Data Network Evaluation Criteria Report" [4] also serves as a source for this Handbook.

1.2.1 Data Network Evaluation Relative to a System Safety Process.

The sheer variety of network and databus technology makes it difficult to characterize generic attributes that can be used for a set of all-encompassing evaluation criteria. The details of the implementation of these networks determine their characteristics; they may be serial, parallel, synchronous, asynchronous, external, internal, intersystem or intrasystem wired, or wireless, etc. In addition, the potential failure behavior of the databus or network technology may be mitigated at the system architecture level, for example, by employing multiple independent data paths, design dissimilarity, or enhanced end-to-end integrity mechanisms above the core network behavior. For these reasons, a bottom-up go/no-go checklist is very difficult to elicit at the network level. Instead, a holistic view of the entire system is required to ensure that the use of the network technology is sufficient to meet the system-level functional responsibility and safety assumptions. Therefore, databus and network technology have traditionally been evaluated on a case-by-case basis against federal aviation regulation Title 14 Code of Federal Regulations (14 CFR) (Aeronautics and Space, Airworthiness Standards) XX.1309 (the safety-related regulations) and 14 CFR XX.1301 (the intended function-related regulations) with a detailed review of the implementation mechanisms. Pertinent regulations related to this research, and adopted and enforced by the Federal Aviation Administration (FAA) are contained in 14 CFR Chapter I Parts 1-199, FAA, Department of Transportation) Part XX (identified below), Subpart F (Equipment), Section XX.1301 (Function and Installation) and Section XX.1309 (Equipment, systems, and installations), and are identified as follows:

- Part 23—Small Airplanes (Normal, Utility, Acrobatic, and Commuter Category Airplanes)
- Part 25—Transport Category Airplanes
- Part 27—Small Helicopters (Normal Category Rotorcraft)
- Part 29—Large Helicopters (Transport Category Rotorcraft)
- Part 33—Aircraft Engines

In addition, 14 CFR 33.28 (Aircraft Engines, Electrical and Electronic Engine Control Systems) also applies. This process is initially top-down, focusing on functions at the aircraft level that are enumerated in a function list.

The hazards associated with the functional failure conditions are determined for each function at the aircraft level. Note that at the initial stages of the process, designers and evaluators may not know how these functions will be allocated to subsystems. While it can be the common cause for failures in multiple functions, the bus or network has not traditionally been viewed as an airplane-level function, rather, it is a tier design choice for how the functions are provided, so at this point, there is no impact. One or more candidate system architectures for aircraft-level

functions are proposed. The system could be a single processing module (analog or digital) with a number of inputs or outputs fed directly to the box, or a single box for each function (analog or digital), or any of a number of alternative architectures. This architecture then forms the basis for an aircraft-level fault tree that demonstrates how failure conditions will flow through the architecture.

At this stage, it is not uncommon to start looking at common cause analysis (CCA). CCA consists of three components: (1) particular risk analysis (e.g., lightning), (2) common-mode analysis (e.g., all boxes receive cooling from a single source or data from a shared network), and (3) zonal analysis (e.g., a fire in the wheel well damages wires that pass through the area but are not related to any equipment in the wheel well), at the architecture level (for example, consider the implications of the two mentioned architectures). As the architecture is refined, an airplane-level network may be derived; this will need to be considered as part of the system fault tree analysis (FTA). This process continues iteratively until a detailed component (i.e., line replaceable unit) level design emerges. This iterative top-down process is captured by a preliminary system safety analysis (PSSA), system and subsystem fault trees, and revisitation of the common cause and zonal analysis as appropriate. The lower levels of the fault tree will contain a number of different faults that can be traced to aircraft-level failure conditions. As the architecture is continuously refined, the use of databuses and network technology can appear at any level and feed into the continuously evolving PSSA. When a preliminary complete design emerges, then a bottom-up approach, called a failure modes and effects analysis (FMEA) or a failure modes, effects, and criticality analysis (FMECA), is instituted on the actual design looking at specific failures of components or group of components and their contribution to the aircraft hazards. A failure condition would be phrased as “loss of all braking” due to a hardware failure (unspecified), and analysis would be conducted to determine all possible failures that could cause the failure condition. The FMEA would start with something like the failure of a power supply and trace it to a system effect. Ideally, the top level of an FMEA or FMECA can be identified with the faults from one or more fault trees. Databuses and network technology services may, therefore, appear in any level of the system design and are required to be analyzed from both the bottom-up (FMEA/FMECA) and top-down (FTA/PSSA, as well as the CCA). When the iterative process is finished, the safety results are documented in the system-safety analysis, including the summaries of the FMEA/FMECAs and the CCA.

For this process to work effectively, it is paramount that the impact of the behavior and potential failure of the databus and network technology is adequately captured and represented in the FTA. For low-complexity network and databus technology, the process above is relatively straightforward. In such cases, the network services assumed by the upper levels of the system behavior are simple and restricted to point-to-point communication primitives only (for example, those concerned with the loss, delay, or corruption of information restricted to a few nodes). However, as silicon integration increases (enabled by continually decreasing process geometries), the failure modes of integrated devices are getting considerably more difficult to bound. Hence, even in the case of simple communication services, great care is required to ensure that the failure mechanisms and assumptions are suitably captured. In addition, as networking technology has advanced, a number of additional services have been implemented at the network level (for example, acknowledgement, message agreement, global time synchronization, system mode change distribution, fault diagnosis, power distribution, etc.). The

system-level impact of such services may be significant, and in many cases, the databus or network may form the intelligence backbone of the system or entire aircraft. In these cases, a more detailed analysis of network behavior and system logic and assumptions is required. For example, if message agreement or interactive consistency is leveraged by applications operating above the network infrastructure to implement active replication strategies (for example, replica determinism for triple-modular replication), the justification of the application-level behavior needs to address implications of network failures or transient upsets that may affect the coverage of such strategies in the event of a fault or external system upset.

1.2.2 The CAST-16 Position Paper.

The CAST-16 position paper, "Databus Evaluation Criteria," [1] was published in February 2003 with the stated purpose of documenting "criteria that should be considered by databus manufacturers, aircraft applicants, and certification authorities when developing, selecting, integrating, or approving a databus technology in the context of an aircraft project".[1] A CAST position paper expresses regulatory concern about technical and safety issues. These concerns have been captured in the August 2006 publication of AC 20-156, Aviation Databus Assurance, which is described in section 1.2.4.

1.2.3 Ethernet Handbook.

In September 2004, the "Handbook for Ethernet-Based Aviation Databuses: Certification and Design Considerations" [2] was published. Its purpose was "to provide the network designer and developer with some guidelines to develop an Ethernet-based databus framework deployable in certifiable avionics systems."

This Ethernet Handbook builds on the CAST-16 position paper and adds guidelines specific for Ethernet-based data networks. These guidelines were not developed solely for the Institute of Electrical and Electronic Engineers (IEEE) 802.3 standards, but also for aviation digital electronics-specific Ethernet derivatives.

The IEEE 802.3 standards constitute a wide variety of data networks. Speeds range from the 1990 version that operated at a maximum of ten megabits per second to ten gigabits per second at the time of this Handbook's publication. It can be expected that even higher-speed versions will be created in the future. The physical line symbol coding includes Manchester, 4b/5b, 8b/10b, and several lesser used encoding schemes. The topologies include buses and stars. There are a number of Ethernet variants, with the simplest using a total of two wires, and the most complex using eight wires per node. Fiber-optic versions of Ethernet use two fibers for each node. The media access control (MAC) mechanisms include Carrier Sense Multiple Access/Collision Detect (CSMA/CD), which is primarily used for buses and in switch-based mechanisms used for stars.

[1] It is important to note that all CAST papers include the following disclaimer: "This position paper has been coordinated among the software specialists of certification authorities from the United States, Europe, and Canada. However, it does not constitute official policy or guidance from any of the authorities. This document is provided for educational and informational purposes only and should be discussed with the appropriate certification authority when considering for actual projects."

Adding to this wide variation of IEEE standard, Ethernets are Ethernet derivatives designed specifically for aviation digital electronics. These include ARINC 646 Ethernet Local Area Network, ARINC 664 Aircraft Data Network, and the Avionics Standard Communications Bus, Version D. These derivatives range from simple adaptations to the aviation digital electronics rugged environment to whole-scale usurping of the MAC protocols with protocols that provide varying degrees of increased media access determinism. (See section 2.3.1 for a detailed discussion of determinism.)

While the Ethernet Handbook covers a wide variety of Ethernet derivatives with guidelines that are more detailed than the CAST-16 position paper, it covers only a small fraction of the possible data networks that can be used for aviation digital electronics.

1.2.4 Advisory Circular 20-156 Aviation Databus Assurance.

AC 20-156 was published in August 2006, which follows very closely to the CAST-16 position paper. Thus, it does not include specific and detailed criteria. AC 20-156 describes a means to gain FAA approval of an aviation data network; wherein the means show that the data network design performs its intended function and satisfies the applicable airworthiness requirements when installed on an aircraft or aircraft engine. This AC is not mandatory and does not constitute a regulation. It describes an acceptable means, but is not the only means, by which a data network can be successfully included in a certified aircraft or aircraft engine.

AC 20-156 calls out eight criteria categories based largely on those created by the CAST-16 paper. Within each category, specific criteria were enumerated. The eight categories and number of criteria in each are:

- Safety—7 criteria
- Data Integrity—10 criteria
- Databus Performance—12 criteria
- Software and Hardware Assurance—3 criteria
- Electromagnetic Compatibility—4 criteria
- Verification and Validation—10 criteria
- Configuration Management—7 criteria
- Security Assurance—2 criteria

1.3 PURPOSE.

This Handbook is intended to facilitate the overall certification process for aircraft or aircraft engines that employ digital electronics systems containing data networks. It builds on the previous work described above, by providing specific and detailed criteria for evaluating a wide range of data network technologies and components with respect to the possible adverse impacts on certification due to their use.

The characteristics of data networks are so varied that it is impossible to create a single set of detailed and specific criteria in which all the criteria are applicable to all data network

technologies and components in all possible applications. Because of this extremely wide variation, creating a concise set of specific and detailed criteria for data networks is much more difficult than creating a similar set of criteria for microprocessors. The combination of extremely wide variation and detail leads to a set of criteria that can be overwhelming.

However, for safety-critical systems, it is usually true that the accuracy of the details is essential. Therefore, this Handbook tries to include as much breadth and depth of criteria as possible. To partially mitigate the problem of having an overwhelming set of criteria, this Handbook presents the criteria on two levels. The higher level is presented in the body of the Handbook with much more detailed discussions included in appendix A. Someone still needs to determine what criteria are applicable to what data network, a task which is too varied to be prescribed in the confines of this Handbook.

Certifiability of a data network means that, if the data network is deployed in aviation digital electronics and complies with all applicable regulations and guidance, one cannot introduce any unacceptable risk to the aircraft as determined by the system safety analysis. Note that this is different from the notion of the product adding quality to the system. An aviation digital electronics component, such as the data network, may add quality. This Handbook does not deal with added quality; rather, it focuses on identifying and preventing aspects of the product that detract from the factors impacting certifiability. Particular attention is given to issues that are generally overlooked or underappreciated in the industry.

This Handbook presents and describes criteria that should be considered by data network manufacturers, aircraft applicants, and certification authorities when developing, selecting, integrating, or approving a data network technology or components in the context of an aircraft project.

1.4 SCOPE.

1.4.1 System Network Role.

The evaluation criteria described in this Handbook were selected to help in the creation or selection of safety-critical aviation digital electronics data networks. The data networks that are safety critical tend to be system data networks (i.e., data networks that connect together a number of subsystems) or data networks that connect together the redundant elements of a safety-critical subsystem. These data networks generally have been “box-to-box” rather than backplane memory or peripheral extension buses, such as the peripheral component interconnect. The latter are used to connect together cards within a box that implement a single function or form a single fault containment zone within the redundancy set (i.e., one replicant).

A few networks, such as SAFEbus, are actually system buses implemented in a backplane. The role of the network is what is important, not where or how it is implemented. Backplane system networks can be differentiated from simple extension backplanes by the fact that they have a higher level of safety criticality or that they may connect together multiple subsystems rather than just the components of one subsystem.

While development of these evaluation criteria were not intended to cover networks within a single-function box, subsystem, or those that connect nodes together within a single fault containment zone, these types of networks could have an impact on safety. For example, a generic failure in a backplane bus used in each copy of a redundant, safety-critical system could cause that system to fail.

If multiple cards and functions are connected by a single backplane network, then the common-mode influence and failure of the backplane network needs to be considered when the availability and integrity of these functions are justified. This is especially true if functions connected by data network infrastructure (backplane or box-to-box) are assumed to fail independently. In such cases, some of the evaluation criteria described by this Handbook may be equally applied to these internal networks. However, the scope of these criteria is not intended to entirely cover the case of internal subsystem networks or onboard networks, where all network connections lie within a common fault zone.

1.4.2 Protocol Stack.

Data network protocols are often designed to comprise multiple layers of functionality organized within a stack. Each layer is sufficiently independent of the layer above and below it so that the layer can be reused in other stacks. Generic models for the stacks have been developed. The most widely known model is the seven-layer ISO OSI model, as shown in table 1.

Table 1. Seven-Layer ISO OSI Model

No.	Name	Description
Layer 7	Application	This is the layer at which communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. This layer is not the application itself, although some applications may perform application layer functions.
Layer 6	Presentation	This is a layer, usually part of an operating system that converts incoming and outgoing data from one presentation format to another.
Layer 5	Session	This layer sets up, coordinates, and terminates conversations, exchanges, and dialogs between the applications at each end. It deals with session and connection coordination, authentication, and end-to-end encryption.
Layer 4	Transport	This layer manages the end-to-end control (for example, determining whether all packets have arrived) and error checking. It ensures complete data transfer.
Layer 3	Network	This layer handles the routing of the data (sending it to the right destination on outgoing transmissions and receiving incoming transmissions at the packet level). This layer routes and forwards the transmissions.
Layer 2	Data link	This layer provides synchronization for the physical level. It furnishes transmission protocol knowledge and management.
Layer 1	Physical	This layer conveys the bit stream through the network at the media and mechanical level. It provides the hardware the means to send and receive data on a carrier.

The widely used Department of Defense Advanced Research Projects Agency Network TCP/IP suite does not have an official stack description document. But it is known to have four or five layers, with the bottom three layers generally corresponding to the bottom three layers of the OSI stack.

For the embedded real-time systems used in the vast majority of safety-critical aviation digital electronics, models with a reduced number of layers have been used to provide lower complexity, latency, and overhead.

The lower layers of the stacks deal with the communication media and the hardware connected to it. The higher layers of the stacks represent functionality that is progressively abstracted further away from the hardware. When developing selection criteria for data networks that will be used in safety-critical systems, a question naturally arises as to which layers need to be evaluated. This question is equivalent to asking: What layers can affect the dependability of the overall communication system? This will depend on where the designers have implemented mitigation means, the type of failures that can be realized at a given layer, and the effect on the system safety analysis. In general, the highest layer, where communication dependability is considered, is usually identified as the transport (or equivalent) layer. Some networks handle dependability issues partly or entirely within layers below the transport layer, while others deal with them at the application layer that interfaces with the transport layer. In many cases, multiple layers will be needed to provide an acceptable dependability argument. Communication network hardware typically implements stack layers below the transport layer, and many of the networks proposed for aviation digital electronics systems only define these lower layers. However, some of these hardware devices also provide special application services that support system fault tolerance.

In general, the scope of the evaluation criteria described in this Handbook extends from the lowest stack layer, up through the dependability features of the transport layer or to the highest layer that is part of the network standard (or definition, if the network is not a standard), if that network does not include functionality up to the dependability features of the transport layer. Safety-critical embedded real-time systems often require services not included in generic protocol stack models, such as a time synchronization service. These special services will be included within the scope of these criteria.

1.4.3 Developmental Time Horizon.

The evaluation criteria were chosen so future data network communication technologies, as well as current technologies, can be evaluated.

2. DATA NETWORK CERTIFICATION ISSUES IN CONTEXT.

2.1 SUPPORTED APPLICATION REQUIREMENTS.

When evaluating a data network for a particular aviation digital electronics application, one must begin by establishing the requirements for that data network. The requirements placed on a data network are highly dependent on the aviation digital electronics architecture that will employ the network.

To develop requirements for an avionics data network, the results from the following tasks should be captured:

- Establish the most critical system failure condition for each data on the network. Determine the failure states of the data that created that condition. Establish the associated probability and assurance requirements.

- Define the network functions that are required by the system's applications.

- For each data element on the network, examine network function failure on the following classes of failure:

  - Inability to provide data

  - Failure to meet specified criteria (e.g., timing, lack of corruption, latency, sequence, identification, etc.)

- Establish the fault containment and fault tolerance requirements from the system safety analysis for data hosted by the network. This may require high-integrity message integrity checks or redundant paths.

- Determine whether the system requires the data network to coordinate action or consensus between different networked components (e.g., for synchronization, fault diagnosis, or voting).

Such a top-down examination of network use is needed to establish a context for the detailed analysis of the network lower-level properties and behaviors presented in the following sections. Therefore, without this system context, the justification of databus and network suitability or unsuitability is impossible to determine.

2.2 MULTIPLE-REQUIREMENT ENGINEERING TRADES.

As with any complex technology, the selection (or creation) of data network technologies requires the evaluation of how well a particular technology alternative meets a large number of requirements, many of which are contradictory. What is more important: size, weight, power, cost, bandwidth, latency, availability, integrity,...? Technology trades must consider all requirements simultaneously. Therefore, the relative importance or weighting of these requirements must be established. After the relative importance rankings of the requirements

have been established, ratings of how well a particular technology alternative meets each requirement must be created. Then, the multiple requirement trade-off can be done.

These trades are most often done in a linear compensatory manner. That is, for each alternative, its goodness value for a particular requirement is multiplied by the ranking or weight of that requirement; then all of these products are summed to get the overall value of that alternative. The best alternative is the one with the highest sum. This process can be represented by the formula

$$V_a = \sum_{r=1}^{n} w_r v_{ar}$$

where:

V_a = Value of the a th technology alternative
w_r = Weight or importance of the r th requirement
v_{ar} = Value of the a th alternative with respect to the r th requirement

Some requirements may have a minimum acceptable level. That is, any alternative that fails to achieve this minimum level will be rejected, regardless of how well it does against other requirements. Above the minimums, how well an alternative meets one requirement can be traded for another requirement. However, even the minimum acceptable levels may be adjustable, because a data network is not solely responsible for any particular system characteristic. For example, a data network by itself cannot guarantee system safety; it is only one component of the system (although it may be the most important component). One can trade-off the characteristics required for safety, e.g., data integrity, between the data network and any architectural mitigation for that characteristic.

The criteria for accepting data network technologies and component implementations with respect to a certification process constitute only a subset of the requirements typically considered when doing a data network trade study. This can lead to a multidimensional trade-off. For example, the inclusion of a certain level of data integrity in a network may force other characteristics to fail their requirements (e.g., excessive size, weight, and power). However, moving the responsibility for the integrity to some other part of the system design may not incur the same level of problem.

2.3 SYSTEM ARCHITECTURE AND DESIGN.

It is not possible to evaluate data network technologies and components without regard for the specific architecture and system design within which they will operate. Note that architecture as used in this context is not synonymous with high-level system design. Here, architecture refers to the set of rules for design, i.e., meta design or design for the design. Architectural rules can apply in any level in a design hierarchy. For example, an architectural rule may be that triple-modular redundancy will be used for fault tolerance. This rule could be applied at a high level, where three boxes of electronics are voted on, or it could apply at a low level, where three memory chips are voted on.

A particular system design may not need certain features of a data network. Shortcomings in a particular data network may be mitigated by architecture. Architectural mitigation could make almost any data network technology work. However, this extreme mitigation may require the use of a large number of replicated networks used in a manner completely outside the original intent of the network. For example, each node in a network could be connected to all other nodes in the system via one-way point-to-point Ethernet links. This would eliminate the nondeterminism of Ethernet because each link would have only a single transmitter. However, this would be very expensive; for N nodes it would require N^2 Ethernets. The topology would not be a standard Ethernet bus or star; it would be a fully connected mesh.

Architectural mitigation schemes should be carefully analyzed. This is especially true when a number of local architectural mitigations may have been employed but not analyzed in terms of the overall system safety requirements. The coverage of one or multiple layers of architectural mitigation should be analyzed for the total effect of their coverage at the system level. Credit should not be given for multiple layers of dubious mitigations.

2.3.1 Determinism.

Determinism is a widely discussed characteristic of digital electronic systems and, in particular, data networks. Very often these discussions narrow the definition of determinism to be applicable only to MAC (see section 4.1). While MAC is an important area to demonstrate determinism, it is not the only area. Determinism has a much broader applicability. In general, determinism means that the behavior of a system can be determined a priori. That is, given knowledge about the system's current state and a sequence of events that will effect the system, one can predict how the system will behave. If one cannot predict the behavior of a system in this way, one cannot claim that it has determinism as a property. In most cases within civil aviation, the property of determinism is needed to contribute to claims in the system safety analysis. Thus, determinism is an essential characteristic of a system that is used in safety-critical applications.

An obvious question is: How accurately must the behavior be known? Most safety-critical aviation applications are real-time systems. That is, for the system's behavior to be correct, its outputs must have correct values with correct timing. There are two types of timing determinism, ordinal and cardinal. Ordinal-timing determinism means that the order of events can be determined a priori. Cardinal-timing determinism means that the time between events can be determined a priori. The degree of precision required for values and timing is specific to each application.

Since all avionics data networks are digital, the behavior of which can be modeled as a finite state machine, value determinism can be established in the absence of failure, including normal failures such as those caused by intersymbol interference (ISI), metastability, or single-event upset. An evaluation of a data network must first establish the requirements for ordinal-time determinism and the required degree of cardinal-time determinism. Then, the evaluation must consider if the data network can be proven to meet the required time determinism. Again, this time determinism applies to more than just MAC. For example, the effect of nondeterminism in the arbitration for local memory between a network interface and the processor it supports may

make it difficult to prove the correct behavior of the processor or the network interface under all possible timing conditions.

2.3.2 Robust Partitioning.

The concept of robust partitioning is such that a function that is robustly partitioned from other functions cannot be adversely affected by those other functions. Partitioning analysis provides assurance that one partitioned function's behavior does not unacceptably affect the behavior of another. With a common data network, there are many opportunities for partition violations to occur. Every data network used in a partitioned environment should be rigorously analyzed with regards to maintaining partitioning integrity. Because partitioning is a system property, it is beyond the scope of this report to comprehensively detail the issues associated with it. However, there are specific areas where the impact of some network feature is elaborated to illustrate the potential effect on partitioning.

3. PHYSICAL LAYER.

The lowest level (layer 1) of most data communication reference model stacks is the physical layer (see section 1.4.2). The function of the physical layer is to send and receive communication symbols via network media. Layer 1 defines mechanical characteristics (such as connector configuration) of the media and characteristics of the signal. The physical layer is responsible for transferring individual bits through the communication media.

Because the physical layer is the foundation that all other protocol layers depend on, any failure in this layer will adversely affect all the layers above it unless adequately mitigated. An obvious question that must be answered is, What is the probability of failure in the physical layer? Failures at the physical layer can be grouped into two main sources, bit errors and component failures. The probability of faults in both of these sources depends on the environment.

3.1 ENVIRONMENT.

Data network components must meet the requirements of an aviation digital electronics environment, as described in RTCA DO-160. This means that the data network components not only must survive this environment, but also must simultaneously satisfy all requirements placed on the data network while residing in this environment.

Criterion 1 Environment: Does the network specification(s) and available components allow for the creation of a network that meets the applicable aerospace environment requirements of the most recent version of DO-160 or other imposed environmental requirements?

Was the components' environment evaluations done in a configuration and with a behavior that will be seen in the fielded system (e.g., the length and type of media segments, the size and rates of various types of messages)?

3.2 PROBABILITY OF BIT ERRORS.

Physical layer specifications often state a bit error rate (BER), which gives the probability of error for each bit. However, these BER specifications are usually created from unknown or irrelevant conditions. For a BER specification to be accurate for a particular implementation, the tests and analyses used to establish the BER must use the same modulation and encoding schemes, noise amplitudes, driver and receiver devices, communication path impedances, and clock quality as to be expected in the worst-case implementation. The BER numbers claimed by the manufacturers of data network components are almost always generated by BER test equipment that use a linear feedback shift register-generated pseudo-random bit sequence that bears no resemblance to the network's actual traffic. Either these BER numbers must be disregarded and tests with actual traffic done or some demonstrable arguments must be made for the BER pattern being a worst-case bound on the actual traffic behavior.

The designs of data network error handling and data network reliability analyses are often based on the misconception that bits traversing a data network are independent of other bits on the same network medium. However, designers of data network physical layers are familiar with a phenomenon called ISI that can cause some bits to affect other bits on a data network. A bit can be adversely affected by other bits that are relatively near (due to level shifting) or by bits that are relatively far away (due to distant reflections). With the phenomena of ISI and its causes now well established, it is hard to argue that bits fail independently.

To summarize, the probability of bit error criterion is that the probability of bit errors has been calculated correctly; the tests providing the inputs to these calculations have been done, accounting for the issues described in this section; the calculated error rate was slow enough to meet system requirements; and justification for the probability calculation was available for regulatory review.

Note that there is a close relationship between the probability of bit errors described here and Criterion 2 in section 4.2, Line-Level Encoding, and Criterion 8 in section 4.4.3, Interactions Between Line-Level Encoding and Error Detection.

Criterion 3 Probability of Bit Errors: Was the BER (or symbol) determined under the worst-case conditions expected to be encountered in the application environment?

Has an upper bound on anticipated BER (or symbol) been established by testing, using the worst-case-anticipated signal path characteristics (e.g., impedances and impedance discontinuities), environment (e.g., DO-160), local clock and clock recovery characteristics (e.g., drift and jitter), sampling margin (worst-case eye pattern), encoding schemes, and data patterns? If the network uses signal regeneration without elasticity buffers, the worst-case-accumulated jitter must be included in this determination. If the data network exploits the mixing of dominant and recessive signals on its media to perform some logic function, have the receivers been designed to tolerate "Wired-Or" glitches or have suitable design rules been created that limit the glitch duration to one that can be tolerated?

3.3 PROBABILITY OF ELECTRICAL COMPONENT FAILURES.

While there are statistics available for determining the failure rate of electronic components, there are insufficient statistics to address the probability of various types of behavior that can be caused by these failures. Too often, analyses restricted to credible failures have been too narrow. Failure modes that have been classified as incredible have actually occurred and must therefore be considered when doing a safety probability analysis (e.g., fault tree, FMEA, etc.). Any such analysis must not dismiss fault behaviors as incredible without a supportable basis for doing so. In particular, integrated circuits must be considered capable of producing any arbitrary output, within the limits of the power supplied to it. And more specifically to data network nodes, the "babble" failure mode cannot be assumed to produce only incomprehensible noise. This noise could be interpreted as a legitimate transmission. Of course, one would expect that the more complex a device is, the more complex its faulty behavior can be. However, there is no way of quantifying this expectation.

Criterion 4 Probability of Electrical Component Failures: Do the data network's electronic components have established hardware failure rates (permanent and transient) and characteristics so that avionics designers can do the required FMEA and fault tree calculations. Is the protocol defined well enough that one can determine the effects on the protocol by any and all component faulty behaviors?

Are signal margins robust enough to handle aging effects on connectors, media, and drivers; i.e., Could aging cause a network to not meet the aerospace environment requirements of DO-160 or other imposed environmental requirements at some point in the avionics system's lifetime? Has a valid synchronizer metastability error rate test been done for every synchronizer design in the data network's electronics and has an acceptable ceiling on the error rate been established?

3.4 ELECTRICAL ISOLATION PROPERTIES.

Most real-world, total system failures are due to a combination of factors. Very often one of these factors is the lack of fault containment. One important aspect of fault containment is the electrical isolation between redundancies so electrical failures, such as power supply overvoltage, are not propagated across defined fault set boundaries.

Many fault-tolerant architectures include the concept of receive-only nodes. The required characteristic of these nodes is that they can receive information that is transferred across the media, but are prevented from having any effect on any shared media. In these architectures, it is essential to provide assurances that these receive-only nodes truly cannot affect the data network.

The electrical isolation criterion is that the network must not provide a conduit for electrical fault propagation through fault containment zone boundaries. In addition, network connections, which by architecture design must be passive, such as receive only connections, must be shown to have the required passive properties even in the event of failures.

Criterion 5 Electrical Isolation Properties: Is there sufficient protection against electrical fault propagation?

Does the data network's physical layer allow for electrical (galvanic) isolation among redundancies? If a fault in a node causes the highest voltage in that node to appear on one side of this isolation barrier, can the isolation prevent damage on the other side of the barrier? If an application of the network requires receive-only connections to the media to prevent fault propagation, can it be shown (with sufficient assurance) that these receive-only connections prevent fault propagation to the media?

3.5 PHYSICAL COMPOSABILITY.

As the size of a network grows in the number of nodes, links, taps/splices/wyes, link distances, etc., performance or signal quality can suffer. A well-designed data network anticipates the effects of growth and can work correctly with any size network up to explicitly stated limits. Design rule effects and restrictions for the expansion of the data network must be included in the criteria.

Criterion 6 Logical and Physical Composability: Does the data network have characteristics or design rules that will guarantee that it will reliably work with any size network, up to an explicitly given maximum size?

For a data network that has the freedom to assume a number of different topologies or topology variations, does the network have characteristics or design rules that will guarantee that it will reliably work with all possible variations that are not precluded by its design rules (including sufficient design margin)? What is the certification affect of changing the size of the network (e.g., number of members in a clock sync algorithm)? Do these design rules and characteristics include the effects of hubs, repeaters, or other devices that extend network propagation delay or electrical loading? For data networks in which bit rate is limited by network distance, are there design rules to ensure that a particular network distance supports a given bit rate under worst-case conditions?

4. DATA LINK LAYER.

The data link layer is the layer immediately above the physical layer in most data communication reference model stacks. It provides the functions, procedures, and protocols needed to establish, maintain, and release data link connections between the nodes of a network. A conceptual level of data processing or control logic in the hierarchical structure of a node is responsible for maintaining control of the data link. The data link layer's functions include bit injection into the transmitter and bit extraction at the receiver; address and control field interpretation; command/response generation, transmission, and interpretation; synchronization; error control; and flow control.

The data link layer is divided into two sublayers: the MAC and the logical link control (LLC). The MAC sublayer controls how a node on the network gains permission to transmit on it. MAC protocols often try to provide prioritization or fairness in granting access to the media. MAC protocols also try to maximize the use of the media and minimize the probability of starvation

(not granting access to requesters). The LLC sublayer controls frame synchronization, flow control, and error checking. Conceptually, the LLC sublayer sits on top of the MAC sublayer.

4.1 THE MAC.

The MAC sublayer is a particularly important part of a data network's protocol when the network is used for real-time systems. Simple problems in the MAC can cause significant loss of the services that the real-time system needs from the data network. These problems include: no access (starvation), not enough access, or wrong time access. One source of these problems is the design of the protocol coupled with access demands and timing of clients, including faulty clients that fail to follow the behaviors expected or required by the data network specification.

Other problems can be introduced by failures (including permanent and transient failures) in the hardware that directly controls or accesses the network media. These failures may be introduced by any of the sources described in section 3, Physical Layer. The possible brittleness (lack of robustness) of the MAC protocol is of particular concern. That is, does the MAC protocol amplify the effect of small failures and errors such that they become large problems? For example, Does the MAC protocol allow transient failures and errors to have an effect that persists longer than current transmissions?

Problems unique to each type of MAC, such as master and slave, bit-dominant arbitration, CSMA/CD, time division multiple access (TDMA), token passing, and minislotting, are discussed in appendix A.

Many data networks used in dependable, real-time systems use the hardware from an existing data communication network that has an inadequate MAC and apply a substitute MAC on top of the existing hardware. This effectively disables the existing MAC without removing its hardware. Many such networks are based on IEEE 802.3 (Ethernet). The system designer must consider whether the unused hardware can cause problems under unintended circumstances.

Criterion 7 MAC: The MAC sublayer protocol must provide appropriately small bounds on message delivery times regardless of likely faults.

Can the behavior of one or more network clients increase latency and jitter beyond the desired bound for other network clients? If collision is used as a protocol element, are there bounds on delays incurred? Can misbehavior of one network client disrupt more than one or two other network clients? Does the MAC sublayer protocol amplify small failures and errors into loss of MAC sublayer protocol services? In particular, does the MAC sublayer protocol allow transient failures and errors to have an effect that persists longer than current transmissions? Does the MAC sublayer protocol specify a single physical point of failure, such as a dedicated protocol master node? Does the MAC sublayer protocol have a single logical point of failure, such as the current token holder crashing in a token-based protocol, duplicated tokens, or a "babbling" node asserting it has high-priority traffic to send? Does the MAC sublayer protocol have a bounded worst-case bandwidth at maximum network loading? If a mixed or hybrid MAC is used (e.g., master and slave polling on top of Ethernet hardware), are there conflicting properties within the portions of the hybrid MAC sublayer protocol that could cause vulnerabilities? Does the MAC sublayer protocol affect message delivery ordering?

4.2 LINE-LEVEL ENCODING.

Line-level encoding is the way that logical data is physically represented on a data network. As discussed in section 3.2, bits on a network can affect each other via ISI. The characteristics of a network's line-level encoding can heavily influence ISI. In addition to affecting the data network's own signal quality, line-level encoding can also affect other equipment via radiated emissions. It is important to determine whether the spectrum radiated from the line-level encoding has components in frequencies that can adversely affect other equipment.

Criterion 2 Line-Level Encoding: Was the electromagnetic compatibility testing (e.g., DO-160 Sections 19-22) done with the actual data line encoding (e.g., Manchester, 8b/10b), worst-case network data, message sizes, bit rate, pulse widths, and message repetition rate(s)?

4.3 MESSAGE FORMATING (FRAMING).

A message (also known as the frame or packet) may contain control and addressing information, as well as error detection, for example, cyclic redundancy code (CRC) information or forward error correction information. In evaluating the dependability of a message format, one must examine the consequences of any part of that format having an error, including the possibility that an error could cause the loss of many messages.

Some information that is transmitted in a message in one protocol (where it is vulnerable to errors) may not be transmitted in another protocol. For example, there are table-driven protocols in which all addressing, length information, etc., are held in a memory protected from errors rather than being transmitted on the network. There also are protocols that use redundant signal lines for error detection and correction instead of adding check bits to the message. Combining these two ideas, one could have a data network (such as SAFEbus) where messages have absolutely no overhead; every message bit is a data bit.

Criterion 8 Message Formatting (Framing): Message framing must ensure that only complete, properly synchronized messages are accepted at clients, and that improper synchronization is recovered from in bounded time.

Are there distinctive preamble or postamble bit patterns, including break characters, used to delimit messages? Is there sufficient hamming distance (HD) and bit-slip tolerance present to prevent an ordinary data pattern from being interpreted as a message preamble under expected error conditions? Do receivers tolerate any preamble bit(s) being erroneous? Do receivers tolerate any preamble bit(s) other than the one(s) that would make the preamble look like a start delimiter? Is the HD between the start delimiter bit pattern and any shift of the preamble bit pattern always greater than one? Is the data network's framing structure brittle (i.e., do simple errors cause a node to lose more than one message per error)? Are there parts of a message where an error could cause the loss of more than one message? Are there two independent checks on whether expected and actual frame length match (e.g., a length field and an unambiguous end delimiter; distinctive start and end delimiters)? If this is an implicit token protocol (reservation CSMA or minislot system) or time-sliced protocol (TDMA), is there

sufficient information in the message to validate that the time position of the message is interpreted correctly, avoiding incorrect interpretation of the message due to timing inaccuracies?

4.4 ERROR DETECTION.

Network criteria that have significant influence on the overall safety are the error detection capabilities of the link layer. This section discusses error detection of the link layer. Link layer errors can occur in the communication media, in its drivers and receivers, or in intermediate nodes (such as repeaters). Section 5 addresses some error detection mechanisms that may reside in the equipment at the ends of the network or at intermediate stages within the network.

4.4.1 Protocol Violation Error Detection.

Many protocols can detect errors by checking for protocol rules or format violations. The probabilities of detecting errors via protocol violation checking should be combined with message data error detection coverage to determine the probability of detecting erroneous messages.

4.4.2 Parity and Frame Check Sequences.

There are many different error detection mechanisms and encodings, such as CRC, Fletcher, Adler, AND, XOR, etc. with different characteristics; however, this document focuses only on the characteristics of a few representative mechanisms.

CRCs are one of the most commonly used error detection schemes. The metric most commonly used for determining the quality of CRC error detection is HD, i.e., the minimum number of independent bit flips that can result in an undetected error. Given the HD and BER for the medium, the designer can compute the probability of an undetected error. This probability should be sufficiently small for the reliability requirements of the data network. However, this coverage assumes that the bit errors are independent, an assumption that is known not to be true with the existence of ISI. With interbit dependencies, the calculation for error detection probability would have to be adjusted accordingly.

4.4.3 Interactions Between Line-Level Encoding and Error Detection.

In addition to the effects of ISI causing interbit dependencies on the medium, line encoding can cause further dependencies among the bits due to the encoding and decoding processes. Typical line-encoding transformations include symbol encoding (e.g., Manchester, 4b/5b, 8b/10b) that produce symbols (each symbol consists of a block of bits), bit stuffing, and phase encoding. For most of these commonly used line-encoding transformations, the corresponding decode processes can cause error expansion. That is, even a fault-free decode process can create more errors on its output than it has on its input. To accurately calculate the coverage of in-line, error detection mechanisms, this error expansion must be taken into account. On the other hand, the line encoding itself often can detect errors on its own. Because the interactions between the line-encoding mechanisms and error detection mechanisms are generally ignored in coverage calculations, published error detection coverage values for most networks are incorrect and must be recalculated.

Criterion 9 Error Detection: The Data Link Layer must provide sufficient guarantees of message delivery free of undetected errors, using error detection mechanisms with coverages that have been calculated correctly.

Does the network meet the required integrity values (undetected error probabilities and HD) for the worst-case error pattern requirements (error bursts, maximum BER, and temporary blackouts)? Does the network deliver valid messages within bounded latency with sufficient probability despite expected error rates? Does the network meet availability requirements for the worst-case error probabilities and distributions? Have the potential effects due to encoding of the data on the physical layer and its implications to the coverage of error detection been quantified? For example, have corrupted bit-stuffing formats been accounted for in an error analysis? Are messages vulnerable to corruption of length fields that cause receiving clients to use the incorrect frame location for frame check sequence (FCS) fields? If it is a required service of the network, can receivers of data be certain of the sender's identity, even in the presence of faults? If messages use FCS hidden data (data that is included in the FCS calculation but not transmitted as part of the message), has there been an accounting for the loss of FCS coverage?

5. NETWORK LAYER, TRANSPORT LAYER, AND NETWORK MANAGEMENT.

In the OSI model, the network layer provides switching and routing technologies, creating logical paths (known as virtual circuits) for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, inter-networking, error handling, congestion control, and packet sequencing. Above the network layer, the transport layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer. In embedded systems, the functionality of these layers is often merged into a single layer of functionality. This section discusses the issues related to the functions of both layers together. In addition, in some newer protocols (for example, Time Trigger Protocol/SAE Class C (TTP/C), FlexRay[2], and Avionics Full-Duplex Switched Ethernet™), a network management layer is emerging to describe hardware or software services that facilitate message agreement, network diagnosis, and synchronization. Related issues are also discussed within this section.

5.1 NETWORK VULNERABILITY TO ADDRESSING INFORMATION FAILURE.

Errors in message content labeling or node-addressing information can cause serious problems in a data network. An example failure mode is the masquerade failure, where one network node can impersonate another node of the system. For the network to be dependable, there must be mechanisms to handle these errors. The influence of software on network-addressing information is also an issue, as discussed in section 5.8. Network technologies that use configuration tables for network routing and addressing are also vulnerable errors. Addressing that is done via tables created at run time have slightly less error exposure than protocols that include addresses in every message. Addressing that is done via tables created at design time have even less exposure.

[2] FlexRay is a registered trademark of the DaimlerChrysler AG Corporation.

Network error-handling logic that may be invoked by erroneous addressing information or that may impact protocol flow needs to be analyzed to establish a bound on the influence of the invocation of the error-handling logic and its impact (degradation) on network performance. The behavior of any such logic and its associated vulnerability needs to be analyzed and justifiably bounded. This is especially true for centralized intermediate stages, as discussed in section 5.3.

Criterion 10 Network Vulnerability to Addressing Information Failure: Mechanisms should ensure correct forwarding, routing, or conversion failures despite likely failure scenarios of network components.

Does the network technology use message addressing or message identification fields? Does the network technology implement mechanisms to detect or mitigate the corruption of message addressing or message identification fields? Has the fault coverage of this detection and mechanism been established? Are the message-addressing or message identification fields vulnerable to host software corruption? Does the network technology use tables to assist with message addressing and routing? Does the network technology implement adequate checking mechanisms to ensure the run-time integrity of the routing tables? Does the network technology build routing information at run time? Are the algorithms and associated mechanisms used to build run-time routing tables vulnerable to corruption or run-time errors? What network action causes the network routing tables to be rebuilt? Can erroneous node software or electronic and electric hardware invoke incorrect invocation in the table-building activity? Is the network routing discovery time suitably bounded? Can addressing or routing errors cause lost packets or fragments to circulate on the media and inordinately consume needed bandwidth? If multiple layers of addressing are used (such as MAC and Internet protocol) then errors and masquerading faults in all levels of addressing must be considered.

5.2 NETWORK VULNERABILITY TO FLOW FAILURE.

As with network-addressing failures, the network technology's flow regulation logic also needs to be evaluated. Issues relating to acknowledgement and retry logic are discussed in section 5.9. Issues relating to host interface load balancing and buffering are discussed in section 6.1. Issues relating to intermediate stages are discussed in section 5.3. Because of the complexity of modern protocols, it is far from obvious how far-reaching the misbehaviors propagated from a failure can be. It is recommended that a protocol level FMEA be performed.

5.3 IMPACT OF INTERMEDIATE STAGES.

If a network encompasses intermediate buffering or relay stages, then the behavior, implementation, and impact of the intermediate stages needs to be established and evaluated with the network behavior. In critical networks, it is common for such intermediate stages to incorporate error detection or fault containment mechanisms. This section discusses some of the issues and network attributes related to such intermediate buffering schemes that need to be considered and evaluated.

5.3.1 Vulnerability to Intermediate-Stage Failure.

In networks that deploy intermediate stages, the influence of the intermediate-stage components may be significant. For example, in networks using stars or hubs, the intermediate-stage component impacts all data flowing through it. One of the difficulties in analyzing the possible adverse impacts of intermediate stages is bounding the failure modes of the intermediate-stage component. If a network intermediate stage is developed to have full coverage (for example using self-checking or monitoring schemes), then the failure modes of the intermediate stage component causing data failure may be suitably justified as benign (e.g., fail-stop, assuming that fail-stop is an acceptable mitigation in the failure analysis). It is imperative, however, that the coverage of the self-checking or monitoring scheme be suitably justified, as discussed in section 7.4.

It is common for networks to rely on in-line integrity mechanisms, for example, CRCs checksums, parity, etc. In such cases, the failure modes of the intermediate-stage component become more significant. With complex intermediate-stage logic, it is difficult to bound failure modes of the intermediate devices. Without such bounds, one cannot determine the coverage of in-line integrity mechanisms.

5.3.2 Vulnerability of the Intermediate Stage to Fault Propagation.

The vulnerability of the network intermediate stages to faults propagating from erroneous end nodes should be analyzed. Such vulnerabilities may be related to erroneous control data or erroneous temporal behavior. Any error-handling logic that may be invoked in response to erroneous end-node traffic and behavior should also be analyzed so that any associated intermediate-stage or -switch performance degradation or other propagated erroneous behavior can be suitably bounded.

Criterion 11 Impact of intermediate stages: Intermediate stages (e.g., repeaters, gateways, routers, and switches) must guarantee sufficient availability and integrity, as well as sufficient logical and physical independence from other replicated intermediate stages to ensure correct operation.

If the network uses intermediate stages, Is the availability of the intermediate stage sufficient to fulfill network channel availability requirements? Are network intermediate stages for different network channels independent? Is intermediate-stage-to-intermediate-stage signaling required between independent network channels? Does the intermediate-stage-to-intermediate-stage signal path introduce any fault propagation or common-mode influence? Do network intermediate stages incorporate sufficient fault detection and coverage? Can fail-stop, intermediate-stage behavior be justified? Does the network technology rely on in-line, integrity-checking mechanisms? Can the network intermediate-stage action introduce failure modes that will defeat network frame- or integrity-checking logic? What is the intermediate-stage response to erroneous signals? Do intermediate stages ignore erroneous framing and cleanup and reshape erroneous data streams? What is the intermediate-stage response to out-of-specification errors; i.e., elasticity exhaustion, etc.? Does the network perform store-and-forward action? Does the intermediate-stage perform recalculation of the integrity check sequences? Is the intermediate-stage buffer memory suitably protected from transient upsets? Does the protection mechanism

simply detect or does it detect and absorb transient upsets? Does the intermediate-stage response to transient upsets lower the availability of the intermediate stage, i.e., do transient errors force intermediate-stage resets and re-integration? What mechanisms exist to detect erroneous message forwarding; i.e., the forwarding of old messages or sending messages to incorrect addresses? Can intermediate-stage errors affect higher-order redundancy management mechanisms to reduce overall network availability? Can network intermediate-stage action introduce head-of-line blocking? What network mechanisms exist to mitigate these effects? Can babbling or other erroneous node action impact intermediate-stage performance, for example, result in buffer exhaustion? Can intermediate-stage start-up and reintegration time be bounded with a faulty node present? Can intermediate-stage start-up or reintegration be impeded by the erroneous action of intermediate stages on other channels? Are the failure assumptions of the interstage justified by a complete FMEA? Is it possible for a faulty guardian to cause irrecoverable network failures?

5.4 NETWORK CONFIGURATION DATA.

Many network technologies require configuration and routing tables to be programmed to assist network operation. Therefore, design correctness of these tables is obviously important to correct network operation. Design assurance issues relating to network table correctness are discussed in section 8. The run-time integrity of the tables is also important. Therefore, the storage, operation, and load integrity mechanisms of the configuration data need to be evaluated with the network technology. In networked systems, the consistency between the copies of run-time tables in different nodes is also an important issue. Hence, protocol mechanisms to ensure table consistency should be evaluated.

Criterion 12 Network Configuration Data: Network topology and component configuration data must be commensurate with the applications' fault tolerance and performance requirements—considering table production, loading, errors during operation, run-time environmental effects (e.g., impact of radiation), and maintenance actions.

Are network configuration data tables stored with sufficient integrity? How are network configuration tables loaded? What network mechanisms are used to ensure network configuration tables are not corrupted during loading? Does network configuration loading use the same network data paths as normal traffic? Are specialized load protocols used for performing network loading? Does the network incorporate sufficient interlocks to prevent the inadvertent invocation of such download protocols? Are network tables updated during live network operation? How are network modes and network table versions agreed at run time? Does the network incorporate maintenance or query protocols operating on top of the live network operation? Does the network incorporate sufficient interlocks to prevent the inadvertent invocation of such protocols?

5.5 START-UP AND RECOVERY.

Network start-up and recovery mechanism are important since, in critical environments, start-up and recovery time of the system is often a key attribute of the system performance. The behavior of network start-up performance is, therefore, another attribute that requires careful evaluation.

During start-up or recovery, the network is usually more vulnerable to faults. Many algorithms and mechanisms are designed to work correctly, only if some minimum amounts of good resources are available. However, if just prior to start-up, it appears that "everything has failed." A good design for start-up must be able to get past this "everything has failed" phase and be able to bootstrap itself up to full operation. However, all too often, network designs assume that the network was "born running" and cannot meet their fault-tolerance claims during start-up. On a related issue, a system experiencing more faults than its design fault limit must return to proper operation within bounded time after the number of faults are reduced to within its design limit.

Criterion 13 Start-Up and Recovery: Component and network integration, start-up, and recovery must be performed in bounded time, considering dependability requirements, environmental and deployment constraints, and interactions with different systems and applications (such as application-level timing impact and power architecture influence).

Is network start-up time bounded even in the case of fault scenarios? Are the network start-up and integration mechanism fault tolerant? Are the assumed faults consistent with the coverage and FMEA declarations? If network start-up requires coordinated power sequencing, is the required power-sequencing action assured to have the required network availability? Is the start-up dependent on single components? What are the effects of such dependence, considering failure scenarios? If host nodes are required to participate in network integration, start-up, or recovery, are those host node behaviors considered in the analysis? Is there a host or other node behavior within the protocol fault model that can cause repeated integration, start-up, or recovery events and, thus lead to unbounded time to achieve normal network operation even if each integration, start-up, or recovery attempt completes individually within bounded time? If there is more than one statically designated network master, is the leader election process guaranteed to converge within bounded time under worst-case assumptions and all faults within the specified fault model? If system start-up is inhibited in the presence of hardware failures (e.g., start-up is precluded with a network fault on one redundant bus), is the risk of system unavailability after an in-flight restart mitigated?

5.6 GLOBAL SYNCHRONIZATION.

Data networks may have a need for synchronization of clocks among nodes for coordinated network access or as an application level service. The following paragraphs focus on clock synchronization services, but a subset of the aspects to be considered for clock synchronization services are equally applicable to synchronization of logical clocks (counters) used for redundancy management.

Clock synchronization algorithms and mechanisms must be able to acquire initial synchronization, reacquire synchronization with a running network, and maintain synchronization even while experiencing data network faults, including Byzantine and masquerade failures. Clock synchronization algorithms and mechanisms are acceptable only if they have been subjected to formal proofs of correctness and the assumptions used in these formal proofs hold for the particular implementation being evaluated. Also, the quality of the clock synchronization (e.g., precision, jitter, and monotonicity) must meet the system requirements. Places that may cause derived requirements in this area include (1) time-triggered operating systems and communication buffer management systems that cannot tolerate

significant jitter or nonmonotonic time (periods where time appears to run backwards due to clock corrections) and (2) applications that use delta time for differentiation or integration.

Criterion 14 Global Synchronization: If synchronization is required, have the synchronization mechanism(s) been shown to work correctly under all defined scenarios, including faults?

Has the stability of the algorithm been analyzed under different environmental and expected fault conditions, including stability after power up of nodes and under all expected network configurations? Similarly, has the precision been analyzed and is it bounded under expected fault scenarios and operational conditions? Have the effects of single dependencies of synchronization reference data been considered (faulty or no synchronization data)? Are such dependencies adequately mitigated for the required safety levels? Have the effects of different synchronization data view (e.g., due to different propagation delay, data acquisition delays, etc.) been considered in the stability analysis? Have the effects of merging of data from different network paths and potential differences due to different paths and propagation delays been considered in the algorithm stability analysis? In algorithms using multiple clock sources, has the use or election of the source been considered under the assumed failure conditions and considering source coverage mechanisms? Are mechanisms in place that adequately verify or support that the data used for synchronization does stem from the assumed or elected source, i.e., the synchronization algorithm is not vulnerable to masquerade faults? Have effects of the clock correction been analyzed (such as task-time dependence on the clock synchronization and influence of the correction on the available (potentially decreased) execution time to tasks)? For asynchronous interfaces between two or more isochronous clock domains running at the same frequency, is there a mechanism to prevent pathological metastability (the metastability condition persists indefinitely due to the clocks not having any relative drift)?

5.7 FAULT DIAGNOSIS.

Some network technologies include fault diagnosis services to identify and isolate faulty member nodes. Such services are strongly related to group membership and interactive consistency services, which may use fault diagnosis services to manage network state-dependent application decisions and guaranteed consistent delivery of messages.

Requirements of group membership being consistent and effects such as inconsistent reception status of messages at different receiving nodes and potential consequences are discussed in detail below. In general, it should be said that any diagnosis service will be nonperfect, e.g., due to transients having local effects or due to failure modes of the sending nodes (e.g., Byzantine failure modes).

Diagnosis information can be used by the network to build additional services for management of redundancy sets or simply as acknowledgement. In this context, the use of the diagnosis information needs to be in alignment with the expectations of the applications.

Applications that use networks that provide group membership services should analyze:

- The underlying assumptions
- The consistency and correctness guarantees of group membership
- Their effects on the application level

Such analysis and effects on the application should also include temporal aspects, because diagnosis information lags in time and so does membership.

Criterion 15 Fault Diagnosis: Any diagnosis, detection, or system-level agreement mechanisms must justify fault assumptions and consider effects of diagnosis action.

What does the source of the diagnosis information rely on? What is the information's source integrity value and how is it in agreement with the assumptions? What is the influence of faulty relaying components, lightning, high-intensity radio frequency (HIRF), and other external effects on the diagnosis data and diagnosis algorithms? Have such influences been analyzed and quantified to have acceptable effects? Does the diagnosis assume certain failure modes? Are the consistency and correctness guarantees of the diagnosis information (such as group membership) in agreement with the application's use? Are the assumptions and properties quantified to the level required? Does the application use diagnosis information? Is the application's use of diagnosis information in compliance with guarantees and assumptions of the diagnosis information? Are there effects on the application level? Is the semantics of the group membership information (e.g., nodes operational or node not operational) in alignment with the application-level assumptions of its use; e.g., does the application assume correctness of the message even though the membership only indicates operation? Has the coverage of the mechanisms used to establish a nodes health with respect to group membership signaling been evaluated against requirements? Has the safe use of membership semantics been analyzed in "corner" cases, like start-up and integration of systems? Have temporal lags in error detection and diagnosis been quantified and found suitable? Have start-up and integration been included in the analysis of diagnosis and group membership?

5.8 CLIENT EFFECT ON NETWORK OPERATIONS.

A data network is often the glue that holds together a dependable system. A system data network tends to become either the main fault containment mechanism in itself or is a major component of the main fault containment mechanism(s). As such, it is important that a system data network is not adversely affected by the clients it serves, no matter how badly the clients misbehave. Many data networks allow their clients to influence the timing of network start-up by affecting the timing of their nodes. It is possible for data network protocols to take an inordinately long time to start, or they may not start at all, if the timing behavior of its nodes follows some pathological pattern during start-up. During data network operation, some protocols allow clients to adversely affect their behavior if the clients can control addressing, routing, priorities, etc. Some systems require applications of different safety criticality levels to share the network. When this is the case, the network must be robustly partitioned so that applications and clients of low criticality cannot adversely affect the use of the network by high-criticality applications or

clients. Another possible avenue for a client to adversely affect a system data network is via unprotected test or network management paths.

Criterion 16 Client Effect on Network Operations: Can a network client adversely effect network operations?

What is the assumed faulty behavior? If a restricted failure model of clients is assumed, what substantiates the restricted faulty behavior? Do the analyses of clients include start-up and integration scenarios and are they considered temporal effects of clients (such as potential long software response times)? Do client-side actions or data impact network addressing (i.e., message identifiers (labels or addresses) are written under client control). Does the network allow client impact of protocol-level control flow, such as mode changes? What network mechanisms are in place to ensure that such actions or data do not endanger the operation of the network? Do any protocol operations, such as start-up, require behaviors or constraints on behaviors of the network clients and, if so, are such behavioral requirements or constraints ensured in the system design?

5.9 ACKNOWLEDGEMENT.

For network protocols that employ acknowledgement schemes, the behaviors of this logic need to be carefully analyzed, especially with respect to inconsistent message reception (i.e., some nodes receive a message or an acknowledgement, while some do not). It cannot be assumed that any acknowledgement mechanism provides, by itself, consistent message reception (also called atomic broadcast). The implications of inconsistent message delivery, different message delivery times to the application, and multiple deliveries should be analyzed with respect to the overall system and its safety. In addition, if the acknowledgement mechanism calls for retries, this additional load must not cause the network to exceed its bandwidth and timing budgets (e.g., jitter).

Criterion 17 Acknowledgement: Does the acknowledgement mechanism work adequately under all fault scenarios?

What are the impacts of the acknowledgement mechanisms? (Are they fault tolerant)? Has the acknowledgement scheme been analyzed, assuming adequate failures modes (e.g., such as inconsistent or Byzantine message reception)? What are the effects of such failures? Have adverse effects been suitably compensated by other means? Have effects of negative acknowledgements (NACK) and message retries been analyzed during the performance analysis? Is the number of retries bounded, and is this bounded number determined as a consequence of the impact of retries on the performance? What are the effects of acknowledgement errors on the application? Are effects suitably bounded (e.g., with respect to processor overhead)? If acknowledgements are not supported, does the protocol support aging or staleness indications for periodic messages that have not been received for more than a certain period?

6. APPLICATION SERVICES.

Current data network technologies comprise a number of application services that may or may not be used by an application. All services need to be analyzed in the context of a safety

assessment. In its simplest form, buffer management has associated properties that need to concur with the application assumptions. Newer generations of data networks supply voting schemes or redundancy management mechanisms. An example for such buses is ARINC 664. In this section, the criteria for data network services used by applications are examined.

6.1 HOST INTERFACE MANAGEMENT.

Buffer management concerns itself with the message access order to the network, partitioning requirements, and performance aspects of the network interface buffer as well as implications to the host.

6.1.1 Client Buffer Queue Management.

Buffer management of systems may have system-level implications. One example of system-level impact may occur if messages are associated with priority. In certain combinations of buffers and accesses, priority inversion on the system-level may occur. When evaluating networking technology for the deployment in systems, client buffer queue management mechanisms should consider effects on the access to the network, such as fairness and implications to the network and the host.

6.1.2 Buffer Management Partitioning.

In a robustly partitioned system, software partitions running on a node have a strict execution budget and should adhere to their execution budget. On nodes where the data from a data network is managed by a direct memory access (DMA) controller, the DMA controller may repeatedly stall the execution time of running partitions, potentially having significant effect on the execution time of software tasks. Unless effects such as cycle stealing are accounted for in the execution budget of software tasks or the overall node architecture, software may miss execution deadlines. Unless the access to the common buffer is restricted or controlled for each partition, software partitions may overwrite messages of other partitions or use network resources from other partitions. A partition may even be able to send data masqueraded as another partition, unless protected. As stated in the introduction, a full treatment of partitioning related to databuses is beyond the scope of this report.

6.1.3 Buffer Management Performance Considerations.

The performance considerations of buffer management should be considered when selecting a network. In the past, the low-speed aviation digital electronics networks (such as ARINC 429 and 629) have put less emphasis on the performance of buffer management, because memory access times or memory bus access times were often an order-of-magnitude quicker than required for serving the data copying and coordination activities. With the advent of high-speed communication in avionic systems, the need for a balance on the buffer management side with respect to performance becomes more prevalent.

A push interface host management mechanism pushes data onto the network when the application has finished computation and has released the associated data buffer. In such models, primarily used in loosely periodic network approaches like ARINC 664, the data release

time from the application is dependent on the application performance because the data is sent when the application is finished with it and indicates it is available. The network data in such push models is dependent on the application execution schedule and its release of data to the network. If the release times and the data size are highly variable, the data released to the network can be bursty in nature. Such host interface approaches can lead to peaks in bandwidth use in integrated multidrop network systems, which can lead to longer data delivery times. Network analysis for sizing of the network bandwidth and related resources may need to consider application-level schedules and performance, taking into account application variations and worst performance due to the influence of release times on data patterns. It should be understood that the effect of release times can have system-level latency and jitter effects, e.g., the latency of data can change over a wide range due to nonaligned data patterns in integrated network systems. The resulting jitter variations need to be considered on the application level to ensure proper performance of the application in all possibly encountered latency values.

Criterion 18 Host Interface Management: The protocol's interface to the host application (including gateways) must provide promised prioritization, latency, and loss prevention of messages for supported categories of service.

Has the buffer management been analyzed with respect to effects and system-level implications (e.g., combined priority and buffer management causing head-of-line blocking or priority inversions)? In mixed criticality systems, are communications buffers reserved per partition and are assured not to be accessible for other software partitions in cases where there are partitioning requirements for software? Have potential masquerading effects due to buffer management been considered (e.g., software partitions masquerading as other partitions)? How is the access to the buffers being controlled (control between different software partitions and control of access between network and host software access)? Has this been analyzed with respect to safety (effects of different criticality partitions)? Are system-level effects of blocking buffer access being analyzed (e.g., increased software execution time and increased buffer size needs)? Has the performance of the buffer been analyzed with respect to the needs of the communication (i.e., is the network speed and the buffer management speed balanced)? Are there relationships between network configuration flow and buffer management configuration? Is there an effect of changing network tables on the buffer management tables or mechanisms? Have such relationships been considered during the design and what is done to ensure compatibility? Is there a way to ensure that received messages are not overwritten in the buffers before being retrieved, or that such overwriting is detectable and handled appropriately by the system an error condition?

6.2 SUPPORT FOR APPLICATION LAYER REDUNDANCY.

6.2.1 Support for Active Replication.

Networks may signal the application of reception status, which may assist the application in voting or selecting a correct value. Such mechanisms should be evaluated with respect to the mechanisms' correctness. In case the indication status stems from the same source as a possible faulty value, the use of such status information might be limited.

Application layer membership is a mechanism to manage the redundancy sets at an application layer. Such application layer membership algorithms should be evaluated with the same scrutiny as the node-level memberships described in section 5.7.

In some networks, the network host interface incorporates a life-sign mechanism to support application membership and health diagnosis. The life-sign action should be evaluated with respect to its effectiveness of detection of the failures.

6.2.2 Support for Passive Replication.

Some networks support mechanisms for passive-redundancy strategies, i.e., the ability of multiple network nodes to share network bandwidth. These mechanisms are discussed in section 7.7. The network's mechanisms to inform clients of the state of the passive-redundancy scheme (what application is in control and how many spare applications are online) should also be considered. Often a certain degree of transparency is required for the switchover between members of the redundancy set. The network will have to be shown to meet this required degree of transparency.

6.2.3 Support for Increased Integrity.

Some network technologies implement host interface support for self-checking pair host configurations. Self-checking pair data is compared, and if it agrees, it is delivered as correct. Self-checking, pairs-based input data should be compared before computation; otherwise, the self-checking pair computation results are likely to diverge, even though both halves of a pair are correct. Self-checking pairs should also be evaluated with respect to their independence from power, common memory, vulnerability, and common design faults.

Self-checking host support is strongly influenced by the network-level, self-checking mechanisms discussed in section 7.4.

Criterion 19 Support for Application Layer Redundancy: Any support for application layer redundancy must fully meet stated redundancy management mechanism requirements.

Has the application group membership service been analyzed with respect to its performance, availability, and integrity targets under adequate fault scenarios? How are failures in membership or similar services covered? What assumptions were made in determining this coverage? How are these assumptions verified? Are the assumptions of correctness and completeness of the application layer group membership service in agreement with the system-level assumptions? Has the effect of the lag in updating the application been analyzed in the system context? Do the provided services conform to the requirements from the application? How is the synchronization ensured between the different replicants that are to be voted or agreed? Any voting or selection scheme may lead to different results at different observers for certain failure modes (Byzantine or local nonreception of a single message). Has the impact of different results been taken into account in the safety analysis? Are there other mechanisms that may influence the selection logic? Are such mechanisms analyzed with respect to potential unwanted interaction? Is the network implementation vulnerable to masquerading faults

resulting in potential defeat of redundancy? Are passive-replication strategies analyzed with respect to control handover in failure cases? How is the state of faulty replica signaled to the application? Do the two halves of a self-checking pair communication interface compare their two copies of received data before it is allowed to be used for computation? Is this comparison designed to be immune to Byzantine faults? Are the halves of a self-checking pair suitably independent (e.g., independent power, separate memory, or memory sections)? If the same message information is sent over redundant buses, are those redundant copies cross-checked, or is the first seemingly valid message used? If the same message information is sent over redundant buses, are those copies sent at the same time or staggered to mitigate against correlated network disturbances?

6.2.4 Support for Robust Partitioning.

While robust partitioning (see section 2.3.2) is a characteristic of an architecture that is largely outside the control of data networking, there are certain facets of robust partitioning that may need to be supported from the network. First, the network should protect itself against misbehavior of any of its clients (see Criterion 16). Then, if required, the network should protect each of its clients from its other clients. This includes protecting robustly partitioned subdivisions of each of its clients from other clients or partitioned subdivisions.

Criterion 20 Robust Partitioning (ARINC 651): If the network is required to provide robust-partitioning guarantees, what has been done to substantiate any partitioning claims?

If required, does the network enforce robust partitioning among its clients, even if multiple clients share the same node? If required, does the network enforce robust partitioning among its partitions within its clients? Does the partitioning against software faults include vulnerabilities resulting from timing faults?

6.3 TIME SERVICE FOR TIME STAMPING AND TIME INTERRUPTS.

Application time services that may be supplied by the data network include time stamping and time interrupt. Synchronization aspects of time are discussed in section 5.6, including a discussion of the implications of time services to the applications. The quality of time services can be adversely affected by a data network time-service design that is not robust.

Criterion 21 Time Service for Time Stamping and Time Interrupt: If time-stamping and time-interrupt services are provided, are they sufficiently dependable and robust?

Is the time service robust with respect to potential implications such as effect of clock differences? Is the use of time-stamping and time-interrupt services adequately mitigated or included in a robustness analysis of application data algorithms? If the protocol is based on a centralized time service, is failover for the time master supported? If it is based on a distributed time service, is timekeeping maintained even for Byzantine clock faults?

7. FAULT TOLERANCE MECHANISMS.

Some network technologies incorporate fault tolerance mechanisms to mitigate the failure of network components such as guardians and monitoring schemes. Such mechanisms may be particularly advantageous in aviation digital electronics environments in which high-network availability and integrity is required. These mechanisms and associated evaluation criteria are discussed the following sections.

7.1 TOPOLOGICAL FAULT TOLERANCE.

The network topology may have significant impact on the network tolerance to zonal or spatial proximity faults, e.g., physical damage that affects a certain area of the vehicle. If the network uses a bus topology, then any failure along the bus path may destroy network availability (even for signals that do not have to traverse the failed part of the bus path). The bus zonal vulnerability is particularly important if multiple redundant buses are assumed to increase network availability. If all units are connected to all buses, then all buses are required to be in physical proximity at the point of their interface to the other nodes. A failure at this point of interface may, therefore, damage all of the independent bus channels.

Networks using intermediate stages may perform better in relation to zonal fault tolerance, as the point-to-point relaying action of such technologies alleviates the impact of physical layer damage. However, the placement and data-path planning of such intermediate-stage schemes should also be considered as the network technology is mapped to a vehicle architecture; i.e., there is little benefit in placing two redundant, central intermediate stages in the same location.

Criterion 22 Topological Fault Tolerance: Redundant network components must be physically separated and isolated to prevent correlated outages due to physical equipment damage, loss of electrical power, and credible media faults.

Is the network vulnerable to spatial proximity faults, such as physical damage that is within a specified distance limit? If a single piece of equipment is faulty, can its faults propagate to take down the entire network (e.g., Can a single fault cause babbling behavior simultaneously on redundant network paths?)? Is an adequate communication path availability ensured despite faulty end equipment (babbling devices or short circuits)? Are common network resources, such as switches, placed at adequate distances from each other so as not to be vulnerable, but to be independent of physical damage or spatial proximity faults? Are redundant resources attached to different power sources within the system?

7.2 GUARDIAN SCHEMES.

Some network technologies incorporate covering functions or guardian mechanisms to contain node faults. Such guardians may be argued to increase network availability. However, the implementation and performance of the guardian function needs to be carefully evaluated to verify that suitable coverage and independence is provided.

Irrespective of any guardian implementation, it is imperative that suitable tolerances for guardian enforcement action are established to provide suitable design margin. As with other critical

protocol parameters, these tolerances should accommodate worst-case aging and expected life-time degradations of all components related to the guardian. The criteria for establishing suitable guardian parameterization would ideally be formalized and verified, as discussed in section 8.

Latent failure of guardian schemes is another consideration; this is discussed in section 7.6.

Criterion 23 Guardian Schemes: Some techniques, such as network guardians, must be used to ensure that a single-point client failure will not take down the network.

Does the network deploy guardians? What is the intent of the guardians? What coverage is the guardian assumed to provide (what failure modes can the guardian detect or contain)? What is done to substantiate the coverage claims? Does the claimed coverage of failure modes consider boundary system conditions such as start-up or integration? What is done to ensure independence of the guardian (power, time, logical dependence, and physical dependence)? Have potential side effects of the guardian behavior been considered (especially central guardians)? Is there an effect on in-line coverage due to guardians? Have tolerance margins (different oscillators) due to difference between the guardian and the guarded device been established and quantified?

7.3 PROTOCOL LOGIC FAULT TOLERANCE.

Networking technology may also incorporate protocol flow and algorithmic fault tolerance strategies, i.e., voting on protocol-state information or required protocol actions. The strength of such protocol mechanisms should be evaluated in the context of the coverage provided by the network implementation. For example, if all nodes are self-checking, then little protocol-state fault tolerance is required as all protocol errors are contained at the source and justified to be benign. Similarly, if the guardian mechanisms contain protocol flow errors, then less protocol-state fault tolerance is required. However, if suitable fault containment or coverage cannot be established, the protocol layer vulnerabilities to erroneous state and addressing information should be evaluated.

If fault-tolerant protocol logic is implemented, the impact of its protocol algorithms will also need to be evaluated. This means that any protocol layer mechanism needs to ensure the required agreement on protocol state for integrity and the required replication for availability. Often in two-channel systems, there is a conflicting goal between availability and integrity. Hence, mechanisms to improve protocol integrity may reduce protocol availability.

Criterion 24 Protocol Logic Fault Tolerance: The protocol must ensure that errors in protocol logic and protocol state do not result in unacceptable reduction of integrity and availability.

Can mismatches in protocol logic and state occur during protocol start-up? If such mismatches occur, is there a bound on the time until they are resolved? Does the network protocol logic rely on information from single sources; or is protocol logic dependent on agreeing data from different sources? Does protocol dependence rely on different sources to increase the robustness of algorithms to potential node failures? Are the integrity and availability levels of the protocol logic met by the network?

7.4 LOCAL TRANSMISSION-MONITORING AND SELF-CHECKING SCHEMES.

Network technologies may also implement monitoring or self-checking services to improve fault detection and fault tolerance. As with the guardian action, the effectiveness of such schemes depends on the amount of independence and coverage that can be claimed by the implementation. For example, Controller Area Network incorporates an error-checking mechanism that will switch the network to a passive state if the transmissions of the controller are not suitably acknowledged. Since this is implemented within the same integrated circuit as the communications component, the action may be degraded by common-mode failures. In addition, such schemes may introduce potential fault propagation vulnerabilities, since it is possible for a node to transition to the passive state in response to the erroneous NACK generated from a faulty node. Such vulnerabilities should be analyzed as the network is evaluated.

Other networks may employ local wrap-back schemes where a node monitors its own transmission via local receivers. Such schemes should be analyzed for vulnerability to Byzantine faults, as a local monitoring circuit may perceive the local wrapped-back signals as good, but receivers at the end of a loaded transmission line may see a degraded or erroneous signal.

Criterion 25 Protocol Transmission-Monitoring and Self-Checking Schemes: The protocol must reliably detect transient and permanent faults in both nodes and message transmissions.

Does the network incorporate local health-monitoring schemes to detect node health? Are the network-monitoring schemes suitably independent? Are network-monitoring schemes vulnerable to faulty status sent by erroneous nodes? Does the network technology use local transmission wrap-back? Has transmission wrap-back been analyzed for Byzantine failure vulnerability? Does the network incorporate self-checking pair configurations? Has the coverage and independence of the self-checking configuration been justified?

7.5 RECONFIGURATION AND DEGRADED OPERATION.

Network technologies may also incorporate mechanisms to implement reconfiguration or continued operation in a degraded mode. For example, some physical layers may incorporate a degraded mode of operation that allows communication to continue even if one-half of a differential communications channel is faulted. If such degraded modes are to be leveraged, then the performance (e.g., BER) of the degraded operation needs to be evaluated to ensure that adequate performance is maintained. The protocol mechanism for the detection and announcement of such degraded operation should also be evaluated to verify that timely and correct diagnosis is provided.

Other protocols such as IEEE 1394 may reroute the network path to mitigate physical or node paths. If such protocol action is to be leveraged by a system, then mechanisms used to implement such actions will need to be evaluated to ensure that the reconfiguration time is suitably bounded. As discussed in section 5.5, the issues surrounding the erroneous invocation of such logic must also be considered. The recovery mechanisms for such logic should also be investigated to ensure nodes are not permanently isolated in response to local transient errors.

Criterion 26 Reconfiguration and Degraded Operation: Capabilities provided in the presence of a specified number and type of faults must be sufficient to meet operational requirements.

Can a faulty node cause good nodes to be evicted from system configuration or otherwise cause degradation into a mode that worsens the effects of the fault? Does the network technology provide degraded modes of operation? Is the network performance under degraded mode sufficient to meet network functional requirements? Is network degraded mode operation suitably annunciated to network clients? Does the network perform dynamic reconfiguration of routing to mitigate bad network paths or nodes? Is the reconfiguration time suitably bounded? If online reintegration is supported, what mechanisms are in place to cope with intermittent failures and ensure the health of nodes to be reintegrated?

7.6 LATENT FAILURE DETECTION.

Fault detection, isolation, and recovery functions used within aviation digital electronics systems are often required to be periodically tested to ensure that the detection and recovery actions remain effective. Such detection and recovery functions are usually transparent to normal-mode operations (i.e., have no visible action as long as there are no failures). Hence, without test, it is possible that such functions may fail passively ("stuck at good") and the protection will be lost without indication. Network fault detection and covering functions have the same issue. Therefore, network mechanisms to assist the latent fault detection should be analyzed to ensure that they do not introduce failure vulnerabilities. Interlocks and protection mechanisms to ensure that such testing occurs only in safe system states should also be evaluated. The coverage of the network test procedures should also be evaluated to verify that all key network mechanisms are suitably verified.

Criterion 27 Latent Failure Detection: Latent faults must not accumulate to where they threaten network failure (availability or integrity).

Can an accumulation of latent faults overwhelm a network's ability to tolerate faults? Can a latent fault (e.g., a stuck-at-good fault detector) lead to network failure due to lack of coverage? Does the network technology support a mechanism for latent fault detection, especially in a mismatched protocol state among network clients? If state variables are maintained by the protocol at each client, can multiple unrelated state variables be corrupted before the first corruption is detected? Is the coverage of the network latent fault detection suitable to establish the health of all critical network protection functions? Are suitable test activation interlocks incorporated into the network technology to inadvertent test mode activation?

7.7 VOTING, SELECTION, OR AGREEMENT SERVICES AND REDUNDANCY MANAGEMENT.

Networks may also incorporate redundancy management and voting mechanisms to simplify application-level fault tolerance. Voting algorithms supplied by the network should be compared with the assumption of the application to avoid unsafe operation.

7.8 BYZANTINE FAULT TOLERANCE.

Byzantine fault tolerance is an essential part of any ultradependable system's design. Today, with over 20 years of published papers on the subject, there are still many misconceptions relating to Byzantine failure, both with respect to what makes a system vulnerable and the nature and reality of Byzantine faults. A Byzantine fault is any fault that produces different symptoms for different observers. This can happen at any point where a signal splits; i.e., one source goes to more than one destination. Byzantine faults are a lot like metastability in that there is no way to prevent them; you can only treat the symptoms so the faults do not become system failures.

A Byzantine generals' problem (BGP) is a system failure caused by a Byzantine fault. If the multiple observers do not require any mutual coordination, a BGP cannot occur. But, if the observers have to coordinate in some way or if their actions are compared (by voting or some other means) for fault tolerance, then a BGP is possible. Thus, if a system uses any mutual coordination to achieve its dependability requirements, Byzantine fault tolerance is needed.

Byzantine fault propagation escapes most classical fault containment techniques. Solutions to the BGP are well known, but require a large amount of additional communication bandwidth, a minimum set of redundancy that exceeds what is needed for systems that do not need to tolerate Byzantine faults, and the use of coverage-tested Byzantine filters. Discussions of actual Byzantine failures and methods for coping with them can be found in references 5 and 6. The theoretical basis for Byzantine agreement protocols can be found in reference 7.

Criterion 28 Voting, Selection, or Agreement Services and Redundancy Management: The network protocol must support the ability to determine what nodes are part of the active network quorum.

Does the network support selection and agreement services directly? Does the network provide adequate mechanisms for application software to implement selection and agreement services? Are such agreement services targeted at integrity or availability? Is the overhead for agreement function evaluated and justified to be acceptable? Have temporal effects been evaluated in the selection? Are the time constants of tracking changes in membership fast enough for the application? Are assumptions made by the membership or agreement service justifiable? What vulnerabilities exist while the quorum is inconsistent?

Criterion 29 Fault Model: The protocol must be evaluated with respect to a precisely and completely stated fault model, and the fault model must be compatible with the system architecture.

Does the network claim to be Byzantine fault tolerant? Has the network been analyzed with respect to Byzantine fault tolerance? What are the effects of Byzantine fault tolerance on integrity and availability? If a hybrid fault model is used, is there justification for the relative rates of occurrence of different classes of faults (e.g., Byzantine, strictly omissive, or symmetric) for different failure scenarios. Has the Byzantine fault-tolerant algorithm been suitably analyzed and has coverage been justified (Byzantine fault containment properties)? Has the Byzantine filtering coverage been justified? Has the fault model been analyzed with respect to influences

to all services even those that are possibly not used (this should exclude any effects of services not used)?

8. DESIGN ASSURANCE.

8.1 DEVELOPMENT PROCESSES.

Often network technology forms the backbone of the system architecture. The design correctness of the network implementation is, therefore, of utmost importance as the network provides a significant common-mode failure vulnerability. With the increasing complexity of network technology, the design correctness problem is increasing with every generation of silicon. Therefore, the network technology should be designed with best-practice design procedures. Within the aviation digital electronics domain, this would correspond to DO-178B for software-related network components and DO-254 for hardware components. Network technologies that have formal design assurance artifacts will pose less certification risk than other technologies and may be preferred for that reason. Technologies without formal design assurance processes will need to be considered on a case-by-case basis. The complexity and degree of commercial use of the networking technology will then need to be considered. The commercial off-the-shelf (COTS) provisions within DO-254 were designed to handle COTS hardware technologies that have been used in many systems and have accumulated a huge service experience base. While this experience may be leveraged to assist the design assurance case, certification credit will require substantiation of the service experience data. This treatment is commonly applied to microprocessors. The techniques' applicability to networking-related hardware will need to be considered as the network is evaluated.

Criterion 30 Design Assurance Processes: Have appropriate design assurance processes been followed for design and deployment of the network?

Has the network technology been developed to be in compliance with avionics design assurance guidelines, i.e., DO-178B or DO-254? Does the network technology's commercial use volume support COTS classification, including intellectual property block used within new integrated circuits, as well as manufactured devices? If COTS classification is used, has the COTS product been deployed in similar applications to justify suitable coverage for correctness claims? Is the network technology behavior simple enough to be fully tested? Do diverse approaches (e.g., design diversity) provide adequate and quantifiable coverage if credit is taken for such diversity? Is there a Plan for Hardware Aspects of Certification and a Plan for Software Aspects of Certification for the network infrastructure?

8.2 AVAILABILITY OF STANDARDS AND CONFORMANCE EVIDENCE.

8.2.1 Open Specification and Standardization.

The use of open specifications and standardization might assist a certification authority in establishing the acceptability of a network. Irrespective of the formality of the design artifacts, the quality of the network technology specification is a key attribute. It is preferable if the technology is open with a standardized and published specification, as this will enable the protocol mechanisms to be analyzed and discussed within the academic and industrial

community, including the application for formal verification studies. The standardization process itself is beneficial because the committee activity usually associated with the open standardization process may also lead to an open detailed examination of the network behaviors. However, care is required for network technology that is not designed specifically for use in a safety-relevant environment; the completeness of the specification will need to be carefully reviewed. Often, such standards may specify the normal mode of operation only; the protocol-actions-to-erroneous behavior and the associated degraded modes of operation may not be sufficiently treated in the standard document. The evaluation of the network specification should include such completeness analysis.

Another area where specification completeness may be lacking for COTS protocols is in the area of implementation choices that have been made below the protocol specification. COTS solutions may not be fully described because of the need to maintain competitive advantages between vendors. Hence, many key implementation choices may not be visible and this may impact assurance process where detailed understanding and analysis of the interactions of all technology layers is required. The availability of suitable design information should be considered as the network technology is evaluated.

8.2.2 Conformance and Interoperability Testing.

As with the specification, the availability of standard conformance test campaigns and specifications may also be advantageous. This is especially important for network technology that is sourced from multiple vendors, since it may assist in identifying interoperability glitches. The issues raised in relation to specification completeness also arise in relation to the completeness of the conformance test campaigns: Are all operating modes covered, and are exception and error reactions sufficiently traveled?

8.2.3 Protocol Design Correctness.

In addition to completeness, the correctness of the specification is obviously important. The use of formal methods and development of formal proof arguments for protocol algorithms show much promise, as they can exhaustively verify the algorithmic behavior. However, when reviewing formal verifications, the assumptions that underpin the formal proofs need to be fully understood and evaluated against the real-world failure expectations and behavior. Similarly, the composability of the formal verifications needs to be understood to ensure interactions between different protocol algorithms (for example, membership services and clock synchronization). In some protocols, for example TTP/C, interdependencies exist that may need to be evaluated with formal arguments. That said, formal verification of protocol algorithms can increase design correctness confidence, and therefore, network technology that has such verification evidence may be more attractive.

Criterion 31 Availability of Standards and Conformance Evidence: The technology and protocol of the network should be clearly specified and analyzable.

The use of open specifications and standardization might assist a certification authority in establishing the acceptability of a network. Credit for analysis of specification properties and interoperability between different networks should be supported by conformance and

interoperability tests. The use of formal methods to demonstrate protocol design correctness should be proposed and accepted by the certification authorities. Is the network technology supported by an open specification? Is the specification that is being standardized available to open-industrial committees? Is the network specification complete? Does the network technology address all operational modes of the network, including erroneous node action and associated fault recovery actions? Is the network specification sufficiently detailed to address all required protocol action? Does the networking technology have published conformance test criteria and campaigns? Do the conformance test criteria cover all network protocol behaviors, including fault detection and recovery actions? Have the critical protocol mechanisms and algorithms of the network technology been formally verified? Have the assumptions underpinning the formal verifications been reviewed to ensure that they are consistent with real-world-targeted environment? Are protocol mechanisms and associated formal proofs composable: i.e., Do protocol mechanisms and associated proofs stand by themselves or are they interrelated? Has the network technology been subjected to other validation activities? Did the fault injection campaign include suitably sufficient visibility to observe the key behaviors of all important network mechanisms? Have anomalies and fault observations from such activities been adequately mitigated?

8.3 DESIGN MARGIN.

The issues discussed in sections 3 and 4 also require some design assurance to ensure that adequate design and safety margins are established for the selected network technology. Such a design needs to be established and justified to be valid over the whole system lifetime, addressing parasitic and parametric shifts due to temperature effects, etc. This safety margin evaluation needs to be established in several domains, such as the time and value domains of signals under worst-case design parameters, network loading, etc.

For physical layer attributes, this means that influencing factors need to be analyzed with respect to their margin and contribution to the safety margin. Such physical layer attributes may include oversampling margins that should include the transceiver skew over the whole lifetime of the product, assuming worst-case loading, aging of components (e.g., clock stability over time), temperature range of environment, etc.

Criterion 32 Design Margin: Required design margins should be supported by reviewable evidence.

Has the network design margin been established for worst-case component behaviors? Have all contributions to network design margins been identified?

8.4 CONFIGURATION TABLE CORRECTNESS AND PERFORMANCE JUSTIFICATION.

In addition to the design correctness of the network implementation, the design correctness of network configuration parameters and tables is also required. This is especially important if the table parameterization impacts protocol algorithmic-level behavior, for example, clock synchronization timing and propagation delay parameterization. In such instances, the parameters may severely impact protocol performance. The incorrect configuration of such parameters may, therefore, invalidate any formal proofs of algorithmic correctness. Similarly,

tools may be used to establish parameters for network-policing policies, for example, message transmission rate limiting and maximum message jitter. In such cases, the correctness of these parameters may severely impact network performance assumptions. Therefore, when evaluating a network for suitability, consideration should be given to the rigor applied to ensuring correct network configuration parameters. Ideally, all parameters critical to network operation will have explicit formal requirements and invariants that are traceable to network functional behavior, assumptions, and requirements. Such traceability may assist the completeness checking of the guidance presented. Thus, the guidance supplied will be suitably assured for correctness and completeness.

The network technology may also provide tooling to assist network configuration and its associated verification. Such tools are often required to handle the size and complexity of modern networking technologies and to assist with the generation of nonhuman readable binary configuration tables. If tooling is used for configuration data generation or verification, then the development pedigree of the tooling may also need to be examined as the network technology suitability is evaluated. If the tooling is in-line (i.e., the tooling generates protocol configuration parameters that are not verified by subsequent process checks), then the generation tooling should be qualified in accordance with the DO-178B guidelines for development tools. Alternatively, if the tooling is simply used to verify the network configuration parameters, then they are less stringent and DO-178B verification tool guidance should be adopted. The data flow path of in-line generation and verification tooling needs to be evaluated to ensure that adequate independence exists within the tool chain to prevent a common tooling failure. Ideally, the configuration inspection tools will be driven from reviewed network-related functional data flow requirements and the formal network parameter constraints and invariants as described above.

For some modern asynchronous networks (for example, ARINC 664), the size and scale of the configuration problem is very large, and end-to-end performance (e.g., data flow latency and jitter) is difficult to analyze and bound. The sheer complexity of the network-level interactions between end-nodes behaviors, switch implementations, and the chosen network-policing policies (e.g., message rate limiting) may greatly complicate network performance justification. However, procedures or tooling to analytically bound and justify the worst-case behavior of such networks is required to meet certification requirements. Therefore, the capability and maturity of available analysis tooling should be given careful consideration as these networks are evaluated. Similarly, network technologies that incorporate complicated MAC interactions may also complicate end-to-end performance calculations. Such interactions and any associated network logic (e.g., retry logic and queuing mechanisms) also need to be considered by performance calculations and associated tooling. It is possible that erroneous node behavior can drain network performance until the network has diagnosed the problem and has contained it. Networks that bound the magnitude and duration of these adverse performance influences may be preferable, as they may greatly simplify performance justification calculations.

Highly integrated, multivendor systems' network technologies that incorporate tooling to assist the incremental change of the network tables, which allows new functions and their associated data paths to be added to the network with minimal impact on previously analyzed functions, may also be attractive, since such tooling may ease incremental certification effort.

Criterion 33 Configuration Table Correctness and Performance Justification: Justification for configuration table correctness and performance justification should be provided.

Has the criteria for correct network parameterization and configuration been established? Are the network configuration criteria traceable to network function behavior or top-level requirements or specification? Does tooling assist network configuration and verification? Is the tooling qualified in accordance with the tool guidance established in DO-178B? Have procedures and criteria to bound the worst-case performance of the network been established? Do the worst-case performance criteria address detailed MAC layer interactions? Do the worst-case performance criteria address worst-case fault detection and reconfiguration actions? Does the network technology provide automated tools to assist worst-case performance calculation? Are intermediate-stage buffers and end-node queues adequately sized? Are tools providing performance-bounding qualified in accordance with the tooling guidelines of DO-178B? Does the network technology and associated tooling accommodate incremental change management?

8.5 NETWORK MONITORING AND TEST EQUIPMENT.

With the complexity of modern network technology, the ability to monitor and observe network behavior is very important to support design validation. Similarly, the ability to insert faults into the different network layers may be required to test the network redundancy management mechanisms or the fault response behavior of applications operating on top of the network infrastructure. Therefore, the availability and capability of the test equipment that exists for the network technology may also be a very important consideration. In the ideal situation, such test equipment will be able to observe all behavior of all network nodes, including network start-up and recovery actions. The portability of the test equipment should also be considered, as such equipment is often required to support flight-testing.

The no-interference guarantees of the test equipment may also need to be evaluated if it is to be deployed in a flight test scenario. The ability to monitor the entire network behavior from limited test inspection access points should also be considered. In some modern switched technologies, such access is more difficult than simpler buses. Hence, work in some newer switched technologies is being performed to develop the network-wide controllability and observability needed to test the maintenance of these new or more complicated networks; while at the same time, trying to minimize the invasiveness and logistics complexity of connecting test equipment to these networks.

Criterion 34 Network Monitoring and Test Equipment: Do adequate test equipment, network access points, mechanisms, and procedures exist to ensure that the network is configured correctly and operating correctly (including meeting its specified behavior in the presence of any faults)?

The use of network monitoring and test equipment to establish certification credit should not invalidate the data being observed and should be demonstrated to perform in accordance with the operational requirements for the features being used. Is test equipment available for the network technology? Does the test equipment facilitate the observation of all network operational modes? Does the network test equipment facilitate sufficient fault injection to exercise sufficient network fault detection mechanisms? Does the network test equipment support observation

modes with sufficient noninterference guarantees to support flight-testing? How many network test access points are required to monitor the entire holistic network behavior? (is this feasible for achieving flight test requirements)? Is monitoring and test equipment assured to be noninterfering during operation? Is it guaranteed that the network behavior does not change if monitoring is not performed and correct behavior is inferred from monitoring or testing?

9. SECURITY.

Historically, data communication security has not been an important issue in commercial aviation digital electronics. This began to change with the growing awareness and sensitivity relating to cyber-security in the 1990s. Subsequent terrorist activities accelerated this awareness and sensitivity trend. At the same time, some developments in aviation digital electronics design have made aviation digital electronics systems more vulnerable. Higher levels of integration and more inter-networking connectivity have increased the chances for entrance paths into critical aviation digital electronics functions. Possible gateways onto these paths include, but are not limited to: radio frequency (RF) in the airplane (e.g., portable maintenance access terminals and cabin crew tablets), off-aircraft radio, external gatelink and maintenance ports (optical and RF), and passenger networks. The increasing use of COTS protocols and networking technologies (with their known weaknesses) has the potential of attracting attackers who are familiar with these weaknesses. With ever-increasing bandwidth of modern network technologies, there are also increasing pressures to fully use spare network capacity. Hence, the mixing of critical and noncritical end systems and associated data on common network infrastructure is another increasing trend.

A network should protect itself against security threats (e.g., denial-of-service attacks) and should not allow itself to be used as a means for supporting attacks against its clients. The ability of the network to suitably secure and authenticate private transmissions between different clients on the network should also be evaluated, if such usage scenarios are also anticipated. Network firewall schemes should also be evaluated, especially if critical and open systems share the same network infrastructure. While it may be only "security through obscurity," aviation digital electronics systems are really more secure when COTS is not used.

Criterion 35: Can security weaknesses adversely affect network dependability (safety)?

Does the network technology have security issues that can adversely affect the ability of the network to supply the services needed to support system safety; in particular, is the network susceptible to denial-of-service attacks (e.g., 100BaseTX "killer packets," Ping of Death)? Does the network technology use open COTS protocols that are well known enough to be targets of security threats? Does the network technology require specialized security augmentations; e.g., firewalls?

Criterion 36: If needed, does the network deal adequately with the security issues of privacy (also known as confidentiality or secrecy), integrity, authentication, or authorization?

Does the network technology support sufficient secure services for user and application authentication? Does the network technology support secured data transmission mechanism?

Does the network support multilevel security? How many levels or security domains does the network technology support? Is network configuration data protected and secured during deployment and during load?

10. EVALUATION PROCESS.

As one or more candidate system architectures for aircraft-level functions are proposed, candidate data networks must be evaluated. These evaluations feed PSSA, system and subsystem fault trees, and revisitation of the common cause and zonal analysis, as appropriate. When a final (relatively speaking) design emerges, then a bottom-up FMEA or a FMECA should be performed. Data network technology services may appear in any level of the system design and are required to be analyzed from both the bottom up (FMEA/FMECA) and top down (FTA/PSSA).

To perform the data network evaluations, the guidance of AC 20-156 and the narrative parts of this Handbook should be read and understood. From this understanding and sufficient knowledge of the data network technologies and component implementations, an evaluation can be made that will provide the required data for the above processes. In numerous places throughout this Handbook, there are exhortations to consider the worst case. This does not mean that the worst case should be considered for only the one particular criteria that is stated. All the worst cases of criteria must be considered simultaneously for each criterion, unless there is a demonstrable reason why multiple worst cases cannot occur simultaneously. Evidence must be made available for regulatory review for each criterion that requires supporting evidence.

11. SUMMARY.

This Data Network Evaluation Criteria Handbook builds on previous work. It was created to facilitate the overall certification process for aircraft or aircraft engines that employ digital electronics systems containing data networks by providing evaluation criteria to be used in the development, selection, modification, adaptation, or approval of data network technologies and components to be deployed in safety-critical aviation systems. This Handbook adds to previous work-specific and detailed criteria for evaluating a wide range of data network technologies and components with respect to the possible adverse impacts on certification due to their use. Particular attention was given to issues that are generally overlooked or underappreciated in the industry.

The characteristics of data networks are so varied that it is impossible to create a single set of detailed and specific criteria to which all the criteria are applicable to all data network technologies and components in all possible applications. The combination of extremely wide variation and detail leads to a set of criteria that can be overwhelming. However, for safety-critical systems, the details are very important. Therefore, this Handbook tries to include as much breadth and depth of criteria as possible. To partially mitigate the problem of having an overwhelming set of criteria, this Handbook presents the criteria on two levels. The higher level is presented in the body of the Handbook with much more detailed discussions included in appendix A. This still leaves someone applying these criteria with the task of determining what criteria are applicable to what data network. Because the number of data network technologies and component implementations that can be created is unbounded, making a definitive mapping

to the applicable set of criteria is impossible. Creating such a mapping just for existing data network technologies and component implementations is beyond the capabilities of just one Handbook.

This Handbook is a good companion to AC 20-156. The Advisory Circular provides a means for manufacturers and designers to gain FAA approval of an aviation data network by showing that the data network, as designed, will perform its intended function and satisfies the applicable airworthiness requirements when installed on an aircraft or aircraft engine. It tells manufacturers and designers what they must do, but does not include detailed explanation of how, nor does it provide any warnings about pitfalls that may be encountered by designers who are not data network experts. There is a mismatch in this companionship; the document and criteria structure is different between them. The Handbook was designed to follow the typical protocol stack structure rather than to match AC 20-156. Only time will tell if the structure of either of these documents and their criteria are easier to follow by most readers or if one document should be changed to match the other. It is not recommended to merge these documents or have one subsume the other. They each have a purpose. AC 20-156 was designed to provide assurance goals and issues. This Handbook was designed to provide a detailed technical framework for aid in providing technical data to support the assurance goals and address the issues published in AC 20-156. However, the Handbook was not intended to provide an acceptable means of compliance.

12. REFERENCES.

1. CAST, *Databus Evaluation Criteria*. position paper CAST-16, Certification Authorities Software Team, February 2003. www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-16.doc

2. Lee, Y.H., Rochlin, E., and Scandura, P.A., *Handbook for Ethernet-Based Aviation Databuses: Certification and Design Considerations*, FAA Research Report, 2004.

3. AC 20-156, "Aviation Databus Assurance," August 2006.

4. Driscoll, K., Hall, B., Koopman, P., Ray, J., and DeWalt, M., "Data Network Evaluation Criteria Report," FAA report DOT/FAA/AR-09/27, 2009.

5. Driscoll, K., Hall, B., Paulitsch, M., Zumsteg, P., and Sivencrona, H., "The Real Byzantine Generals," *Proc. of the 23rd Digital Avionics System Conference* (DASC), October 2004.

6. Driscoll, K., Hall, B., Sivencrona, H., and Zumsteg, P., "Byzantine Fault Tolerance, From Theory to Reality," *SAFECOMP 2003*, September 2003.

7. Lamport, L., Shostak, R., and Pease, M., "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, 4(3): 382–401, 1982.

13. GLOSSARY OF TERMS.

Databus and network technologies' terminology can vary considerably. For this reason, a standard set of definitions for all terminology, with respect to aviation digital electronics networks, does not exist. For example, the term "slot" can be used to refer to either a physical address in a cabinet or a transmitting node's temporal position within a table-driven sequence. This glossary is provided to resolve ambiguity and to keep this Handbook consistent.

Anisochronous (also Aperiodic): The essential characteristic of a time scale or signal such that the time intervals between consecutive significant instants do not necessarily have the same duration or durations that are integral multiples of the shortest duration.

Asynchronous: The essential characteristic of time scales or signals such that their corresponding significant instants do not necessarily occur at the same average rate. This term often is misused to mean anisochronous.

Babbler: A node that has babbling transmissions.

Babbling: The act of transmitting a signal not in accordance with a network's protocol. Typically, this means transmitting at times not allowed by the protocol.

Backplane: A card that connects together one or more cards or modules.

Bit-dominant signaling: A bit-dominant signaling method has at least two classes of signals, having the property of dominance. Signals with this dominance property have a priority such that if two or more signals appear on the media the same time, only the (most) dominant signal is perceived by receivers.

Box (also Cabinet or Rack): A mechanical enclosure that contains one or more cards or modules that are typically connected together via a backplane.

Bridge: A client that conveys data through and between two or more networks.

Byzantine Fault: A fault presenting different symptoms to different observers.

Byzantine Failure: The loss of a system service due to a Byzantine fault in systems that require consensus.

Card: A thin rectangular supporting member that electronic components are mounted on. These components could be mounted on one or both sides of the card.

Client: A function that uses one or more services of the network. Note that this is a functional definition; whereas Card, Module, and Box are mechanical definitions; and Node, Device, and Drop are electrical definitions. There may be clients that have a need for, or are only capable of, using a subset of the services provided by the network. Roles of the client are Master, Contender, Slave, Peer, Bridge, or Monitor. A client that is capable of performing one of these

roles, whether or not it is currently performing such a role, is called role-capable. For example, a client that can be a Master (even if it is not the current Master) is called Master-capable.

Criteria (see Evaluation Criteria)

Databus (see Data Network)

Data Network: The communication connection among electronic components. The term bus and databus many times are used in a sense that is synonymous with network. However, the strict definition of bus is a particular network topology. Other topologies include mesh, ring, and star. The term network is preferred to avoid ambiguity. The term bus is used in this document only to denote a particular topology. This is to avoid oxymorons like ring bus. However, bus and databus are used in this document to mean network when referencing other documents that use these terms in the ambiguous sense.

Device (see Node)

Drop: An electrical connection to a network. A box or module may have none, one, or multiple connections to the network.

End Node: A node that is the ultimate producer or consumer of a data network's service (e.g., the transmitter or receiver of a message).

Evaluation Criteria: A characteristic or feature of a data network that may have an impact on system safety. One cannot definitively say that a particular characteristic or feature would have a safety impact for any particular system, because the system's architecture in which the network is used may be insensitive (e.g., not needed) to the particular characteristic or feature, but sensitive architectures would be a problem. For example, a network with a flawed retry mechanism could work just fine in a network that did not do any retries.

Frame: A term that has two distinct definitions in wide use by the aviation digital electronics communication and other communication fields. In aviation digital electronics, the term usually means one repetition of a repeating sequence of scheduled message times. In other communication fields, the term often is used as being synonymous with message or packet. Because of the wide use of both of these definitions, selecting one definition over the other would be foreign to a large number of the intended audience of this Handbook. Compounding this problem is the fact that it is often difficult to distinguish between the two definitions purely by context. Therefore, this Handbook will try to minimize the use of this term and ensure that the correct meaning is obvious whenever it is used.

Guardian: A device placed in the signal flow of a data network that is used to contain failures.

Head-of-Line Blocking: The characteristic of a first-in, first-out (FIFO) buffer that causes priority inversion when the head (next item to be output) of a FIFO queue is blocked from being outputted because it has a low priority, while items behind it have a high enough priority that they could have been outputted from the FIFO if the head item was not blocked.

Host: Client hardware.

In-line Error Detection: Any error detection scheme that does not compare or vote among redundant paths.

Intermediate Stage: A bridge, guardian, or other device through which data network signals must pass.

Intrinsic Safety: A design technique applied to electrical equipment and wiring for hazardous locations. The technique is based on limiting electrical and thermal energy to a level below that required to ignite a specific hazardous atmospheric mixture.

Isochronous: The essential characteristic of a time scale or a signal such that the time intervals between consecutive significant instants either have the same duration or durations that are integral multiples of the shortest duration.

Masquerade Failure: A failure that causes one node to pretend to be another.

Master: A client that has control of the assets (or a subset of the assets) of a network. Generally, there is, at most, one master at any time. However, there may be networks that use an oligarchy, where several masters jointly and concurrently control a set of assets. In the cases where an oligarchy is used, the term Master shall mean every member of the oligarchy that can concurrently affect control of its assets. There are some sophisticated networks that allow a Master to control just a subset of its assets. In this case, there may be multiple Masters, as long as the assets they each control are not also under the control of another Master. These asset subsets may be of different services, e.g., there may be a data transfer Master and an interrupt Master; or the assets of a service may be partitionable, e.g., the individual links of media in a mesh topology.

Master/Shadow: A fault-tolerant scheme in which one redundant device (the Master) is in control until it fails. Upon the Master's failure, another redundant device (the Slave) takes over. This scheme may have multiple Slaves in a priority chain in which a Slave takes over whenever all higher-priority, redundant devices have failed.

Message: One continuous transmission on the network.

Module: A unit of electronics that consists of one or more cards mechanically bound such that they are inserted and removed from a backplane as a single unit.

Monitor: A client that nonintrusively observes the actions of the network without being a Master, Peer, or Slave.

Network (see Data Network)

Node or Device: The electronics connected to a network via a single drop.

Partitioning (see Robust Partitioning)

Peer: A client that has equal authority over the assets of a data network.

Robust Partitioning: A mechanism for assuring the intended isolation of independent aircraft operational functions residing in shared computing resources in all circumstances, including hardware and programming errors. This mechanism was developed for the ARINC 650 family of characteristics. Support for this mechanism is provided particularly by ARINC 653 and 659.

SERDES: A portmanteau for "serializer deserializer." An electronic component that converts parallel data to serial, and serial to parallel. This component usually includes a method for encoding the serial data such that a clock can be reconstructed when the data is converted from serial to parallel. This encoding may also be designed to provide for their desirable features such as direct current balance.

Signal: A variation of a physical quantity used to convey data.

Slave: A client that is responding to the control of a Master.

Slot: A predefined interval of time in which a node (or subset of a system's nodes) has exclusive access to network resources. In minislotting, the minislot interval of time defines when the node may claim access to resources and then the excess is held beyond the end of a minislot time interval.

Source coverage: Fault tolerance mechanisms that contain the effects of a fault to remain within the fault's source or provide means to make all the source's faults easily detectable.

Symbol: A signal state within a defined time interval that is recognized as distinct from other symbols.

Synchronous: The essential characteristic of time scales or signals such that their corresponding significant instants occur at precisely the same average rate. Note: The timing relationship between corresponding significant instants usually varies between specified limits.

APPENDIX A—DATA NETWORK TECHNOLOGY AND ISSUES

A.1 PHYSICAL LAYER.

The lowest level (Layer 1) of most data communication reference model stacks, such as the International Standard Organization Open System Interconnect, Society of Automotive Engineering (SAE) International, or Department of Defense, is the physical layer. The function of the physical layer is to send and receive communication symbols via network media. Layer 1 defines mechanical characteristics (such as connector configuration), characteristics of the media, and characteristics of the signal. The physical layer is responsible for transferring individual bits through the communication media. This level is concerned with the following:

- Connector geometry, gender, and pin assignments
- Physical connections to the media and their characteristics
- Media topology
- Media characteristics (attenuation, delay distortion, impedance, noise, etc.)
- Full-duplex or half-duplex transmission
- Signal speed
- Definition of symbols with respect to signal characteristics (e.g., in amplitude and time)
- Physical service data units; serial bits or multiple bits in parallel
- Handshaking
- Notification of physical fault conditions

The laws of physics impose limits on the frequency and quality of a signal that can be transmitted through a given media, as described by the works of Nyquist [A-1] and Shannon [A-2]. Designers of each data network try to create a physical layer that maximizes data rate and quality for a given cost.

Given the physics and cost constraints, some compromises and trade-offs must be made. Users of data networks in safety-critical applications must be aware of how these design choices for the physical layer can impact system safety via the quality of data transmission provided by the data network.

A.1.1 ENVIRONMENT.

Data network components must meet the requirements of an aviation digital electronics environment, as described in RTCA DO-160. This means that the data network components not only must survive this environment but must also simultaneously satisfy all requirements placed on the data network while residing in this environment.

A.1.2 PROBABILITY OF BIT ERRORS.

Because the physical layer is the foundation upon which all other protocol layers depend, any failure in this layer will adversely affect all the layers above it unless adequately mitigated. An obvious question that must be answered is: What is the probability of failure in the physical layer?

Physical layer specifications often state a bit error rate (BER), which gives the probability of error for each bit. This number is typically in the 10^{-6} to 10^{-15} range. The purported source of these errors is the signal-to-noise ratio (SNR). Formulas relating SNR to BER have a form similar to:

$$BER = \tfrac{1}{2}(1-erf)(Eb/No)^{1/2}$$

where *erf* is the error function, *Eb* is the energy in one bit, and *No* is the noise power spectral density (noise power in a 1-Hz bandwidth). The ratio *Eb/No* is a form of SNR. The energy per bit, *Eb*, can be determined by dividing the carrier power by the bit rate. As an energy measure, *Eb* is measured in joules. *No* is in power (joules per second) per Hz (seconds), so *Eb/No* is a dimensionless term, or simply, a numerical ratio.

It is important to note that the exact formulas for BER depend on the modulation and encoding schemes used, because these schemes, coupled with the physical properties of the media, are important for establishing the so-called "eye pattern." This pattern encloses the space bounded by the minimum upper value, maximum lower value, and the minimum spacing between transitions of a signal. Figure A-1 shows a typical eye pattern created by the superposition of many symbols and the effects of additional signal noise. The two dashed horizontal lines in the figure represent the minimum and maximum value of the receiver's input threshold that the receiver uses to determine whether an incoming signal is high or low. The two vertical dashed lines represent the variation in the time that the receiver samples the input. A receiver's decision about the data value of an incoming signal takes place within the area enclosed by these dashed lines, which is highlighted by the gray box in the figure. The distance between this box and the incoming signal's eye pattern determines the noise margin of the receiver. It is clear that the smaller the area enclosed by the eye pattern, the higher the probability that an error will occur. The size of the eye pattern is determined by the modulation and encoding schemes plus signal noise. Thus, claims of a specific BER without reference to the modulation and encoding schemes and assumed noise amplitudes are worthless for predicting the probability of bit errors. When establishing the actual value for BER, the test patterns used in an evaluation must be those that are actually used by the network, not just the linear feedback shift register-generated pseudo-random bit sequences used by most BER test equipment. The BER test must also be run in the same noise environment as the actual system will experience.

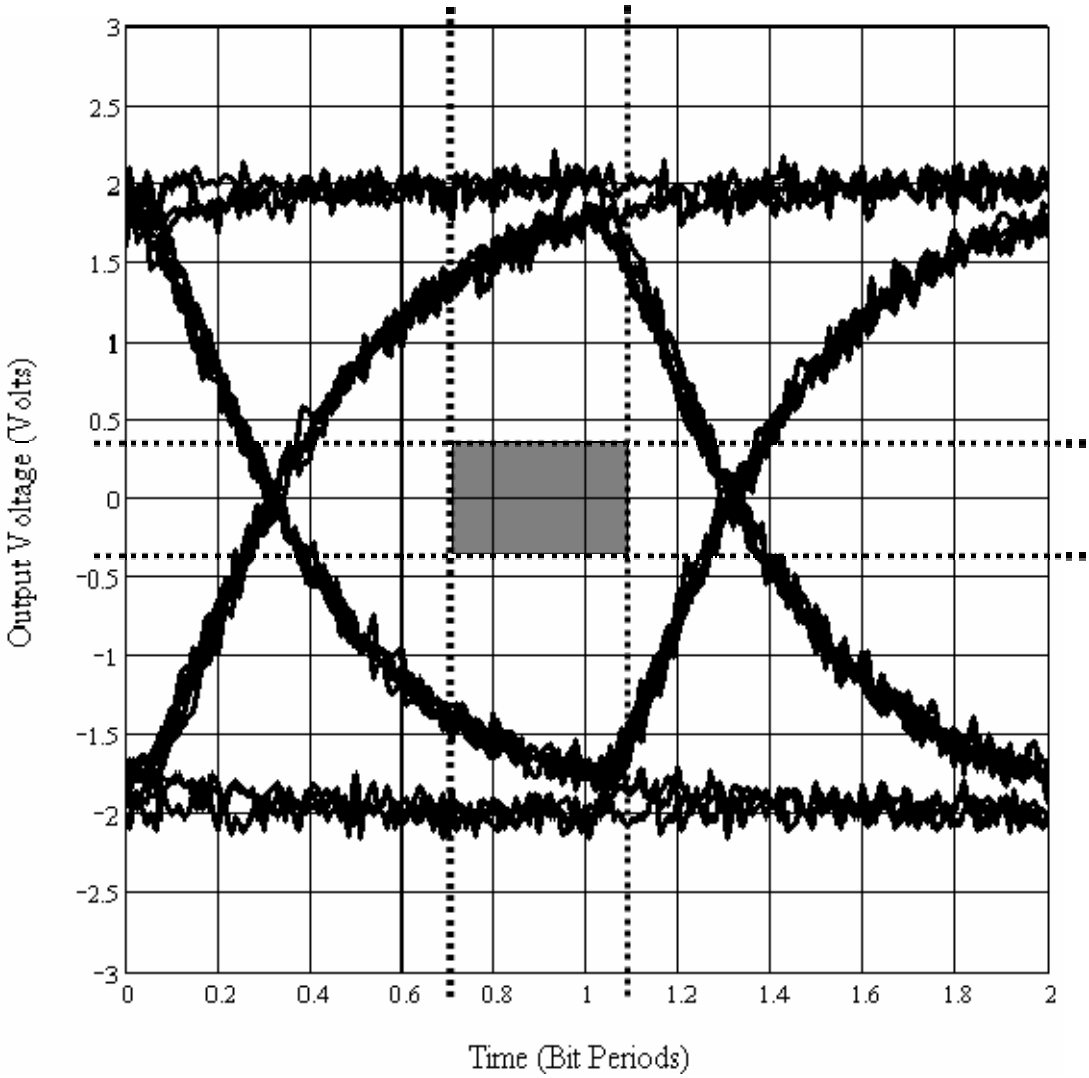

Figure A-1. Eye Pattern

The designs of data network error handling and data network reliability analyses are often based on the misconception that bits traversing a data network are independent of other bits on the same network medium. However, designers of data network physical layers are familiar with a phenomenon called intersymbol interference (ISI). The definition of ISI from the Federal Standard 1037C [A-3] is:

> "**1**. In a digital transmission system, distortion of the received signal, which distortion is manifested in the temporal spreading and consequent overlap of individual pulses to the degree that the receiver cannot reliably distinguish between changes of state, *i.e.*, between individual signal elements. *Note 1*: At a certain threshold, intersymbol interference will compromise the integrity of the received data. *Note 2*: Intersymbol interference attributable to the statistical nature of quantum mechanisms sets the fundamental limit to receiver sensitivity. *Note 3*: Intersymbol interference may be measured by eye patterns.

2. Extraneous energy from the signal in one or more keying intervals that interferes with the reception of the signal in another keying interval.

3. The disturbance caused by extraneous energy from the signal in one or more keying intervals that interferes with the reception of the signal in another keying interval."

ISI can be caused by timing jitter from adjacent bits, "baseline wander," or reflections due to impedance mismatches.

It is clear that the transitions that form the sides of an eye pattern are affected by the adjacent bits. Bits much further away also can impact an eye pattern via baseline wander caused by accumulated effects of direct current (dc) imbalance that charge the capacitors and inductors in the communication path. These capacitors and inductors include the components that a signal must go through (such as transformers) and the intrinsic characteristics of the media and any other components that touch the media (e.g., each receiver and each transmitter adds parasitic capacitance to the media). This baseline wander raises or lowers the eye pattern for every bit, shifting it with respect to the receivers' input threshold. This shift affects the probability that the receiver sees an input as one value or another as long as the baseline has wandered away from nominal. During its development, opponents of the 100BaseTX Ethernet design touted the fact that there existed "killer packets." These are packets containing particular data patterns that produce baseline wander bad enough to induce bit errors on their own. Most Ethernet PHY chips have active compensation for some amount of baseline wander. When using 100BaseTX, care should be taken to provide a means to prevent killer packets from appearing on the network or to use only those PHY devices that can compensate for killer packet levels of baseline wander.

Another adverse effect that can cause correlated bit errors is reflections due to impedance mismatches. Impedance mismatches can occur whenever the media or the electrical properties surrounding it changes. This happens whenever the media is split (e.g., for stubs or drops), when the signal passes through a connector, at receivers and transmitters, or by having inadequately shielded media pass near materials of different electrical characteristics. Note that these impedance concerns are true for both electrical and optical data communication.

Another phenomena with some characteristics similar to that of impedance mismatch reflections is the problem of the "Wired-Or" glitch (which can be viewed as "Wired-And" with the application of DeMorgan's theorem). Data networks that are susceptible to this problem are those that exploit a bit-dominant signaling method on the media to perform some logic function. One such function is bit-dominant bitwise priority arbitration. Examples of networks that use bit-dominant bitwise priority arbitration include the Controller Area Network (CAN), which is commonly used in automotive applications, and the SAE AS4710 PI-bus, which is the military avionics standard backplane bus. The Wired-Or glitch occurs when two or more transmitters drive a dominant signal onto the medium and a proper subset of these transmitters stops driving the dominant signal. When this happens, the medium state near these transmitters changes to the recessive state for a time equal to the round-trip delay between them and the nearest transmitter(s) still driving the medium to the dominant state. To keep a receiver from erroneously interpreting these glitches as valid changes of signal state, the receivers must be designed to tolerate these glitches, or design rules must be followed that limit the duration of the

glitches (usually by limiting the length of the medium as a function of the bit width) to a duration that can be tolerated.

BER tests must be done using the worst-case modulation and encoding scheme symbol patterns, the worst-case signal path (including the effects of all inductors and capacitors), and the worst-case reflections due to impedance mismatch. Design, installation, and repair rules must be established so that situations worse than those used in this test do not occur.

BER is only the beginning, the lower bound, of the probability for erroneous bits. There are a number of other sources that can cause erroneous bits. These sources include: external noise sources (including crosstalk), clock jitter and drift within transmitters and receivers, metastability within receivers, and other hardware failures.

Trying to tolerate failures caused by external noise sources is difficult because the external noise sources (such as crosstalk, lightning, and high-intensity radio frequency (HIRF)) can cause arbitrary error patterns with unknown probabilities. The best, and most widely used, way of dealing with external noise sources is to try to make the bits immune to upset. This immunity can be produced by shielding the media from external noise sources, making the signaling scheme robust (e.g., differential drivers with large margins), and adding components to filter out noise. A number of recent developments have eroded these protections. Composite skin aircraft provide less protection against noise sources outside the aircraft. Newer, higher-speed data networks use signaling levels with smaller margins. The wider bandwidths of the higher-speed data networks make it more difficult to filter out noise. One way to counter the erosion caused by the last two trends is to minimize the speed and bandwidth required to meet the system's data throughput needs; that is, to use a network that is as efficient as possible. No matter how much effort is put into trying to make a data network immune to external noise, there can always be some noise source with a large enough magnitude to overcome these efforts. When a data network is overwhelmed by large-amplitude external noise, it is important for the network to recover as soon as possible.

Some designers are now suggesting the use of wireless data networks within an aircraft. There have even been suggestions that these wireless data networks be used for safety-critical functions. However, this appears to be a daunting design challenge given that the external noise sources (such as lightning and HIRF) can cause arbitrary error patterns with unknown probabilities.

It is hard to reconcile an existing electromagnetic-interference requirement for a wired data network to survive 200 volts per meter of noise versus a wireless receiver's input being almost a million times more sensitive.

A.1.3 PROBABILITY OF ELECTRICAL COMPONENT FAILURES.

The avionics industry has a long history of evaluating the dependability of a system, at least for benign faults (i.e., faults that are inherently self-containing or can obviously be detected and contained). However, the fact that complex integrated circuits can have arbitrarily bad behavior is too often ignored. Even extremely simple analog devices can have surprising failure modes. For example, a simple Military Standard (MIL-STD)-1553 databus transmitter was observed that

produced a perfect Manchester waveform output when the component had no signal input. A similar problem has been observed with a fault-free RS-485 driver transmitting a rectangular waveform when its inputs were "stuck high." When applying the evaluation criteria described in this report, one must remember that electronic circuitry can fail in a way that produces arbitrarily bad behavior, limited only by the energy provided to it (which can be considerable when stored, e.g., capacitors).

With the advent of higher-speed networks, smaller impairments to a signal can cause problems. This creates a concern for the quality (including aging effects) of connectors, media, and drivers.

As data network speeds increase, not only does the SNR decrease on the network media, it also decreases within the electronics. This reduction in SNR makes electronics more susceptible to single-event upset (SEU) and metastability. The evaluation of SEU susceptibility should be done as part of the environmental evaluation.

Metastability is an electronic circuit design issue rather than an environmental issue. As clock speeds increase to create higher-performance electronics, the amount of circuitry that can be driven by a single clock zone decreases. This creates more clock zones and the need for a larger number of synchronizers at the boundaries between the different clock zones. Each synchronizer has some probability of metastability failure. The metastability failure rate for a synchronizer is given by the formula

$$\alpha * f_{data} * f_{clock} * e^{-\beta t}$$

where α and β are constants unique to each synchronizer implementation, f_{data} is the frequency of the data, f_{clock} is the frequency of the clock, and t is the time that the synchronizer waits for its first-stage flip-flop to settle to a valid value. As data network speed increases, f_{data} and f_{clock} tend to increase proportionately and t is the inverse of f_{clock} (in the design of most synchronizers, t is one *clock* period). This means that synchronizer transient failure rates increase with system speed (S) proportional to S^2e^S. This already very steep function is exacerbated by the fact that higher clock speeds require more synchronizers. Luckily, the very characteristics that allow increased speed also tend to improve the values of α and β. However, the only way to determine accurate values of α and β is to test for them.

Synchronizer metastability error rate is tested by giving the synchronizer data that is asynchronous (and statistically independent) to the synchronizer's clock while sweeping the value of t. The resulting number of errors for each t is fitted to a semi-log line versus t. The intercept of this line is α and the slope is β. It is important to note that this test must be done on an actual implementation of each synchronizer design. A very widely held misconception is that α and β depend only on the design of the first-stage flip-flop in the synchronizer. While the characteristics of this flip-flop have a large influence on α and β, it is not the only influence. As electronic component geometries shrink, the electrical characteristics of interconnect actually become more important than that of transistors. This increases the importance of characterizing the interconnect between the first-stage flip-flop and the second-stage flip-flop in each

synchronizer design. Electrical characteristics of the second-stage flip-flop (such as input thresholds) are also important. The difference in results from testing the actual synchronizer design versus testing just the first-stage flip-flop may not be significant for applications that do not have stringent safety requirements. But, for systems that have dependability requirements in the neighborhood of 10^{-9}, this difference could consume the entire dependability budget. Thus, for a synchronizer metastability error rate test to be valid, the test must be performed on the actual synchronizer design.

One factor that affects BER is jitter on the input-sampling clock in the receivers. Higher data rates are more sensitive to this jitter. Often, the receivers use phase-locked loops (PLL) to create these clocks. The PLLs are driven from an external clock source. The tighter jitter requirements for higher-speed data networks often require higher-quality external clocks that drive the PLLs. These clock quality requirements often include restrictions on short-term and long-term jitter. The same clock quality issues affect transmitters. In this case, jitter on a transmitter clock causes jitter in the data. The sum of the transmitted data jitter and received clock jitter affect the error rate of the received data.

A design factor related to the input-sampling clock jitter in receivers is the ratio of the clock's period to the smallest interval between input signal transitions. This has a large impact on the gray box in figure A-1. A higher-frequency sample clock simultaneously makes the box smaller horizontally and allows the box to be placed more precisely in the center of the eye. A sample clock that is too slow can place the box too close to the edge of the eye pattern. This can be a source of bit errors in the receiver and be a source of asymmetric or so-called Byzantine faults.

Jitter and frequency offsets between a transmitter's clock and a receiver's clock also can cause buffer overruns and underruns in elasticity buffers and can be the source of asymmetric and Byzantine faults.

A.1.4 ELECTRICAL ISOLATION PROPERTIES.

The causes of total system failure can be segregated into three main classes: exhaustion of redundancy, single point of failure, and lack of fault containment. Of these, the one that is most often seen as part of real-world total system failures is the lack of fault containment. One important aspect of fault containment is the electrical isolation between redundancies. In examining a system design for possible electrical fault propagations, one can use the following mental process, which imagines that:

- Each redundant power supply is painted a unique color
- Each electron leaving a power supply is painted the same color as that supply
- Each component or conductor that an electron enters is painted that electron's color
- If there is a color conflict, a possible galvanic fault propagation path has been found

To prevent the data network from becoming a galvanic fault propagation path, these paths are usually interrupted with attenuators and resistors, fiber optic cables, optical isolators, or transformers. Some of these isolation methods impose requirements on the physical layer signaling. For example, transformers needed dc-balanced signaling, such as Manchester or

8b/10b. Some of these isolation methods may preclude the use of collision detection or the use of mixing dominant and recessive signals on the media to perform some logic function.

Some networks, such as universal serial bus, power over Ethernet, and Institute of Electrical and Electronic Engineers (IEEE 1394), transmit power on some of the conductors in their cables. Requiring use of this power creates a significant problem for galvanic isolation.

Many fault-tolerant architectures include the concept of receive-only nodes. The required characteristic of these nodes is that they can receive information that is transferred across the media, but are prevented from having any effect on any shared media. In these architectures, it is essential to provide assurances that these receive-only nodes cannot affect the data network.

A.1.5 PHYSICAL COMPOSABILITY.

As the size of a network grows in the number of nodes, number of links, number of taps/splices/wyes, link distances, etc., performance or signal quality can suffer. Some of the physical layer characteristics that can be adversely affected include a decrease in signal margins, added latency or propagation delays, an increase in reflections due to impedance mismatches, and an increase in the probability of reflections constructively adding together to create higher-amplitude problems. A well-designed data network anticipates the effects of network growth and can work correctly with any size network up to explicitly stated limits. The description of a data network may include design rules that must be followed for the data network to maintain sufficient physical layer quality margin as the size of the network changes. These rules can include such things as topology restrictions (e.g., nodes on a bus cannot be connected any closer than a certain interval), limitations of signaling speed versus distance, or setting certain parameters within the data network's components that affect its performance (e.g., setting intermessage gap sizes or contention resolution times based on the maximum round-trip delay over a given topology installation).

A.2 DATA LINK LAYER.

The data link layer is the layer immediately above the physical layer in most data communication reference model stacks. It provides the functions, procedures, and protocols needed to establish, maintain, and release data link connections between the nodes of a network. A conceptual level of data processing or control logic in the hierarchical structure of a node is responsible for maintaining control of the data link. The data link layer's functions include bit injection into the transmitter and bit extraction at the receiver; address and control field interpretation; command and response generation, transmission, and interpretation; synchronization; error control; and flow control.

The data link layer is divided into two sublayers: the media access control (MAC) sublayer and the logical link control (LLC) sublayer. The MAC sublayer controls how a node on the network gains permission to transmit on it. MAC sublayer protocols often try to provide prioritization or fairness in granting access to the media. MAC sublayer protocols also try to maximize the use of the media and minimize the probability of starvation (not granting access to requesters). The LLC sublayer controls frame synchronization, flow control, and error checking. Conceptually, the LLC sublayer sits on top of the MAC sublayer.

A.2.1 The MAC.

The MAC sublayer is a particularly important part of a data network's protocol when the network is used for real-time systems. Simple problems in the MAC sublayer can cause catastrophic loss of the services that the real-time system needs from the data network. These problems include no access (starvation), not enough access, or wrong time access. One source of these problems is the design of the protocol itself, coupled with access demands and timing of clients, including faulty clients that fail to follow the behaviors expected or required by the data network specification.

Other problems can be caused by failures (including permanent and transient failures) in the hardware that directly controls or accesses the network media. These failures may be introduced by any of the sources described in section A.1. Of particular concern is the possible brittleness (lack of robustness) of the MAC sublayer protocol. That is, does the MAC sublayer protocol amplify the effect of small failures and errors such that they become large problems? For example, does the MAC sublayer protocol allow transient failures and errors to have an effect that persists longer than current transmissions?

The following six sections describe problems unique to each type of MAC.

A.2.1.1 Master and Slave.

The simplest MAC mechanism is to designate a single node as the master controller. This single node will have sole authority to grant access to data network's media. The most common example of this kind of MAC in avionics is the MIL-STD-1553. A centralized media access controller has several weaknesses; the most obvious is that it is a single point of failure. That is, if the controller fails to function or functions incorrectly, the entire communication system will fail. This problem can be mitigated by adding fault tolerance, either within the controller or by having multiple controllers. However, designing such fault tolerance is difficult, and no known data networks that are now used or proposed for aviation digital electronics employ such a scheme.

A.2.1.2 Bit-Dominant Arbitration.

Bit-dominant bitwise arbitration, sometimes called the Lanning protocol, is a very old MAC mechanism. It was used in telegraphy about a century ago. This mechanism uses two (or more) classes of signal that have a dominance characteristic such that if more than one signal appears on the media simultaneously, only the (most) dominant signal is perceived by receivers. Each message begins with a sequence of bits representing the message's priority, most significant bit first. As each bit is transmitted, each transmitting node checks the value on the media. If the value transmitted by a node is recessive, but the value on the media is dominant, the node recognizes that it has a lower priority than some other node that is currently transmitting. As soon as a node recognizes that it has lower priority, it stops transmitting its message. The lower-priority node(s) may again try to transmit after the current message transmission completes.

This arbitration method has a number of physical layer issues. Another issue is the constraint that each bit must have a duration that is longer than the worst-case, round-trip delay on the

media. To this constraint, one must add the effects of local clock jitter, sampling granularity error, and the signal jitter caused by the dc component of these relatively large bits.

This type of arbitration has no fairness. It is possible for one node to use all the network bandwidth and cause starvation in all other nodes. The system designer must add fairness on top of these protocols.

A.2.1.3 Carrier Sense Multiple Access/Collision Detect.

Carrier sense multiple access/collision detect is the MAC used for IEEE 802.3 (Ethernet). A node that wants to transmit first, listens to the media. If the media is busy, the node waits. If the media is not busy, the node attempts to transmit. If more than one node tries to transmit at the same time, a collision is detected. When a collision is detected, the transmitting nodes stop transmitting and try again later.

Well-known problems with this arbitration scheme include the following.

- It is nondeterministic, e.g., miniscule changes in timing can cause changes in message order, and there is the small (but unknown) probability that collisions among transmitters can recur until they abort and then recur again so no messages ever get delivered).
- It has no fairness guarantees.
- It turns simple deaf nodes into babblers.
- How is the message schedule determined or agreed upon?
- How are system clocks synchronized to ensure that all nodes have the correct notion of system time?
- Corrupted tokens—which means the next node that should have gained access to the media will not know that it should have done so and traffic will cease.
- Swallowed tokens—where the current node holding the token dies before it can send the token on, again traffic will cease.
- Counterfeit tokens—to solve the above problems, new tokens have to be minted, failures in this mechanism can cause duplicate tokens.

A.2.1.4 Time Division Multiple Access.

Time division multiple access (TDMA) and its variants use a preagreed order of transmission for the size of the message. These types of MAC require some form of clock synchronization.

A.2.1.5 Token Passing.

In a token-passing MAC, the node that currently has access to the media must hold a token. For another node to gain access to the media, the current node must pass this token on to the other node.

A.2.1.6 Minislotting.

A node using a minislotting MAC measures time from the end of each transmission. A node is allowed to transmit if the time it measures exceeds a threshold unique to that node and no other node has started to transmit. ARINC 629 uses a variation of this MAC. One problem with basic minislotting is that it has no fairness. ARINC 629 attempts to solve this problem by adding another timer, which blocks a node from transmitting more than once in a period that is long enough to allow other nodes fair access to the media. However, this scheme does not prevent a node (or multiple nodes) from transmitting for a length of time that will starve other nodes.

A.2.2 MAC Replacements.

Many dependable, real-time data network systems use the hardware from an existing data communication network that has an inadequate MAC and then apply a substitute MAC on top of the existing hardware. This effectively removes the MAC and turns the existing data communication network node hardware into something that is little more than a simple serializer/deserializer (SERDES) that just converts parallel data to serial data and back again, but requires much more hardware (e.g., in the form of gate count) than would be needed to build just a SERDES. Many such networks are based on IEEE 802.3 (Ethernet). The system designer must consider whether the excess hardware can cause problems under unintended circumstances.

A.2.3 LINE-LEVEL ENCODING.

Line-level encoding is the way that logical data is physically represented on a network.

BER is heavily influenced by the eye pattern that is created by a network's line-level encoding scheme. In addition to affecting the data network's own signal quality, line-level encoding can also affect other equipment via radiated emissions. Thus, it is important to determine whether the spectrum radiated from the line-level encoding has components in frequencies that can adversely affect other equipment.

A.2.4 MESSAGE FORMATING (FRAMING).

The message formatting or framing part of the LLC sublayer handles groups of bits sent over a link as discrete units. A message (also known as the frame or a packet) may contain control and addressing information, as well as error detection, for example, cyclic redundancy code (CRC) information or forward error correction information. The size and composition of the frame varies according to the protocol. Depending on the protocol, components of a message may include preamble, start delimiter, source address, destination address, routing information, length field, flow control information, MAC information, error detection or correction information, or end delimiter.

In evaluating the dependability of a message format, one must examine the consequences of any part of that format having an error.

Preambles need to be of sufficient size to restore dc levels to the nominal value, facilitate synchronization of the bit-sampling clock to the incoming data stream, etc. Because dc levels may not have nominal values during the receipt of a preamble, there is a good probability that receiving nodes will see errors in the preamble. Some poor receiver designs assume these errors will always be at the beginning of the preamble and, thus, only tolerate errors there. A more robust design would tolerate any number of failures in the preamble except for errors that make the preamble look like the next part of the message, typically a start delimiter. This is possible because preambles typically are highly redundant with no unique information residing in any one bit.

Are there parts of a message where an error could cause the loss of more than one message? This question includes not only bit errors that occur while the message transits drivers, media, and receivers, but also erroneous values that may be created by the source node or intermediate stages. An example is the corruption or counterfeiting of a token bit pattern in a token-passing MAC.

Other than redundancy bits (e.g., error detection or correction fields), is the message format efficient? Note that inefficiency leads to more bits, which leads to a greater possibility of an error. Related to this concept is the observation that some information that is transmitted in a message in one protocol (where it is vulnerable to errors) may not be transmitted in another protocol. For example, there are table-driven protocols in which all addressing, length information, etc., are held in a memory protected from errors rather than being transmitted on the network. There also are protocols that use redundant signal lines for error detection and correction (EDAC) instead of adding check bits to the message. Combining these two ideas, one could have a data network (such as SAFEbus) where messages have absolutely no overhead; every message bit is a data bit.

A.2.5 ERROR DETECTION.

Network criteria that have significant influence on the overall safety are the error detection capabilities of the link layer, because efficient error detection directly affects the integrity of the data. The key to the underlying effectiveness of the error detection mechanism is the assumed failure model of links. It is often assumed that link failures are primarily bit flips, and the vulnerability of the link layer error detection mechanisms to undetected errors in data is evaluated in the context of BER. Yet, the BER effectiveness evaluation is only one criterion to be evaluated. Error detection criteria should stretch to include effects, such as wire crosstalk, and correlated errors, such as HIRF events, unless mitigated with other means (such as shielding).

This section discusses error detection of the link layer. Link layer errors can occur in the communication media, in its drivers and receivers, or in intermediate nodes (such as repeaters). Section 5 addresses some error detection mechanisms that may reside in the equipment at the ends of the network or at intermediate stages within the network.

A.2.5.1 Protocol Violation Error Detection.

Detection of errors on the link layer should include a strength evaluation of a network protocol state machine to detect errors that are semantically incorrect. For example, message format fields may exhaustively use all combinations of possible values. Implementation of the protocol and protocol state machine should be able to detect violations caused by values that are not valid. Otherwise, erroneous messages may be interpreted in a nonintended way, resulting in safety implications.

A.2.5.2 Parity and Frame Check Sequences.

Parity and frame check sequence evaluation criteria that do not include the adequate description and validation of error pattern may result in the use of mechanisms that do not have adequate error detection capabilities. Typically, the validated BER can be used for adequate error detection coverage assessment. There are many different error detection mechanisms and encodings, such as CRC, Fletcher, Adler, AND, XOR, etc., with different characteristics; however, this document only focuses on the characteristics of a few representative mechanisms.

CRCs are commonly used to detect link errors. In general, CRCs have been found to be extremely strong in detection of bit flips. Yet, the number of bit flips a certain CRC polynomial can detect depends on the length of the covered data and the generator polynomial used in the check-data computation. The metric most commonly used for determining the quality of error detection is hamming distance (HD), i.e., the minimum number of independent bit flips that can result in an undetected error. Given the HD and BER for the medium, the designer can compute the probability of an undetected error. This probability should be sufficiently small for the reliability requirements of the data network.

Another error detection metric is the ability to detect error bursts. An error burst of a particular length, n, is defined as sequence of n bits, the first and last of which are erroneous. The CRC can detect all error bursts of length k (where k is the degree of the generator polynomial) or smaller. While CRCs can also detect some error bursts longer than k bits, some error patterns are guaranteed to be undetectable, so the CRC should not be relied upon to detect error bursts greater than k bits in length.

High error rates and correlated error probabilities may especially be encountered in wireless networks, where there is basically no shielding from external effects. The worst-case analysis may become a real challenge for such networks in the aviation digital electronics domain. Architectural means, such as voting of triple-redundant data channels as mitigation to in-line error detection techniques, can only detect errors on the link if the channels are truly independent. Wireless network connections may be extremely vulnerable to common-mode effects on different channels due to unavailability of shielding protection.

In-line error detection may not only affect integrity (namely the probability of undetected errors), but also availability. While BER can be a useful figure to describe environmental effects and the integrity mechanisms can be very effective in detecting errors, the detection of an error again has implications on the availability of data at the end node. Any detected erroneous message cannot

be used by the application, resulting in decreased availability. The longer a message gets, the more likely a message may not be available due to an error. Unavailability of data can have similar safety effects as incorrect data.

A.2.5.3 Interactions Between Line-Level Encoding and Error Detection.

When assessing the system safety, one must not overlook the potential impact of line encoding on the error detection capabilities. Such interactions should be examined for the worst case. As the CRC (or similar in-line, error-encoding mechanism) is computed over the data, which is then transformed to a representation that is sent over the physical layer, the properties of the error detection change. Properties change because the encoder and decoder transform the representation. As a consequence, a single bit flip can result in multiple bit flips for the data at the link layer where the CRC is calculated. Similarly, the perceived error burst length that a CRC can tolerate may be shorter than expected due to the encoding. Figure A-2 depicts a scenario of data with a frame check sequence (FCS) that is encoded using 8b/10b as transmission format. The actual error burst is smaller than the maximum error burst tolerated by the CRC. Yet, due to the decoding of the physical data, the perceived error burst as seen at the receiver is longer than the tolerated value. If not considered, such interactions between encoding and error detection can invalidate error detection analysis.

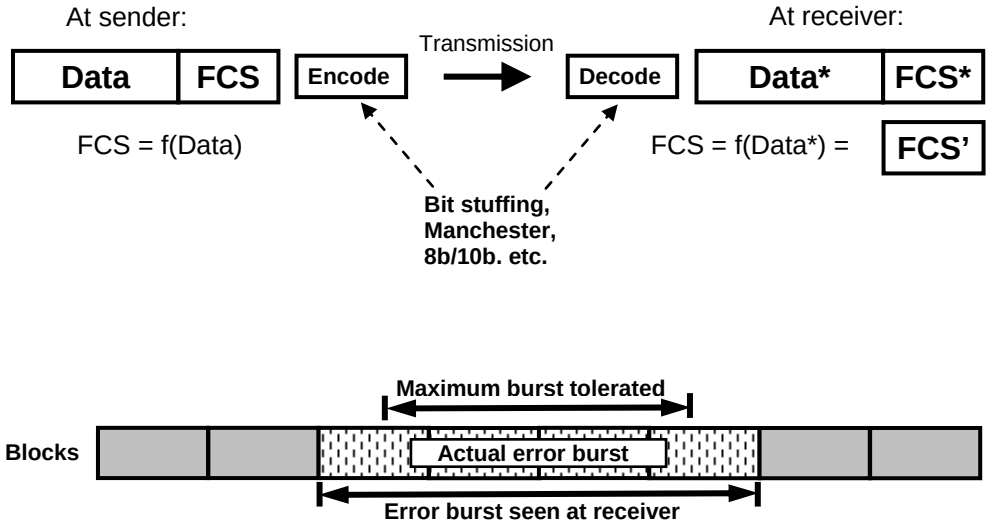

Figure A-2. Error Burst Length Extension Due to Encoding

A similar effect between the coding and physical layers is the multibit error vulnerability of protocols that employ bit stuffing to guarantee a minimum number of transitions at the line level. To properly encode and decode the bit-stuffed data, the entire message, including any CRC or other error detection field, must be bit-stuffed. However, a small number of bit-flips in a bit-stuffed message can result in a cascade error where data bits are interpreted as stuff bits and vice versa. In this case, the number of actual bit flips (as few as two can cause the cascade error) can result in a much larger number of bit flips in the decoded message that can exceed the error detecting capabilities of the CRC

When evaluating the overall error detection coverage, one must include the error detection probability of the FCS, the error detection probability (if any) available from the coding scheme, and the possible interactions between the two.

Active intermediate stages (such as network relay stations containing active logic, i.e., silicon devices) may defeat any in-line error detection mechanisms (such as CRCs), because the potential failure mode of such devices may be arbitrary. The assumption of a uniform error model may not hold for such scenarios, because silicon failures may transform in-line error detection codes in a way that the frame check sequence is unable to signal an error in the worst case.

A.3 NETWORK LAYER, TRANSPORT LAYER, AND NETWORK MANAGEMENT.

In the OSI model, the network layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, inter-networking, error handling, congestion control, and packet sequencing. Above the network layer, the transport layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer. In embedded systems, the functionality of these layers is often merged into a single layer of functionality. This section discusses the issues related to the functions of both layers together. In addition, in some newer protocols, (for example, Time Trigger Protocol/SAE Class C (TTP/C) and FlexRay) a network management layer is emerging to describe hardware or software services that facilitate message agreement, network diagnosis, and synchronization. Issues relating to these issues are also discussed.

A.3.1 NETWORK VULNERABILITY TO ADDRESSING INFORMATION FAILURE.

If the network technology encompasses message-labeling or node-addressing identification information, then the failure modes of the addressing or labeling mechanisms need to be evaluated; as such, mechanisms may be vulnerable to component failures or transport corruption. An example failure mode is the masquerade failure, where one network node can impersonate another node of the system. Failures of addressing or message-labeling information is especially important in integrated modular aviation digital electronics systems comprising numerous aircraft functions, because failure of these mechanism can lead to unbounded data flow failures, which makes functional failure isolation almost impossible at the application layer.

The data network vulnerabilities to technology shortcomings may differ, depending on the network implementation. If the network packet format includes addressing or other information that indicates message content (e.g., a message or label identification), then the network is obviously vulnerable to corruptions of these fields during transmission. For the network to be dependable, there must be mechanisms to handle any such corruptions. These mechanisms must be evaluated to establish their coverage (their ability to handle these corruptions). Note that network integrity mechanisms (e.g., frame-check sequence) may detect transmission errors; however, these mechanisms have limited coverage, as described in section 4.4 of reference A-4. Any fault-handling mechanism must have provable coverage against all the possible failure modes of the communication channel. Message routing and blocking enforcement of

intermediate stages (e.g., guardians, etc.) based on addresses or labels need also to be evaluated to ensure they provide adequate coverage, as described in section 7.2 of reference A-4.

The influence of software on network-addressing information is also an issue, as discussed in section 5.8 of reference A-4. Such software-directed access may leave a network vulnerable to failures that corrupt the addressing information.

In addition to the vulnerability from dynamic errors incurred during transmission, many network technologies require configuration tables to assist the network-routing and -addressing logic that may be vulnerable to static errors. The mechanisms to ensure the design correctness and run-time integrity of these configuration tables must also be evaluated and justified. These issues are discussed in sections 5.3 and 8 of reference A-4.

In some network technologies, routing information and logical topologies may be built at run time. An example is the tree-building discovery protocol of IEEE 1394. These mechanisms must obviously be evaluated in relation to their vulnerability to component failures or data corruption, unless failure modes or error detection can be suitably justified. The vulnerabilities that may cause the erroneous invocation of mechanisms that recreate routing information and logical topologies must also be understood and analyzed, since such invocation may seriously degrade (if not prevent) network operation. These issues are further discussed in section 5.5 of reference A-4.

Similarly, network error-handling logic (that may be invoked by erroneous addressing information or that may impact protocol flow) needs to be analyzed to establish a bound on the influence of the invocation of the error-handling logic and its impact (i.e., degradation) on network performance. The behavior of any such logic and its associated vulnerability needs to be analyzed and justifiably bounded. This is especially true for centralized intermediate stages, as discussed in section 5.3 of reference A-4.

A.3.2 NETWORK VULNERABILITY TO FLOW FAILURE.

As with network-addressing failures, the network technology's flow regulation logic also needs to be evaluated. Issues relating to acknowledgement and retry logic are discussed in section 5.9 of reference A-4. Issues relating to host interface load balancing and buffering are discussed in section 6.1 of reference A-4. Issues relating to intermediate stages are discussed in section 5.3 of reference A-4.

A.3.3 IMPACT OF INTERMEDIATE STAGES.

If a network encompasses intermediate buffering or relay stages, then the behavior, implementation, and impact of the intermediate stages needs to be established and evaluated with the network behavior. The behavior of these intermediate stages can vary considerably with network implementation. In simple form, they may be solely relaying stages. In more elaborate schemes, they can comprise store-and-forward and routing logic. Finally, in critical networks, it is common for such intermediate stages to incorporate error detection or fault containment mechanisms. This section discusses some of the issues and network attributes related to such intermediate-buffering schemes that need to be considered and evaluated.

A.3.3.1 Vulnerability to Intermediate-Stage Failure.

In networks that deploy intermediate stages, the influence of the intermediate-stage components may be significant. For example, in networks using stars or hubs, the intermediate-stage component impacts all the data flowing through it. The availability of the intermediate-stage component must, therefore, be analyzed and justified to be adequate to fulfill the network availability requirements. If multiple intermediate stages are deployed, then the independence of intermediate-stage failure should be analyzed and suitably justified. If intermediate-stage-to-intermediate-stage signaling is required, then this signaling and logic needs to be analyzed for failure vulnerabilities and possible fault propagation. Similarly, any protocol common-mode influence on the intermediate-stage availability must also be understood.

Integrity implications of intermediate-stage mechanisms must be carefully analyzed and evaluated. One of the difficulties of such an analysis is bounding the failure modes of the intermediate-stage component. Since the intermediate stage influences every bit that it is relaying, the effects of a faulty intermediate-stage component can be significant. The integrity implications of a failing intermediate stage are very much dependent on network implementation and architecture. For example, in the ROBUS network elements of the SPIDER architecture (which votes data from three independent channels), the failure of a single intermediate-stage component can be easily detected and is effectively masked from the receiving node.

Alternatively, if a network intermediate stage is developed to have full coverage (for example, using self-checking or monitoring schemes), then the failure modes of the intermediate-stage component may be suitably justified as benign (e.g., fail-stop). It is imperative, however, that the coverage of the self-checking or monitoring scheme is suitably justified, as discussed in section 7.4 of reference A-4.

It is common for networks to rely on in-line integrity mechanisms, for example, CRCs checksums, parity, etc. In such cases, the failure modes of the intermediate-stage component become more significant, since the network integrity is dependent on the coverage of these codes. With complex intermediate-stage logic, it is difficult to bound failure modes of the intermediate devices, and relatively simple failure modes may have significant impact on in-line coverage techniques. To illustrate the impact of a relatively simple failure mechanism, consider the scenario of an intermediate-stage elasticity buffer erroneously underrunning or overrunning. If the result of such an overrun or underrun is the insertion or deletion of a single cell from a relayed Manchester stream, the resultant relayed stream may suffer a cell shift that causes data corruption for the remainder of the transmission. If such a failure is not detected by the encoding or framing scheme, this shifted data stream may easily defeat CRC coverage (as discussed in section 4.4 of reference A-4). In such cases, the data integrity claims of the network are, therefore, limited to the failure rate of the relaying component. This scenario is important as it illustrates the interdependencies of the error detection logic (i.e., the framing and encoding layer) strength and the CRC coverage. Strict enforcement and error detection mechanisms may strengthen the data integrity claims; and with that said, quantifying such behavior may be difficult. It is also important to understand where the error detection is performed. For example, if the error detection is only performed at receivers, and intermediate stages do not perform such action, the reshaping or retiming behavior of the intermediate stage may degrade the end-to-end

effectiveness of such detection (i.e., the scenario of erroneous signals at the intermediate-stage input getting cleaned up and reshaped by the intermediate-stage action that produces a relayed output stream with no encoding errors). The impact of reshaping and re-encoding layers of intermediate-stage logic needs also be considered in this regard.

In addition to hard or transient logic errors, intermediate stages may also be vulnerable to out-of-specification behavior. For example, clock drift may lead to similar overrun scenarios, as described above. The network vulnerability to such errors, together with the potential contributors to such out-of-specification behavior, need to be understood as the network is evaluated. It should be noted that there may be both systematic contributions, such as long-term drift of oscillators and their performance under aging and temperature variations, etc. They may also be due to local transients; for example, acceleration or gravity forces on crystals or PLL modulations resulting from power supply instability or fluctuation. It is obviously important that the intermediate-stage elasticity buffers are sized to accommodate such variations. In addition, the intermediate-stage reaction and response to out-of-specification errors is another attribute that warrants careful consideration.

For intermediate stages that encompass store-and-forward behavior, the situation is complicated further since the behavior of the intermediate-stage component is more complex. The vulnerability of the intermediate buffer memory to transient upsets (such as SEUs) needs to be established. It is preferable if some form of protection is in place. If only error detection is in place (for example, parity mechanisms), then intermediate-stage response to such errors needs to be analyzed and understood. For example, if a parity error causes a reset or machine-check exception, then the availability of the intermediate stage would be impacted as the reset procedure is initiated. The vulnerability of the intermediate stage to SEUs and the subsequent reinitialization time will then need to be considered when justifying network channel availability. It should be noted that a similar analysis is also required for software-implemented switching schemes that use random access memory (RAM)-based data with parity-type schemes.

If the intermediate buffer memory is not protected via parity or EDAC schemes, then the impact of such upsets on end-to-end integrity claims needs to be understood. Obviously, should the intermediate stage perform recalculation of integrity checksums (such as CRCs), then the impact to buffer memory upset is limited by the SEU vulnerability of the intermediate buffer RAM.

In addition to buffer memory errors, faults of the intermediate-stage configuration and routing tables also need to be analyzed in a similar manner. If these are not protected, then the impact of erroneous control flow, routing information, etc., needs to be carefully analyzed. The responses to detected errors also need to be understood in relation to their impact on intermediate-stage availability, as discussed above.

Permanent faults of the intermediate-stage control and buffering logic need to be considered. As with the simple relaying logic, these may impact data integrity claims. In addition, when buffering action is present, the vulnerabilities to erroneous message forwarding need to be analyzed and understood. Network mechanisms to detect old or out-of-order packet forwarding must, therefore, be analyzed and evaluated with the network performance, unless suitable benign

failure modes of the intermediate stage can be justified via coverage techniques (self checking, monitoring, etc.).

For protocols that incorporate control flow information in the transmission, for example, the reset-sequence indicator proposed by ARINC 664 part 7, the erroneous behavior of a single network channel needs to be considered in the redundancy management of networks. Mechanisms for redundancy management can be potential logical fault propagation paths and, thus, need to be considered when justifying the network availability. For example, consider the scenario of an ARINC 664 P7 babbling switch that only sends frames with sequence number of 0. Such a failure could be due to a stuck address line in a switch, resulting in continuous sending of the same frame, which might happen to be a frame announcing a reset. The receipt of such a frame on either channel (of the dual-network paths) causes the receiving node to reset its frame sequence for a virtual link once (if the redundancy mechanism of ARINC 664 part 7 is used). Hence, the erroneous channel may upset the sequencing and data flow of the independent, good channel at least once. The reset-sequence indicator will only be considered once due to the integrity mechanism in ARINC 664. Thus, the influence of a stuck frame is bounded. Failure modes with more complicated failure behavior that could defeat the integrity check of ARINC 664 end systems may have to be considered if it cannot be argued that their probability of occurrence is low enough.

Note that due to the centralized position of the intermediate stage, failure effects, such as those described in the previous sections, may touch many parts of the system. Hence, the common-mode influence of such intermediate-stage behavior needs to be carefully evaluated and understood.

In addition, the intermediate-stage buffering mechanisms and protocol interactions need to be understood to mitigate issues relating to head-of-line blocking. This same problem can occur in application services, as discussed in section 6.1.1 of reference A-4.

A.3.3.2 Vulnerability of Intermediate Stage to Fault Propagation.

The vulnerability of the network intermediate stages to faults propagating from erroneous end nodes should be established. Such vulnerabilities may be related to erroneous control data or erroneous temporal behavior. For example, consider a central guardian reintegrating onto a TDMA scheme; if the TDMA position is resolved by listening to TDMA sequence index indicators included in the TDMA traffic stream and if the integration logic of the intermediate stage is not tolerant to erroneous information, a faulty node may be able to delay or prevent the intermediate stage recovery.

Similarly, incorrect flow management may impact the intermediate stage or switch performance. For example, babbling end nodes or other babbling intermediate stages and switches (sending syntactically continuous frames) may impact available buffer space and cause overruns unless suitable enforcement and error containment policies are in place. The buffer management policies, associated buffer sizing, etc., need to be carefully analyzed, as network performance under normal and erroneous node behavior is justified.

Finally, any error-handling logic that may be invoked in response to erroneous end-node traffic and behavior should also be analyzed so that any associated intermediate stage or switch performance degradation or other propagated erroneous behavior can be suitably bounded.

A.3.4 NETWORK CONFIGURATION DATA.

Many network technologies require configuration and routing tables to be programmed to assist network operation. Therefore, design correctness of these tables is obviously important to correct network operation. Design assurance issues relating to network table correctness is discussed in section 8 of reference A-4. The run-time integrity of the tables is also important. Therefore, the storage, operation, and load integrity mechanisms of the configuration data need to be evaluated with the network technology. This examination should also address run-time table placement, for example, RAM protection schemes such as parity and EDAC. Similar considerations, as to those discussed in section 5.3.1 of reference A-4, in relation to buffer protection and recovery actions should be considered in relation to run-time configuration table placement.

In networked systems, the consistency between the copies of run-time tables in different nodes is also an important issue. Hence, protocol mechanisms to ensure table consistency should be evaluated. However, as discussed in section 5.5 of reference A-4, the availability impact of such consistency enforcement mechanisms needs to be considered. In systems where the network tables and system or application software are tightly coupled, mechanisms to ensure software/network compatibility are needed. This is especially true if the network tables are configured separately from the application software images.

The mechanisms to load the configuration and routing tables are also important. First, the integrity of the loading mechanism needs to be established so that the configuration data does not get corrupted during the load process. Second, if the table load mechanism uses the same data path as the normal network data flow, the partitioning properties of the network path have to be established. For example, some network technologies use dedicated load protocols to facilitate the loading process. The interlock mechanisms and mode selection logic used for these load protocols need to be analyzed to ensure that the erroneous invocation of the protocols does not degrade network availability. In addition, network tables may need to be updated in a "live operational mode," for example, when a network is running minimal network traffic to support the "hotel functions" of operating doors, lights, and basic power distribution. In the live operational mode, the mechanisms to coordinate mode change and table switching need to be carefully evaluated.

Similarly, network maintenance and query protocols are sometimes used on top of the network infrastructure, such as in the Simple Network Management Protocol. These network maintenance protocols also can introduce safety implications. The network vulnerability to the actions performed with these protocols needs to be analyzed and considered. For example, if it is possible to invoke software exceptions via these maintenance interfaces, the impact of the exception processing on the normal application functionality needs to be bounded.

A.3.5 START-UP AND RECOVERY.

Network start-up and recovery mechanism are important since, in critical environments, start-up and recovery time of the system is often a key attribute of the system performance. The behavior of network start-up performance is, therefore, another attribute that requires careful evaluation.

During start-up, the network is usually more vulnerable to faults. Unless the start-up algorithms have been designed to be fault tolerant, or the network hardware has been designed with adequate fault containment, it may not be possible to guarantee a correct or timely network start. In such cases, the availability of the network channel will need to be re-evaluated to consider the impact of potential contributors to erroneous start-up action. To illustrate such a scenario, consider a TDMA protocol where the initial frame of the protocol contains a table version identifier. This may be sent explicitly or implicitly, i.e., buried within the CRC calculation of the network frame. If the first node to send at start-up sends an incorrect table identifier and the response of the other good nodes receiving such a frame is to back off for a defined period of time, the erroneous node can hold off network start-up. If that node continues to send, then network start-up action may be delayed indefinitely. This is an interesting example, because it illustrates the interaction between availability and integrity mechanisms, which often occurs when integrity patches are implemented on top of networks that have been designed with a availability mindset, which is very common in commercial off-the-shelf (COTS) protocols.

Another interesting issue relating to start-up is the constraint the network implementation may place on external aircraft systems, for example, power sequencing. The network may assume that network components are powered-on within certain intervals or in a specified sequence. Such is the case when central guardian action is assumed, as described in the section 7.2 of reference A-4. In time-driven networks, the alignment of start-up behavior may significantly impact network start-up. Such issues are further discussed in section 5.8 of reference A-4. It is, therefore, desirable to note such constraints while the network is being evaluated. If the network is required to be safety critical, the assurance of the network-assumed behavior may drive considerable complexity and cost into these other systems. However, without such assumptions, the network justification and associated availability claims may be incomplete.

In some network technologies, for example IEEE 1394, network-addressing and routing information is built when new nodes are added to the network. Such behavior should be carefully analyzed for its impact on network start-up time, which will essentially be limited by the slowest node. A faulty node, for example, a node undergoing continuous restart behavior, may also need to be considered (depending on the fault containment and coverage of the nodes implementation). If sufficient coverage cannot be justified, a faulty node may disrupt network availability by continuously initiating network restarts. In such a scenario, all components that may contribute to such a failure need to be analyzed while network availability is being justified. In addition, the fault tolerance of “discovery routing protocols” should be established and analyzed to bound the influence of faulty logical behavior.

Authentication is another issue that is often worth considering during start-up. This is especially true in TDMA networks that may use the temporal order of messages to assist as an authentication mechanism when the network is running. At start-up, when there is not an established time base, this authentication technique is not available. Hence, the network may be

more vulnerable to authentication failure. This is particularly interesting for dual-channel networks, because the lack of suitable authentication may enable a faulty node to impact multiple transmissions as it appears differently on each of the channels. If the algorithms of the associated network are only tolerant to a single failure, then such a dual-error manifestation may break the protocol assumptions and prevent correct network operation. This may even be true if guardian schemes are in place (because the guardians may lack suitable authentication capability). The strength of guardian enforcement is another attribute that requires careful consideration. This is discussed in section 7.2 of reference A-4.

In general, it is important that the network technology protocol mechanism and algorithmic claims are carefully evaluated for their performance during start-up. For example, a clock-synchronization algorithm may tolerate a Byzantine error when the data network is fully up and running, but may require a certain minimum number of correct nodes to be up before the fault tolerance mechanism can operate correctly. The impact of the algorithmic behavior when fewer clocks than the minimum are available should be understood. For example, what is the impact of having only two clocks available for an algorithm that requires four clocks to be Byzantine fault tolerant? Bounding such effects is required as the network performance and safety case is evaluated. In addition to the analysis of the network's start-up mechanisms, network technology reintegration mechanisms also need to be analyzed to establish their tolerance or vulnerability to erroneous protocol control flow information that may delay or prevent a node's timely integration.

Many algorithms and mechanisms are designed to work correctly only if some minimum amounts of good resources are available. However, just prior to start-up, it can appear that everything has failed. A good design for start-up must be able to get past this "everything has failed" phase and be able to bootstrap itself up to full operation. However, all too often, network designs assume that the network was "born running" and cannot tolerate failures during start-up.

A.3.6 GLOBAL SYNCHRONIZATION.

Data networks may have a need for synchronization of clocks within nodes for coordinated network access or as an application-level service. The following paragraphs focus on clock synchronization, but a subset of the aspects to be considered for clock synchronization are equally applicable to synchronization of logical clocks (counters) used for redundancy management.

A.3.6.1 Algorithm Mechanism.

Clock-synchronization algorithms consist of several steps during synchronous operation. The first step is the initial clock acquisition, in which synchronizing nodes acquire the current clock values or counter values from one or more different nodes. This can be done via messages prescheduled according to a table or can be triggered upon request. After a node has acquired synchronization with the data network, it must maintain that synchronization. It does so by periodically acquiring clock-difference information from the other nodes. This also can be done via messages prescheduled according to a table or can be triggered upon request. Another method is to time the arrival of normal data messages (i.e., expected arrival time of a message versus the actual arrival time) and infer the current state of the transmitter clock of that message

versus local clock state. After preprocessing of the clock-difference information (such as for the elimination of propagation delay influences), the second step at each node is the calculation of the correction value for the local clock based on the (preprocessed) clock values. This step is sometimes called the convergence function, because it should ensure convergence of the distributed node clocks towards a common clock. The next step is applying the correction value to the clock so that all nodes have clock times that are more closely synchronized to each other than before the synchronization step was taken.

Several properties and influences, if not mitigated, may lead to an unstable or failing clock synchronization algorithm, which can lead to potentially unsafe system state.

The clock-synchronization algorithms depend on the propagation delays through the network. Different propagation delay between different nodes or different data acquisition delays at nodes may lead to inconsistent or inaccurate views of the actual propagation delay, which is used to judge the clock difference between different nodes. Such differences may have effects on the stability of the algorithms. The most often cited challenge in clock synchronization is synchronization in the presence of Byzantine failures. Byzantine in this context means that different nodes have different views of the clock values from another node. It seems hard to quantify the possibility of Byzantine scenarios in the clock synchronization at first. Considering that clock-synchronization algorithms often measure the difference between expected arrival of a message and the actual arrival, any arrival time differences of sync data in the context of different or slightly varying propagation delays lead to scenarios similar to Byzantine fault scenarios. If the system does not compensate for propagation delay differences, the views of the clock values can be significantly different. Compensation of propagation delays decreases the difference but does not remove them. Such scenarios of different views of clock values at different nodes, if not considered in a stability analysis, may pose a safety threat. The stability analysis is normally captured in an analytical bound of a precision value.

The clock data values may also contain some faulty values either due to failures during propagation or due to end-node failures. Considering source coverage, the correction value calculation may have to tolerate certain failure modes to achieve a bounded precision value. One issue in clock synchronization is the combination of clock values from different communication paths that stem from the same source. In case of triplex redundant channels, the clock values arriving on different communication channels can be voted. For dual replication, voting is not possible. In certain topologies, an intermediate active component may have effects on all values from different clock sources that travel through it; e.g., a single short-circuited central star component in TTP/C or FlexRay may have influence on all clock values from different sources. If the star topology is only dual (which is the case for most well-known data networks aimed at embedded real-time systems, such as TTP/C, FlexRay, and Avionics Full-Duplex Switched Ethernet), these failures of the stars can cause failures in the end systems that need to be considered by the end systems themselves, due to the fact that the duel inputs from the data network cannot be voted. The selection process between the different replicated communication paths determines the amount of influence a failure in one of the communications can have on the overall precision values. This effect, unless restricted, may have dependability implications.

The algorithm calculating or selecting the correction amount for the local clock should consider the assumed failure conditions and the number of faulty nodes it may have to tolerate in the context of the systems source coverage mechanisms to ensure a bounded precision. An analysis of the precision should consider the effects of potential masquerade failures. For example, in FlexRay, the correction value calculation function can tolerate several incorrect clock values stemming from different nodes, but may be unable to tolerate a single faulty node, if the node fails such that it masquerades as other nodes when transmitting synchronization frames.

In the case of master and slave synchronization schemes, the switchover time from one master to another master used for synchronization may need to contain the time to diagnose a faulty master. During such a diagnosis time, slave nodes may still synchronize to their (faulty) master node or not synchronize at all. Both scenarios can affect the precision.

When a node applies the correction of the clocks to its local clock, any task dependent on the local node clock may have to consider potential influences of this correction on its execution time. For example, the correction of clocks may have implications on the period available for the execution of tasks. Effectively, the execution time available to a node may be shortened by the precision due to correction and coordination with other nodes. This influence may have impact on the execution time available to tasks.

The clock-synchronization algorithm needs to be careful in what information it uses to do clock corrections. For example, if the correction function of the clock-synchronization algorithm uses the same collected time values twice for calculation of the correction values, the algorithm can become unstable. Such a scenario may happen during start-up of the system. Such instability has been observed. Any tool verifying the stability properties of clock synchronization or person analyzing the properties should consider such effects and different configurations. Also, the compatibility and consistency of the clock-synchronization configurations at different times should be checked.

Start-up of clock synchronization may have availability implications if it is dependent on the fault coverage of a single node. Similarly, during start-up, clocks may drift for a much longer time than during normal operation, because less or insufficient clock value sources are available. Such longer drift times should be included in the stability analysis and consequently the precision values.

A.3.7 FAULT DIAGNOSIS.

Some network technologies include fault diagnosis services to identify and isolate faulty member nodes. The services may run autonomously in the network hardware or comprise software application services that run on top of the network, which diagnosis information provided by the network layer. Such services are strongly related to group membership and interactive consistency services, which may use fault diagnosis services to manage network state-dependent application decisions and guaranteed consistent delivery of messages.

A group membership service delivers the operational status of some or all nodes of a data network to other nodes. Group membership service, or variants thereof, is a subset of an interactive consistency service. Group membership indicates the operational state of nodes

(ideally consistent), while interactive consistency provides consistent agreement of nodes on any (sent) value.

Group membership information indicates the health state of a node. It can be concluded from this information that the node is operating correctly. Yet, the information that a node sends out may not reflect the current state of a message; e.g., the operation of a node as indicated by group membership does not ensure that the message is correct (integrity violation) unless sufficient error detection coverage for the node and the communication path is assured.

Group membership is usually derived from the correct or incorrect reception of a message from a node. If these messages are correct, group membership infers that the node is correct. On the other hand, if the reception of a frame is not correct, group membership mechanisms can attribute this to a transient or permanent fault on the communication path. In aviation digital electronics systems where transient upsets may be experienced in relation to power drop outs or massive upsets from HIRF or lightning events, the ability of the diagnosis schemes to distinguish transient external upsets and permanent node errors should also be carefully analyzed to ensure that the diagnosis algorithm meets the real-world expectations. The persistence of any indictment action that may result from the invocation of such diagnosis also needs to be understood to ensure that the loss of network availability resulting from such indictments is suitably bounded.

Requirements of group membership being consistent and the effects, such as inconsistent reception status of messages at different receiving nodes and potential consequences, are discussed in detail below. In general, it should be said that any diagnosis service will not be perfect, e.g., due to transients having local effects or due to failure modes of the sending nodes (Byzantine failure modes).

A.3.7.1 Application Use of Diagnosis Information.

Diagnosis information can be used by the network to build additional services for management of redundancy sets or simply as acknowledgement. In this context, the use of the diagnosis information needs to be in alignment with the expectations of the applications. These services are (1) group membership, from an application perspective, intends to use the membership information for the selection of the correct data, and (2) interactive consistency intends to provide the data consistently at all nodes.

Group membership is often based on the reception of messages from nodes and will have some temporal lag until state changes are updated. Effects of time lag should be considered in the evaluation process.

It has been proven that group membership in arbitrary fault scenarios (without source coverage) cannot guarantee correctness and consistency at the same time, whereas correctness means that all correct nodes are regarded as nonfaulty by all other correct nodes (but faulty nodes may also be viewed as correct), and consistency means that all correct nodes are consistently seen as correct at all correct nodes. An implementation of group membership with insufficient coverage or insufficient communication rounds required (and theoretically proven) to tolerate certain

failure modes may either sacrifice availability or integrity; e.g., enforcing consistency in a single string implementation can lead to availability loss.

In CAN, any acknowledgement algorithm claiming consistency (atomic broadcast) despite an arbitrary failure mode should be analyzed in a similar manner. This is discussed in section 5.9 of reference A-4.

Applications that use networks that provide group membership services should analyze

- the underlying assumptions,
- the consistency and correctness guarantees of group membership, and
- their effects on the application level.

Such analysis and effects on the application should also include temporal aspects, because diagnosis information and membership lag in time.

During reintegration and start-up of the data network, the group membership information provided by a newly integrated or started node may include information about the system's state, which the network may not have observed itself or obtained from other nodes. In detail, integrating nodes may observe the network's operational state over a period of time or may integrate quickly by accepting other nodes' views of the operational state of the network. In the latter case, the use of the information provided by other nodes needs to be in alignment with application-level assumptions of the membership information; e.g., if an integrating node adopts the group membership state from other nodes and the application assumes that the membership state information includes agreement on, or availability of, application state information, it may also have to acquire the application state information that is associated with the semantics of a group membership bit.

A.3.8 CLIENT EFFECT ON NETWORK OPERATIONS.

A data network is often the glue that holds together a dependable system. A system data network tends to become either the main fault containment mechanism in itself or is a major component of the main fault containment mechanism(s). As such, it is important for a system data network to not be adversely affected by the clients it serves, no matter how badly the clients misbehave.

Many data networks allow their clients to adversely affect their operation in several ways.

The first adverse interaction occurs immediately upon start-up. Many data networks allow their clients to influence the timing of network start-up by affecting the timing of their nodes. Variations in node start-up times can be caused by different host power-up sequences, different self-test mechanisms, etc., coupled with the requirement for the client to enable its node to participate in the network, e.g., in FlexRay and TTP/C, the host needs to switch on the controller. Different start-up times of node components should not be allowed to cause starvation of components (retry exhaustion). For example, after some retries of an insufficient response, a FlexRay network chip starts over again and the software needs to interact; if software is too slow, then there is no availability. It is possible for data network protocols to take an

inordinately long time to start, or it may not start at all, if the timing behavior of its nodes follow some pathological pattern during start-up. When such networks are used, the performance of higher-protocol layers (such as those implemented in software) should be analyzed to ensure proper start-up time and to avoid problem scenarios like those described here.

During data network operation, some protocols allow clients to adversely affect their behavior if the clients can control addressing, routing, priorities, etc. Some systems require applications of different safety-criticality levels to share the network. When this is the case, the network must be robustly partitioned so applications and clients of low criticality cannot adversely affect the use of the network by high-criticality application or clients.

Another possible avenue for a client to adversely affect a system data network is via unprotected test or network management paths.

A.3.9 ACKNOWLEDGEMENT.

For network protocols that employ acknowledgement schemes, the behaviors of this logic need to be carefully analyzed, especially with respect to inconsistent message reception (some nodes receive a message or an acknowledgement and some do not). It cannot be assumed that any acknowledgement mechanism provides, by itself, consistent message reception also called atomic broadcast). Also, the sender of the message may fail before resending the message; e.g., mechanisms with negative acknowledgement schemes need a way to signal such negative acknowledgement. If signaling is not possible, inconsistencies may arise. An example of acknowledgement causing inconsistent message reception is the negative acknowledgement algorithm (sending of error flag) in CAN. If an inconsistent bit reception in the next-to-last bit occurs, some nodes will accept the message and others will not. In such a scenario, a retransmission will occur, leading to multiple message copies at some receivers and a single message copy at others. As a consequence, the delivery semantics have gone from “exactly once” to “at least once.” Receiving nodes may not be able to distinguish the duplicate message from a legitimate second message, and message delivery to different nodes may occur at different times. In case the sender suffers a failure and it is not able to resend the message, permanent inconsistencies in message reception will arise. The implications of inconsistent message delivery, different message delivery times to the application, and multiple deliveries should be analyzed with respect to the overall system and its safety.

Similarly, in a network where acknowledgement vectors or bits are used, an inconsistent reception may cause system-level effects. Generally, if acknowledgement is only based on an action of a subset of nodes, inconsistencies may occur as a consequence of the design (e.g., the recessive or dominant physical layer acknowledgement, in which one receiver is sufficient to signal a dominant state, is an action of only a subset of the receivers; or the acknowledgement-signaling mechanism relying on the reception status of a subset of receivers, such as the reception status of the next one or two receivers, may be vulnerable to inconsistent reception).

Message retry mechanisms, due to negative acknowledgement or missing positive acknowledgement, may have implications on the network performance and maximum loading. To bound network loading, retry mechanisms should be analyzed for the number of retries to

make sure they are bounded, or they should be analyzed whether retries are forced to be bounded via counters for retransmissions or bounded via timeouts.

Acknowledgement errors can affect application-level error handling or exception mechanisms, such as invocation of error routines leading to additional overhead for processors. Any safety implications of increased workload should be analyzed.

A.4 APPLICATION SERVICES.

Current data network technologies comprise a number of application services that may or may not be used by an application. All services need to be analyzed in the context of a safety assessment. In its simplest form, any buffer management mechanism has associated properties that need to concur with the application assumptions. Newer generations of networks also supply voting schemes or redundancy management mechanisms. An example of such buses is the ARINC 664. In this section, the criteria for data network services used by applications are examined.

A.4.1 HOST INTERFACE MANAGEMENT.

Buffer management should be concerned about the message access order to the network, partitioning requirements, and performance aspects of the network interface buffer, as well as implications to the host.

A.4.1.1 Client Buffer Queue Management.

Buffer management of systems may have system-level implications. One example of system-level impact may occur if messages are associated with priority. In certain combinations of buffers and accesses, priority inversion on the system level may occur; e.g., certain implementations of CAN can have a priority inversion of messages.

These CAN implementations have a priority message queue that holds a large number of messages and an intermediate buffer that intends to contain only the two highest-priority CAN messages. The buffer with the highest-priority messages is used for network arbitration and for serializing and sending on the network out of this buffer. This is called the sending buffer. One message position in the sending buffer, which is normally a dual-port memory, is intended for sending, while another position is intended for "refilling" from the larger message buffer with the next higher-priority message, while the message on the other position is sent on the network. If a higher-priority message (priority 3) arrives at the large message buffer, the lowest priority that is currently in the sending buffer (priority 4) needs to be replaced by this new message. If this replacement action coincides with arbitration on the network, another message with lower priority (priority 5) may win the arbitration, because one position of the sending buffer has just been sent and is empty and the other message (priority 4) is being replaced by a message with a priority 3 message. This is an example where a lower-priority message has won arbitration over a higher one. While this situation can be improved (i.e., not suffering from priority inversion) by supplying a sending buffer with space for three messages, similar situations may exist, and the system-level and safety implications should be checked due to such scenarios.

Similar phenomena can occur due to local buffer management. One example is a priority arbitration scheme where only a single first-in, first-out transmit buffer is used. If only the head-of-line message contends for the communication resource, the performance drop due to head-of-line blocking can be significant. In the worst case, a node may not get any access to the network and will not be able to send.

When evaluating networking technology for the deployment in systems, client buffer queue management mechanisms should consider effects on the access to the network, such as fairness and implications to the network and the host.

A.4.1.2 Buffer Management Partitioning.

In a robustly partitioned system, software partitions running on a node have a strict execution budget and should adhere to it. On nodes where the data from a data network is managed by a direct memory access (DMA) controller, the DMA controller may repeatedly stall the execution time of running partitions, potentially having significant effect on the execution time of software tasks. Unless such effects of "cycle stealing" are accounted for in the execution budget of software tasks or the overall node architecture, software may miss execution deadlines.

Partitioning violations, due to addressing and masquerading nodes, were discussed in section 5.1 of reference A-4. Partitioning violations may also occur due to buffer management. In systems having applications of differing safety criticalities running on one node (processor) and each having common access to the communication buffer, any wrong access to the common communication system buffer can result in several undesired phenomena.

Unless the access to the common buffer is restricted or controlled for each partition, software partitions may overwrite messages of other partitions or use network resources from other partitions. A partition may even be able to send data masqueraded as another partition, unless protected. A simple, but potentially unsafe, example may be a common address area where all partitions have access, but each partition is assigned a source address on the network based on its assigned range within the buffer. Any faulty access to another partition's memory area can result in faulty addressing on the network, masquerading effects, and data overwrite, just to name a few potential safety hazards.

Another area of control to the buffer is the coordination between the network and the software (or host processor) sides of access to the buffer. The buffer management should be analyzed to ensure the mutual exclusion of buffer access (or access to certain areas). Unless the access is controlled for both the data area and potential control areas (interaction between status area potentially updated by the network, while at the same time, the software side tries to change or read the control information), interactions may result in unwanted effects. One example is the atomic write for messages sent on the network (transmit buffers). A message should not be sent until it has been completely written into intermediate buffers. If the contents of a transmit buffer are sent out onto the data network's media while software is still writing to that buffer, the resulting transmitted message could contain contents that are a mix of old contents and new contents. While such coordination may occur automatically for different processes due to scheduling of process execution on processors that is tied to a communication schedule for the

data network, the dual-port memories often used for buffer management may be much more likely to be affected by such coordination errors.

The host needs access to the receive-buffer areas common to the networks, and the host needs to be restricted or otherwise carefully controlled during the reception of incoming traffic from the data network. If not coordinated or controlled, data inconsistency in applications may arise. If blocked from either side, blocking effects should be considered. Blocking of message reception while the host reads the buffer (if such blocking is possible) may have system-level impacts, such as requiring the re-sending of messages or queuing at sender side. Blocking the host-side access while the data network is updating a receive buffer may increase the execution time of software. In cases where dual buffers are used to allow an incoming data network message to be written to one buffer while the host software is reading from another buffer (ping-pong buffers), potential delays in the availability of data and the switchover logic after message reception needs to be considered.

Well-designed data network interfaces that have solved these receive buffer access problems for all corner cases on a single node may still have problems for architectures that use broadcast messages. It is possible that the receive buffer mutual exclusion mechanism on each of the receivers works correctly (the host never receives messages from the data network that have inconsistent contents due to buffer timing and access issues), but could cause the atomic property of the broadcast to be lost. That is, timing differences among the receivers may cause different receivers to see their buffers in different states, even if they all receive exactly the same sequence of messages from the data network. In the cases where atomic broadcast must be supported, the data network may also be required to support receive buffer consistency.

Similarly, network errors that can trigger host software exception loops also need to be considered. This is to ensure that such exception loops do not interfere with the time budgets and partitioning mechanisms of host software.

A.4.1.3 Buffer Management Performance Considerations.

The performance considerations of buffer management should be considered when selecting a network. In the past, the low-speed aviation digital electronics networks (such as ARINC 429 and 629) have put less emphasis on the performance of buffer management, because memory access times or memory bus access times were often an order of magnitude quicker than required for serving the data-copying and coordination activities. With the advent of high-speed communication in avionic systems, the need for a balance on the buffer management side with respect to performance becomes more prevalent. Performance evaluations should consider the required access needed from the network and the host sides, memory device and memory bus access times, and special support provided by the hardware, such as burst memory access. Interactions between performance enhancement schemes, such as burst memory access that reduces buffer access time, and interactions with access time and requirements (e.g., blocking of memory devices or memory bus may have implications to arbitration of the memory or the memory bus) should be considered in the evaluation.

For network technologies that require software functions to assist the network data flow (for example, data unpacking, data copying, etc.), the software impact of changing the network tables

also needs to be considered and suitably bounded. This is especially true of network tables that are configured and loaded independently of software application images. Ideally, software execution margins can be suitably bounded and argued to meet the worst-case network data flow assumptions that can be run-time configured. Network technologies that support the bounding of such interactions are preferable.

A.4.2 SUPPORT FOR APPLICATION-LAYER REDUNDANCY.

A.4.2.1 Support for Active Replication.

Networks may signal the application of reception status, which may assist the application in voting or selecting a correct value. Such mechanisms should be evaluated with respect to their correctness. In case the indication status stems from the same source as the possible faulty value, the use of such status information might be limited.

Application-layer membership is a mechanism to manage the redundancy sets at an application level. Such application-layer membership algorithms should be evaluated with the same scrutiny as the node-level memberships described in section 5.7 of reference A-4. One example that can be regarded as application-level membership information is the network management vector in FlexRay.

Node-level and application-layer membership is often combined within some networks to incorporate message agreement and redundancy mechanisms. Such services provide a foundation on which to build active replication strategies for applications. For example, the National Aeronautics and Space Administration ROBUS protocol used in the SPIDER architecture presents voted message data to the network interfaces, containing the visibility and impact of erroneous data to within the network infrastructure. In another example, one version of TTP implements enforced message agreement strategies, where nodes not in agreement with the majority of network nodes are forced to re-integrate. While the membership mechanisms of these two networks can be equally effective in providing a consistent view of system-wide membership, there is a difference in the amount of system resources that are adversely affected while these mechanisms sort out errors. For example, a Byzantine error in a SPIDER architecture is effectively masked with the network layer. In one version of TTP, depending on the degree of Byzantine fault containment provided by the guardian in a particular fault scenario, the same error may force multiple nodes to re-integrate. The side effects of these policies and their effect on applications should therefore be understood as the network technology is evaluated.

In some networks, the network host interface incorporates a life-sign mechanism to support application membership and health diagnosis. A life-sign mechanism requires an application to perform a specific action that is used to judge an application as correct. Based on the correctness of the action, the application may be removed from the membership. The life-sign action should be evaluated with respect to its effectiveness of detection of the failures. Due to the minimal action in normal operation, the error detection coverage may be limited.

A.4.2.2 Support for Passive Replication.

Some networks support mechanisms for passive-redundancy strategies, i.e., the ability of multiple network nodes to share network bandwidth. These mechanisms are discussed in section 7.7 of reference A-4. The networks mechanisms to inform clients of the state of the passive-redundancy scheme, i.e., what application is in control and how many "spares applications" are online, should also be considered. Since such information may aid the detection of latent spares exhaustion, services to synchronize the state of spares should also be evaluated to ensure that such mechanisms do not introduce potential fault propagation paths.

A.4.2.3 Support for Increased Integrity.

Some network technologies implement host interface support for self-checking pair host configurations. Self-checking pair data is compared, and if it agrees, it is delivered as correct. Self-checking pairs-based input data should be compared before computation; otherwise, the self-checking pair computation results are likely to diverge even though both halves of a pair are correct. Self-checking pairs should also be evaluated with respect to their independence from power, common memory, vulnerability common design faults, etc.

Self-checking host support is strongly influenced by the network-level, self-checking mechanisms discussed in section 7.4 of reference A-4.

A.4.3 TIME SERVICE FOR TIME STAMPING AND TIME INTERRUPTS.

Application time services that may be supplied by the data network include time stamping and time interrupt. Synchronization aspects of time have been discussed in section 5.6 of reference A-4, including a discussion of the implications of time services to the applications.

The quality of time services can be adversely affected by a data network time-service design that is not robust. Time stamping of data allows an application to determine data freshness. Sometimes all that it is needed is ordinal freshness; that is, the application only needs to know what data set is newer. In some instances, an application may use time stamps to determine the interval between two data samples, which could effect calculations that use delta time. Some applications may use data network supplied time interrupts as a replacement for a local real-time clock to do task scheduling. This has the benefits of a time source that is independent of the effects of possible faulty software and allows for the synchronization of task scheduling among multiple processors. If these services from the data network are faulty, either from network internal faults or by propagating faults from clients, a time stamp service could cause wrong time values to be used as inputs to calculations or, when coupled to a host's tasking clock, could cause tasks to not have enough time to execute.

A.5 FAULT TOLERANCE MECHANISMS.

Some network technologies incorporate fault tolerance mechanisms to mitigate the failure of network components, such as guardians and monitoring schemes. Such mechanisms may be particularly advantageous in aviation digital electronics environments where high-network

availability and integrity is required. These mechanisms and associated evaluation criteria are discussed in the following sections.

A.5.1 TOPOLOGICAL FAULT TOLERANCE.

The network topology may have a significant impact on the network tolerance to zonal or spatial proximity faults, for example, physical damage that affects a certain area of the vehicle.

If the network uses a bus topology, then any failure along the bus path may destroy network availability. Similarly, faults in network termination may lead to loss of availability and may also introduce other Byzantine vulnerabilities, as discussed in section 7.8 of reference A-4. The bus zonal vulnerability is particularly important if multiple redundant buses are assumed to increase network availability. If all units are connected to all buses, then all buses are required to be in physical proximity at the point of their interface to the different nodes. A failure at this point of interface may therefore damage all of the independent bus channels. Similarly, a chronic failure of a node, (for example, fire) may also damage all buses that are close to the node. Therefore, when evaluating the suitability of bus-related network technology, care should be taken to ensure that the technology or network architecture has suitably mitigated such zonal vulnerabilities, either by separating bus and or by isolating network interfaces. The secondary effects of incorporating isolation schemes should also be considered in relation to their impact on the physical layer performance and the potential to Byzantine failure, as discussed in section 7.8 of reference A-4 and section A.5.8 herein.

Networks using intermediate stages may perform better in relation to zonal fault tolerance, as the point-to-point relaying action of such technologies alleviates the impact of physical layer damage. However the placement and data path planning of such intermediate-stage schemes should also be considered, as the network technology is mapped to a vehicle architecture; i.e., there is little benefit in placing two redundant central intermediate stages in the same location.

A.5.2 GUARDIAN SCHEMES.

Some network technologies incorporate covering functions or guardian mechanisms to contain node faults. Such guardians may be argued to increase network availability. However, the implementation and performance of the guardian function needs to be carefully evaluated to verify that suitable coverage and independence is provided.

There are several variants of guardian implementations; they may be locally (i.e., one node) implemented on-chip or placed with independent guardian chips. Alternatively, the guardian action may be supplied by network intermediate stages, for example, in centralized guardians or peer-based ring schemes. The first attribute that needs to be considered in relation to the guardian action is the amount of coverage that the guardian provides: i.e., what failure modes of the node does the guardian contain. Often due to the cost optimizations, the coverage of the guardian may be focused to cover only a subset of a node's failures. For example, in low-cost TDMA networks (e.g., FlexRay and TTP/C), local guardian schemes are often limited to time-window enforcement. The extent of the protection provided is also limited to specific network modes; for example, network start-up is often left uncovered. Time-window enforcement does not protect against logical protocol errors, for example, erroneous protocol signaling. Such faults

must therefore be mitigated with additional guardian behavior or fault-tolerant protocol logic, as described in section 7.3 of reference A-4.

Irrespective of the coverage provided by the guardian scheme, the independence of the guardian enforcement is another attribute that requires careful consideration. Often in network technology targeted for low-cost domains, the guardian function may be implemented on the same die (silicon integrated circuit (IC)) as the communications controller. The justification of independence may therefore be more difficult; as such, schemes may be vulnerable to common-mode failures that disable or degrade the guardian actions. The use of independent clocks and partitioned dies may assist here, although detailed analysis of failure modes will be needed to support independent failure claims. Another common dependence may be the power source. Network technology with truly independent physical guardian action will require less analysis and may be preferred as it presents less certification risk.

In addition to the physical independence, logical guardian dependencies should also be considered. For example, if the guardian is dependent on its host controller for global time or protocol state synchronization, the coverage of the guardian may be compromised. For example, consider a TDMA time enforcement guardian that relies on its host for schedule synchronization. If the host is "deaf," i.e., simply unable to hear network traffic, it may continuously try to start. If it performs in accordance with the correct start-up activity, then—from the guardian's perspective—the faulty host may appear to operate correctly. In reality, it will be continuously disturbing protocol traffic. Such dependencies should be considered when network and guardian technology is evaluated.

To mitigate the shortcomings of simple local guardian schemes, several network technologies have evolved to incorporate intelligent central guardian schemes. The degree of intelligence in the central guardian is dependent on the network technology, varying from simple time enforcement and slightly-out-of-specification (SOS) fault containment, to full protocol-level policing functions; e.g., protocol semantic-state enforcement or similar message policing. Centralizing these protection mechanisms allows for more intelligent guardians to be implemented at lower costs. However, implementations of the guardian schemes should also be evaluated to ensure that they provide adequate levels of independence and fault coverage. Protocol and node failures not covered by the guardian will need to be addressed by other means, either by fault-tolerant protocol logic (discussed in section 7.3 of reference A-4) or additional fault detection implemented on the client nodes, such as self-checking, as described in section 7.4 of reference A-4.

The use of intermediate-stage guardians introduces additional constraints on the target system. Consider for example, a dual-star (central guardian) network configuration. If the implementation of the central guardians lacks sufficient fault detection coverage, then it is difficult to bound the failure modes of the guardians. The influence of a faulty guardian on protocol action must be established. For example, is it possible for the guardian to cause nonrecoverable protocol flow errors in the establishment of disjoint TDMA cliques if the other (good) guardian is not available? If this is the case, then a system-level power sequence may be required to ensure at least one good guardian before the end nodes commence communication. In addition, the vulnerability of the guardian implementation to transient errors (SEUs, etc.) will

need to be bounded, as such events may take the good guardian off line long enough for a faulty guardian to force irrecoverable error scenarios.

As discussed in section 5.3.2 of reference A-4, the implementation of central- and intermediate-stage guardian integration and start-up logic schemes should also be evaluated to ensure it is suitably fault tolerant to erroneous end-node faults. If guardians from different network availability channels share signals or protocol state information, then the vulnerability of such mechanisms to failures of the other channel guardian failure should also be evaluated. Similarly, the self-test and scrubbing of intelligent guardian actions may be challenging.

Irrespective of any guardian implementation, it is imperative that suitable tolerances for guardian enforcement action are established to provide suitable design margin. As with other critical protocol parameters, these tolerances should accommodate for worst-case aging and expected life-time degradations of all components related to the guardian. The criteria for establishing suitable guardian parameterization would ideally be formalized and verified.

Latent failure of guardian schemes is another consideration, as discussed in section A.5.6.

A.5.3 PROTOCOL LOGIC FAULT TOLERANCE.

Networking technology may also incorporate protocol flow and algorithmic fault tolerance strategies, i.e., voting on protocol-state information or required protocol actions. Such voting may effectively contain protocol-state faults propagating from an erroneous node or other network device. The fault-tolerant global clock synchronization action discussed in section 5.6 of reference A-4 is an example of such action. Similar strategies may be applied to other protocol actions, such as start-up, reintegration, and mode change. The strength of such protocol mechanisms should be evaluated in the context of the coverage provided by the network implementation. For example, if all nodes are self-checking, then little protocol-state fault tolerance is required, as all protocol errors are contained at the source and justified to be benign. Similarly, if the guardian mechanisms contain protocol flow errors, then less protocol state fault tolerance is required. However, if suitable fault containment or coverage cannot be established, the protocol layer's vulnerabilities to erroneous state and addressing information should be evaluated.

If protocol logic fault tolerance is implemented, the impact of its protocol algorithms will also need to be evaluated. This means that any protocol-level mechanism needs to ensure the required agreement on protocol state for integrity and the required replication for availability. Often in two-channel systems, there is a conflicting goal between availability and integrity. Hence, mechanisms to improve protocol integrity may reduce protocol availability; for example, logic to contain errors during start-up may render the protocol unable to start.

A.5.4 LOCAL TRANSMISSION-MONITORING AND SELF-CHECKING SCHEMES.

Network technologies may also implement monitoring or self-checking services to improve fault detection and fault tolerance. As with the guardian action, the effectiveness of such schemes depends on the amount of independence and coverage that can be claimed by the implementation. For example, CAN incorporates an error-checking mechanism that will switch

the network to a passive state if the transmissions of the controller are not suitably acknowledged. Since this is implemented within the same IC as the communications component, the action may be degraded by common-mode failures. In addition, such schemes may introduce potential fault propagation vulnerabilities, as it is possible for a node to transition to the passive state in response to the erroneous negative acknowledgements generated from a faulty node. Such vulnerabilities should be analyzed as the network is evaluated.

Other networks may employ local wrap-back schemes where a node monitors its own transmission via local receivers. Such schemes should be analyzed for vulnerability to Byzantine faults, as a local monitoring circuit may perceive the local wrapped-back signals as good, but receivers at the end of a loaded transmission line may see a degraded or erroneous signal. Hence, the wrap-back signal state may not be representative of the network observed state. Byzantine faults and fault tolerance strategies are discussed in more detail in section 7.8 of reference A-4.

Some networks and protocols implement support for self-checking configurations, allowing multiple network interface circuits to be tightly synchronized and to cross-check each other. An example of such a network is ARINC-659. When evaluating the coverage provided by such schemes, care should be taken to examine where the cross-checking and error-containment voting is performed. In ARINC-659, checking and voting is performed at each receiver, hence full coverage of the entire transmission path is assured. Local checking in ARINC-659 is performed solely to increase network availability, with each network IC enabling and monitoring the transmissions of its other half. As with guardian functions, such cross-enabling schemes should be analyzed to ensure that there is sufficient margin for the enabling and disabling action to ensure transmissions are not truncated to produce potentially Byzantine signals. Similarly, self-checking errors that rely on loop-back monitors may be vulnerable to Byzantine faults as discussed above.

A.5.5 RECONFIGURATION AND DEGRADED OPERATION.

Network technologies may also incorporate mechanisms to implement reconfiguration or continued operation in a degraded mode. For example, some physical layers may incorporate a degraded mode of operation that allows communication to continue even if one-half of a differential communications channel is faulted. If such degraded modes are to be leveraged, then the performance (e.g., BER) of the degraded operation needs to be evaluated to ensure that adequate performance is maintained. The protocol mechanism for the detection and announcement of such degraded operation should also be evaluated to verify that timely and correct diagnosis is provided.

Other protocols, such as IEEE 1394, may reroute the network path to mitigate physical or node paths. If such protocol action is to be leveraged by a system, then mechanisms used to implement such actions will need to be evaluated to ensure that the reconfiguration time is suitably bounded. The issues surrounding the erroneous invocation of such logic must also be considered. The recovery mechanisms for such logic should also be investigated to ensure nodes are not permanently isolated in response to local transient errors.

A.5.6 LATENT FAILURE DETECTION.

Fault detection, isolation, and recovery functions used within aviation digital electronics systems are often required to be periodically tested to ensure that the detection and recovery actions remain active. Such covering functions are usually transparent to normal mode operations, hence, without test, it is possible that such functions may fail passively and the protection will be lost. Network fault detection and covering functions are no different; therefore, network mechanisms to assist the latent fault detection should be considered as the network technology is evaluated. To illustrate common network vulnerabilities to latent failure, consider the following scenarios: the short-circuit of intermediate-stage guardian function and if the network traffic can propagate through the shorted guardian without error then the passive state of the guardian enforcement action may pass unnoticed, leaving the system vulnerable to a second uncontained failure of another network component. Similarly, consider a network component with a "stuck at good" CRC calculation circuit; i.e., all data received results in a good CRC, unless such functions are tested. It will be difficult to detect such a state in normal protocol operation since all CRCs are nominally good.

Network mechanisms that incorporate modes and mechanisms to assist the latent fault detection of network components may be preferred. However, such mechanisms should be analyzed to ensure that they do not introduce failure vulnerabilities, because testing for latent failure may disrupt nominal network performance. Interlocks and protection mechanisms should also be evaluated to ensure that such testing occurs only in safe system states. The coverage of the network test procedures should also be evaluated to verify that all key network mechanisms are suitably verified. For complex error detection and enforcement schemes (for example, protocol semantic correctness enforcement), the ability to achieve adequate coverage via the self-test mechanism may be challenging, since such coverage will require all decisions causal to the enforcement actions to be suitably exercised.

A.5.7 VOTING, SELECTION, OR AGREEMENT SERVICES AND REDUNDANCY MANAGEMENT.

Networks may also incorporate redundancy management and voting mechanisms to simplify application-level fault tolerance. The self-checking configuration, discussed in section 7.4 of reference A-4, is an example of such a scheme where increased network component redundancy is leveraged to achieve increased network integrity and availability. In self-checking configuration, a pair works and sends out messages in coordination (that is, at the same point in time). Depending on the required availability targets, self-checking may need two or more self-checking pairs.

Another form of network redundancy is active replication in a triple modular redundancy (TMR) voting scheme. In contrast to self-checking configurations where messages are sent at the same point in time for a pair, nodes always send out the data at different points in time in a TMR scheme. Thus, TMR implements a type of temporal redundancy. In TMR schemes, end nodes need to correlate messages sent at different times before being able to vote, while in self-checking pair configurations, nodes can take the first valid message with integrity (messages that agree and stem from two halves of pair).

Network selection should consider what active replication scheme fits its needs best. Self-checking pair schemes may require special hardware support for synchronized sending of messages, but simplify voting schemes to become a "pick-first valid" message. On the other side, TMR-based systems may not require additional hardware but require, message management (storing) for the messages received at different times from different hosts before voting as well as a voting function implemented at each end node.

Dual replication can either be targeted at ensuring availability or integrity. That is, the replication ensures continuous service despite a (single) failure. The integrity of the value provided is equal to the source integrity of the node and, of course, the communication integrity. On the other side, if replication targets integrity, the end node would perform a comparison of two values. If they agree, the integrity of the data is ensured; if they do not agree, this is a signal to the application, and the integrity is not lost. Yet, the availability achieved is similar to the availability of either component and, of course, the availability of the communication path. Such voting algorithms supplied by the network should be compared with the assumption of the application to avoid unsafe operation; e.g., in ARINC 664, the redundancy management layer chooses the first syntactically correct frame and is targeting availability, assuming that any failure on the communication path is detected by in-line integrity mechanisms (like CRCs). The first syntactically correct frame is of the integrity of the communication source. Any fault defeating the integrity mechanism leading to an undetected error in a dual-replicated, pick-first valid scheme may impact integrity of the data. Masquerading faults are faults where a faulty node pretends to be another node. Masquerading faults can defeat any redundancy management, because multiple inputs to any voting or selection functions may stem from the same fault zone. Network implementations and mechanism should be analyzed with respect to masquerading fault vulnerabilities.

The network technology may also support mechanisms to implement passive replication strategies, for example, the capability of redundant or replicated nodes to share the same network transmission slot. The replicated or redundant nodes take over when the first replica ceases control. In such active and shadow schemes, consideration should be given to the time it takes to detect the failure of the components; e.g., in the case of a failure during sending of a message monitor by another component, the coverage scheme may only detect the failure after it has already been (partially) sent. Thus, the receiving node may have to wait for the next message that can be sent. In addition, the network Master-shadow mechanism should be evaluated for its ability to hand over control in a fault scenario. The effectiveness of release of control mostly depends on guardians or coverage schemes deployed.

A.5.8 BYZANTINE FAULT TOLERANCE.

The Byzantine failure scenario or Byzantine generals' problem (BGP) was first presented nearly 20 years ago. Since its introduction, it has become the subject of a great many papers and scrutiny by the fault tolerance community. Numerous Byzantine-tolerant algorithms and architectures have been presented in the subsequent two decades. With the ever-increasing dependency on electronic hardware and software to perform safety-critical control functions and the emerging trend to implement control with distributed multiprocessor systems, where consensus may be a prerequisite, the practical issues relating to Byzantine behavior need to be understood. For these types of systems, existence of Byzantine fault tolerance is a litmus test for

dependable systems design. However, the wide-scale industrial acceptance of the problem is yet to find maturity. Only recently has SOS faults, a subset of the Byzantine fault class, received some widespread attention. Today, there are still many misconceptions relating to Byzantine failure, both with respect to what makes a system vulnerable, and the very nature and reality of Byzantine faults. This handbook revisits the Byzantine problem from a practitioner's perspective. It is the intention to provide the reader with a working appreciation of the Byzantine failure, from a practical, as well as a theoretical perspective. A discussion of typical circuit-centric failures and the difficulties in preventing the associated failure propagation is presented. These will be illustrated with real-world Byzantine failure observations. Finally, various solutions to the Byzantine problem are presented and discussed within a context of the viability of their industrial deployment.

A Byzantine fault is any fault that produces different symptoms for different observers. This can happen at any point where a signal splits; i.e., one source goes to more than one destination. Byzantine faults are a lot like metastability in that there is no way to prevent them; you can only treat the symptoms so the faults do not become system failures.

Byzantine faults can happen in the amplitude domain. For example, assume that a digital driver gets stuck at 1/2. Because of manufacturing tolerances, other digital circuits using this value may assume it is a 0 or may assume a 1. The most common fault of all (an open) into a complementary metal-oxide semiconductor input looks like a 1/2. Byzantine faults can happen in the time domain. For example, in a synchronous redundant system, no matter how tightly you synchronized the redundant channels, there will always be some (infinitesimally small) time skew between the channels. An input that goes to multiple channels can arrive at a clock tick and within the skew so some channels will see the input arriving before the clock tick and some see it arriving after the clock tick. If the redundant channels vote on the input's value at the clock tick, some will use the old value and some will use the new value. Note that the voters will say some of the channels are faulty even though no hardware fault occurred. This is a design-induced Byzantine fault.

A BGP is a system failure caused by a Byzantine fault. If the multiple observers do not require any mutual coordination, a BGP cannot occur. But, if the observers have to coordinate in some way, or if their actions are compared (by voting or some other means) for fault tolerance, then a BGP is possible.

Byzantine-fault propagation escapes most of the classical fault containment techniques. Solutions to the BGP are well known, but do require a large amount of communication bandwidth. It has been proven that to tolerate F Byzantine faults, you need 3F+1 fault containment zones. From this, one can deduce the surprising result that a simple triple-channel system cannot tolerate even one Byzantine fault, no matter how cleverly it is designed. The next surprising result is that to be fully tolerant to two faults, you need seven fault containment zones.

To further illustrate the Byzantine propagation capability, one can envision a "Schrödinger's CRC," similar to the Copenhagen misinterpretation of "Schrödinger's Cat," where the CRC is simultaneously correct for any interpretation of Byzantine data. The behavior of a 1/2 bit on a CCITT-8 CRC circuit is shown in figure A-3. This figure shows 8 data bits followed by the 8

CCITT-8 CRC bits with one of the data bits to be transmitted stuck at 1/2. Because the transmitter's CRC is a linear (XOR) combination of its data bits, each CRC bit affect by the 1/2 data bit can also be 1/2. The switching threshold voltages are shown for two receivers (***a*** and ***b***). The resulting data received by ***a*** and ***b*** are different, but each copy has a correct CRC for its data. Thus, CRCs can provide no guarantee of protection against Byzantine fault propagation.

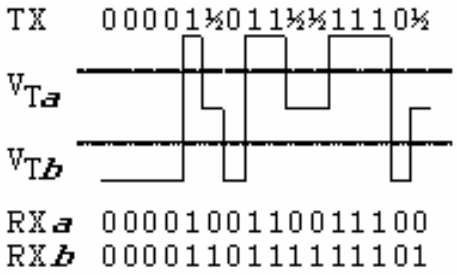

Figure A-3. A Schrödinger's CRC

Interactive consistency of messages is a service provided by a data network to ensure consistent message reception in the presence of Byzantine failures. Byzantine failures manifest as different nodes having a different view of the messages communicated (either no message at all or even different values). It is not possible to diagnose Byzantine faulty nodes unless source coverage (aiming at fault containment) is provided.

Any voting mechanism at end nodes may have a different input set (due to one Byzantine fault, resulting in one different value, or due to a node-local transient fault, resulting in possible faulty but normally detected value). Any voting scheme (e.g., TMR or any selection logic) may select or vote and result in a different value (but possibly a value stemming from a correct node). Such effects may need to be considered by applications for evaluation.

Even if source coverage (e.g., self-checking sources) is used, any voting or selection scheme may still result in a different selection outcome. Similarly, master and shadow implementations at a network level may send out a frame that is corrupted either by a faulty node or transient faults. Switchover from master to shadow may be problematic in arbitrary fault scenarios, because some nodes may correctly receive the master's frame and some may not. If the shadow node correctly receives the master's frame, it may never be able to take over control. Such scenarios should be evaluated.

Coding techniques (such as the use of CRCs or cryptographic signatures) are sometimes proposed as a solution to the Byzantine problem. These solutions assume that certain fault behaviors cannot occur. But, they offer no supportable rationale or enforcement mechanisms to ensure that these fault behaviors do not occur. Until such supportable rationale or enforcement mechanisms are made available, these solutions have no value.

Networks requiring Byzantine tolerance need to mitigate Byzantine failures by incorporating Byzantine-filtering actions. The Byzantine filter transforms Byzantine input signals to consistent erroneous or correct signals. In such systems, the coverage of the Byzantine filter action is a

critical network parameter. In addition, networks are required to incorporate classical Byzantine agreement protocols.

Network evaluations should consider the need for topological support or fault containment mechanism to achieve consistent message delivery.

A.6 DESIGN ASSURANCE.

A.6.1 DEVELOPMENT PROCESSES.

Often network technology forms the backbone of the system architecture. The design correctness of the network implementation is, therefore, of utmost importance, as the network provides a significant common-mode failure vulnerability. With the increasing complexity of network technology, the design correctness problem is increasing with every generation of silicon. Although multiple independent lanes of redundancy may be suitable to mitigate random component failure, if common network technology is used across all lanes, then the system is vulnerable to generic design defects of the network technology implementation. Dissimilar network redundancy schemes may be deployed to mitigate such issues; however, such architectural strategies are beyond this evaluation. Therefore, it is preferable if the network technology was designed with best-practice design procedures. Within the aviation digital electronics domain, this would correspond to RTCA DO-178B for software-related network components and DO-254 for hardware components. Network technologies that have such formal design assurance artifacts will pose less certification risk than other technologies and may be preferred for that reason. Technologies without formal design assurance processes will need to be considered on a case-by-case basis. The complexity and degree of commercial use of the networking technology will then need to be considered. The COTS provisions within DO-254 were designed to handle hardware technologies that have been used in many systems that have accumulated a huge service experience basis. This use experience may be leveraged to assist the design assurance case. This treatment is commonly applied to microprocessors. The applicability of such techniques to networking-related hardware will need to be considered as the network is evaluated.

A.6.2 AVAILABILITY OF STANDARDS AND CONFORMANCE EVIDENCE.

A.6.2.1 Open Specification and Standardization.

The use of open specifications and standardization might assist a certification authority in establishing the acceptability of a network. Irrespective of the formality of the design artifacts, the quality of the network technology specification is a key attribute of the network technology. It is preferable if the technology is open with a standardized and published specification, as this will enable the protocol mechanisms to be analyzed and discussed within the academic and industrial community, including the application for formal verification studies. The standardization process itself is beneficial, as the committee activity usually associated with the open standardization process may also lead to an open, detailed examination of the network behaviors. However, care is required for network technology that is not designed specifically for use in a safety-relevant environment. The completeness of the specification will need to be carefully reviewed. Often, such standards may specify the normal mode of operation only, the

protocol actions-to-erroneous behavior and the associated degraded modes of operation may not be sufficiently treated in the standard document. The evaluation of the network specification should include such completeness analysis.

Another area where specification completeness may be lacking for COTS protocols is in the area of implementation choices that have been made below the protocol specification. COTS solutions may not be not fully described because of the need to maintain competitive advantages between vendors. Hence, many key implementation choices may not be visible and this may impact assurance process where a detailed understanding and analysis of the interactions of all technology layers is required. The availability of suitable design information should be considered as the network technology is evaluated.

A.6.2.2 Conformance and Interoperability Testing.

As with the specification, the availability of standard conformance test campaigns and specifications may also be advantageous. This is especially important for network technology that is sourced from multiple vendors, since it may assist in identifying interoperability glitches. The issues raised above, in relation to specification completeness, also arise in relation to the completeness of the conformance test campaigns; i.e., Are all operating modes covered, and are exception and error reactions sufficiently traveled?

A.6.2.3 Protocol Design Correctness.

In addition to completeness, the correctness of the specification is obviously important. The use of formal methods and development of formal proof arguments for protocol algorithms show much promise here, as they can exhaustively verify the algorithmic behavior. However, when reviewing such formal verifications, the assumptions that underpin the formal proofs need to be fully understood and evaluated against the real-world failure expectations and behavior. Similarly, the composability of the formal verifications needs to be understood to ensure interactions between different protocol algorithms (for example, membership services and clock synchronization). In some protocols, for example TTP/C, interdependencies exist that may need to be evaluated with the formal arguments. That said, formal verification of protocol algorithms can increase design-correctness confidence and, therefore, network technology that has such verification evidence may be more attractive.

Informal validations (for example, random fault injections campaigns) may also increase confidence in the network architecture. However, the conclusions that can be drawn from the fault-injection campaign need to be carefully scrutinized with a detailed understanding of the effects of the fault injection in relation to the technology implementation. For example, consider a heavy-ion fault injection on a communications controller with and without parity on its microsequencer memory. Without parity, this technique may provide useful insight into the performance of the system architecture. This was demonstrated during the TTP/C FIT research program that illustrated the architectural vulnerabilities to Byzantine errors (in this instance caused by bit flips in the instruction memory that resulted in slight deviations of transmission time). However, the same campaign performed on a controller incorporating parity for all onboard random access memory locations may not have been so revealing, as the parity

detection mechanism may swamp the observations with parity-induced fail-stops that cover up the other design weaknesses. This type of study would be less relevant in finding these other architecture and design weaknesses. The issues relating to the design and implementation visibility of COTS technologies are reiterated here, as this visibility may be required to draw any architectural inference from these studies.

A.6.3 DESIGN MARGIN.

The issues discussed in sections A.1 and A.2 also require some design assurances so adequate design and safety margins are established for the selected network technology. Such a design needs to be established and justified to be valid over the whole system lifetime, addressing parasitic and parametric shifts due to temperature effects, etc. This safety margin evaluation needs to be established in several domains, such as the time and value domains of signals under worst-case design parameters and network loading.

For physical layer attributes, this means that influencing factors need to be analyzed with respect to their margin and contribution to the safety margin. Such physical layer attributes may include an oversampling margin that should include the transceiver skew over the lifetime of the product, assuming worst-case loading, aging of components (e.g., clock stability overtime), temperature range of environment, etc.

A.6.4 CONFIGURATION TABLE CORRECTNESS AND PERFORMANCE JUSTIFICATION.

In addition to the design correctness of the network implementation, the design correctness of network configuration parameters and tables is also required. This is especially important if the table parameterization impacts protocol algorithmic-level behavior, for example, clock synchronization timing and propagation delay parameterization. In such instances, the parameters may severely impact protocol performance. The incorrect configuration of such parameters may, therefore, invalidate any formal proofs of algorithmic correctness. Similarly, tools may be used to establish parameters for network-policing policies, for example, message transmission rate limiting and maximum message jitter. In such cases, the correctness of these parameters may severely impact network performance assumptions. Therefore, when evaluating a network for suitability, consideration should be given to the rigor applied to ensuring correct network configuration parameters. Ideally, all parameters critical to network operation will have explicit formal requirements and invariants that are traceable to network functional behavior, assumptions, and requirements. Such traceability may assist the completeness checking of the guidance presented. Ideally, the guidance supplied will be suitably assured for correctness and completeness.

The network technology may also provide tooling to assist network configuration and its associated verification. Such tools are often required to handle the size and complexity of modern networking technologies and to assist with the generation of nonhuman readable binary configuration tables. If tooling is used for configuration data generation or verification, then the development pedigree of the tooling may also need to be examined as the network technology suitability is evaluated. If the tooling is in-line, i.e., the tooling generates protocol configuration parameters that are not verified by subsequent process checks, then the generation tooling should

be qualified in accordance with the DO-178B guidelines for development tools. Alternatively, if the tooling is simply used to verify the network configuration parameters, then they are less stringent and DO-178B verification tool guidance should be adopted. The data flow path of in-line generation and verification tooling need to be evaluated to ensure that adequate independence exists within the tool chain to prevent a common tooling failure. In the ideal, the configuration inspection tools will be driven from reviewed network-related functional data flow requirements and the formal network parameter constraints and invariants.

For some modern asynchronous networks, for example ARINC 664, the size and scale of the configuration problem is very large and end-to-end performance (e.g., data flow latency and jitter) is difficult to analyze and bound. The sheer complexity of the network level interactions between end-node behavior, switch implementation, and the chosen network policing policies (e.g., message rate limiting) may greatly complicate network performance justification. However, procedures or tooling to analytically bound and justify the worst-case behavior of such networks is required to meet certification requirements. Therefore, the capability and maturity of available analysis tooling should be given careful consideration, as such networks are evaluated. Similarly, network technologies that incorporate complicated MAC interactions may also complicate end-to-end performance calculations. Such interactions and any associated network logic (e.g., retry logic and queuing mechanisms) need also to be considered by performance calculations and associated tooling. Similarly, erroneous node behavior and associated diagnosis latencies should also be considered to bound the influence of the faulty node behavior on network performance. Networks that bound such influences may therefore be preferable, as they may greatly simplify performance justification calculations

For highly integrated multivendor systems network technologies that incorporate tooling to assist the incremental change of the network tables, allowing new functions and their associated data paths to be added to the network with minimal impact on previously analyzed functions may also be attractive, since such tooling may ease incremental certification effort.

A.6.5 NETWORK MONITORING AND TEST EQUIPMENT.

With complexity of modern network technology, the ability to monitor and observe network behavior is very important to support design validation. Similarly, the ability to insert faults into the different network layers may be required to test the network redundancy management mechanisms, or the fault response behavior of applications operating on top of the network infrastructure. Therefore, the availability and capability of the test equipment that exists for the network technology may also be a very important consideration. In the ideal situation, such test equipment is able to observe all behavior of all network nodes, including network start-up and recovery actions. The portability of the test equipment should also be considered, as such equipment is often required to support flight-testing.

The no-interference guarantees of the test equipment may also need to be evaluated if it is to be deployed in a flight test scenario. The ability to monitor the entire network behavior from limited test inspection access points should also be considered. In some modern switched technologies, such access is more difficult than in simpler buses. Hence, work in some newer switched technologies is being performed to develop the network-wide controllability and observability needed to test the maintenance of these new or more complicated networks, while

at the same time, trying to minimize the invasiveness and logistics complexity of connecting test equipment to these networks.

A.7 REFERENCES.

A-1. Nyquist, H., "Certain Topics in Telegraph Transmission Theory," *Trans. AIEE*, Vol. 47, April 1928, pp. 617-644, Reprinted as a classic paper in: *Proc. IEEE*, Vol. 90, No. 2, February 2002.

A-2. Shannon, C.E., "Communication in the Presence of Noise," *Proc. Institute of Radio Engineers,* Vol. 37, No. 1, January 1949, pp. 10-21, Reprinted as a classic paper in: *Proc. IEEE,* Vol. 86, No. 2, February 1998.

A-3. United States Government General Services Administration, *Federal Standard 1037C. Telecommunications: Glossary of Telecommunication Terms,* August 7, 1996. http://www.its.bldrdoc.gov/fs-1037/

A-4. Driscoll, K., Hall, B., Koopman, P., Ray, J., and DeWalt, M., "Data Network Evaluation Criteria Report," FAA report DOT/FAA/AR-09/27, 2009.