ISSN 0280-5316
ISRN LUTFD2/TFRT--5782--SE

Safety-Critical Communication in Avionics

Dan Gunnarsson

Department of Automatic Control
Lund University
December 2006

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
December 2006

Document Number
ISRNLUTFD2/TFRT--5782--SE

Author(s)
Dan Gunnarsson

Supervisor
Kristina Forsberg at Saab Avitronics in Jönköping.
Anton Cervin at Automatic Control in Lund.

Sponsoring organization

Title and subtitle
Safety-Critical Communication in Avionics (Säkerhetskritisk kommunikation för flygsystem)

Abstract
The aircraft of today use electrical fly-by-wire systems for manoeuvring. These safety-critical distributed systems are called flight control systems and put high requirements on the communication networks that interconnect the parts of the systems. Reliability, predictability, flexibility, low weight and cost are important factors that all need to be taken in to consideration when designing a safety-critical communication system.
In this thesis certification issues, requirements in avionics, fault management, protocols and topologies for safety-critical communication systems in avionics are discussed and investigated. The protocols that are investigated in this thesis are: TTP/C, FlexRay and AFDX, as a reference protocol MIL-STD-1553 is used. As reference architecture analogue point-to-point is used. The protocols are described and evaluated regarding features such as services, maturity, supported physical layers and topologies.Pros and cons with each protocol are then illustrated by a theoretical implementation of a flight control system that uses each protocol for the highly critical communication between sensors, actuators and flight computers.The results show that from a theoretical point of view TTP/C could be used as a replacement for a point-to-point flight control system. However, there are a number of issues regarding the physical layer that needs to be examined.
Finally a TTP/C cluster has been implemented and basic functionality tests have been conducted. The plan was to perform tests on delays, start-up time and reintegration time but the time to acquire the proper hardware for these tests exceeded the time for the thesis work.
More advanced testing will be continued here at Saab beyond the time frame of this thesis.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
63

Recipient's notes

Security classification

Acknowledgements

First of all I want to thank my advisor at Saab, Kristina Forsberg. Your friendly attitude, ideas and extensive knowledge in the area of safety-critical systems has made my work interesting and my time here at Saab enjoyable. Without your help this work would not have been possible!

I want to thank my friends and work colleagues at Saab: Stellan Nordenbro and Åke Cederbom. Other people at Saab that has given me useful advice are Jonas Nordqvist, Jonas Dahlqvist, Håkan Forsberg and Anita Karlsson. Per-Olof Bergman has been a very helpful with the numerous modifications and adapters that has been manufactured in order to make a complete setup in the laboratory. Jan-Olof Nilsson has been very helpful providing the excellent photographs used in the report and presentation.

A special gratitude goes out to Thomas Mörth who first negotiated the contact that gave me the opportunity to write this thesis in cooperation with Saab.

I also want to thank my advisor at LTH, Anton Cervin for giving useful comments on my work and for taking care of the administrative parts of the thesis at the university.

Thanks to Saab for sponsoring TTP-Seminar and Workshop in at TTTech in Vienna and for employing me!.

1 Introduction

This thesis has been carried out at Saab Avitronics in Jönköping with the over-all goal to gain knowledge about safety-critical communication systems. In particular physical implementations issues are addressed via practical experience. In this work the safety-critical communication is exemplified with communication within a flight control system.

Fly-by-wire systems are becoming increasingly common in civil transport aircraft due to the economic and technological benefit that the technology provides. These systems are comprised of two major components; the flight control laws, which govern the aircraft's handling characteristics, and the flight control architecture, or the hardware, which is used to implement the control laws. This thesis does only address the latter.

1.1 Background

Over the years new aircraft have brought with them new opportunities to introduce new technologies. The demands for increased functionality conflict with the demand of lower cost hence, the challenge avionics engineers are facing today is to build systems meet this requirement for high functionality and less maintenance, at reduced cost.

Fly-by-wire systems have followed the development exploiting the benefits of the new technology. These systems use electrical signalling to control the control surfaces with the aid of flight control computers that contain control laws. The fact that a large number of subsystems in an aircraft use microprocessors has led to the development of a variety of digital data buses. In an integrated architecture the flight control system takes advantage of these subsystems to perform its tasks.

Most flight control systems are traditionally designed with a central computer that is connected to sensors and actuators using point-to-point connections. Traditional point-to-point communication systems are very reliable since they use a federated architecture where the whole system is physically isolated from other systems. Physical separated subsystems make the system robust to fault propagation, local disturbances, fire etc.

In order to decrease weight and increase flexibility replacing the traditional point-to-point system an integrated system with broadcast bus communication could be an alternative.
The bus communication topology decreases weight but introduced new concerns such as possible failure modes which emphasis the need of fault containment techniques, structured medium access method and communication scheduling. These issues might be handled by a communication protocol and hence, the choice of protocol a key decisions that strongly influences the design.

There are both Time-Triggered (TT) and Event-Triggered (ET) protocols that could be used for broadcast bus communication. Communication in a safety-critical application, such as flight control, must be predictable and analysable. Hence, TT communication out-roles ET communication since it guarantees delays and does not suffer the risk of collisions as ET. This thesis concentrates on protocols using TT medium access; comparing chosen protocol and investigating problems involved in implementation of these in a flight control system.

1.2 Scope and goals

Saab's goal is to gain knowledge of and identify communication protocols for safety-critical systems. From a research project GAST [6] we have received evaluation boards with three time-triggered protocols, TTP/C [1] , FlexRay and TTCAN. Initially the goal of this thesis was to investigate/evaluate these protocols suitability for the avionic industry. Since there are still unsolved issues involved in setting up this hardware we decided not to go further with these boards. Instead we will use a platform for the TTP/C protocol with the main goal for the practical part to investigate the TTP/C protocol's suitability for safety-critical systems.
Protocols besides TTP/C that are included for evaluation are:
FlexRay; has no previous use in avionics applications but has similarities with TTP/C. It is believed to be the replacement for the industry standard protocol CAN in the automotive industry which would mean that it would be widely available and tested.
AFDX (time-triggered Ethernet developed by the aviation industry); could also be a candidate that has targeted the highest criticality level (level A, see Table 2.1) for some applications.

Practical and analytical issues to be investigated are:

- Delays in a cluster of 20 nodes (start-up time, re-synchronisation)
- Physical bus set-up (RS485 connectors, termination issues, cable lengths)
- Physical star/hub set-up (100BASE-TX)
- Pros and cons of bus respectively star (delays, bandwidth, fault modes, weight)
- Fault propagation/isolation (disturbances to one node)
- Reliability of the communication architectures

Problem statement:
The main question that this thesis aims to answer is whether TT broadcast bus communication (exemplified by TTP/C) is suitable for usage in safety-critical communication system in avionics (exemplified by a flight control system).

Before a broadcast communication bus is to be used in a level A system certification issues must be properly addressed as well as acceptance from possible customer (low risk – new technique). This thesis will address both practical and analytical certification issues of the TTP/C protocol. However, far from all aspects are treated – interesting issues that are outside of the scope of this research are presented in chapter 12 (conclusions and future work).

The thesis contains three parts:

- Theory and background in dependability, fault tolerance, certification, requirements and topologies.
- Evaluation of three time-trigged protocols (TTP/C, FlexRay and AFDX) usability for safety-critical communication.
- Practical evaluation of the TTP/C protocol from experiments.

The first two parts gives the reader a basic understanding of the concepts, terminology and a brief introduction to protocols that are currently used or are considered to be used in avionics.

1.3 Thesis outline

Chapter 2 gives a brief background to the certification process and the most important guidelines for the systems discussed in this thesis.
Chapter 3 contains requirements that apply to all airborne electrical safety-critical systems.

Chapter 4 illustrates different fault types and the importance of fault tolerance and error detection.
Chapter 5 describes protocols used in safety-critical communication. STD-MIL-1553, AFDX and FlexRay are described in brief, for comparison, and TTP/C in detail.
Chapter 6 gives an introduction to network topologies with focus on safety-critical communication.
Chapter 7 contains a conceptual study of communication architectures based on the protocols described in Chapter 5.
Chapters 8-12 describe the experiments conducted, analyses and discuss the results and give conclusions of the study.

Appendix A gives the reader an introduction to dependability terminology. It also includes glossary and abbreviations.
Appendix B contains the test cases used to evaluate TTP/C in a laboratory environment.
Appendix C briefly describes the design of a cluster using TTP-Tools and contains data for the example cluster described in 7.2.

2 Avionic standards and regulations

The aviation industry has a long tradition of strict rules and regulations regarding the development and design of aircraft. Over the last decades increasing complexity and integration of electrical systems has created even higher demands for review and guidance to ensure safety and proper operation.[9]
Assurance through certification is generally achieved by a combination of testing, analysis and review of the design and design process.

The authority that does the final certification for civil aircraft is FAA in the United States and EASA (formerly known as JAA) in Europe. The rules and regulations stated by these authorities are almost identical. However, the assessment can vary between the two. FAR and JAR are regulation documents authored by FAA and EASA. They contain rules and regulations how an aircraft should function in order to pass certification and to be allowed to fly.
To aid the certification there are guideline documents that specify how design, implementation and usage of hard- and software should be done to provide a satisfying level of reliability (picture in Figure 2.1). These guidelines are international standards and have to be followed in order to manufacture aircraft. The guidelines contain requirements, in detail, how every part of the systems treated should function and behave under certain conditions.

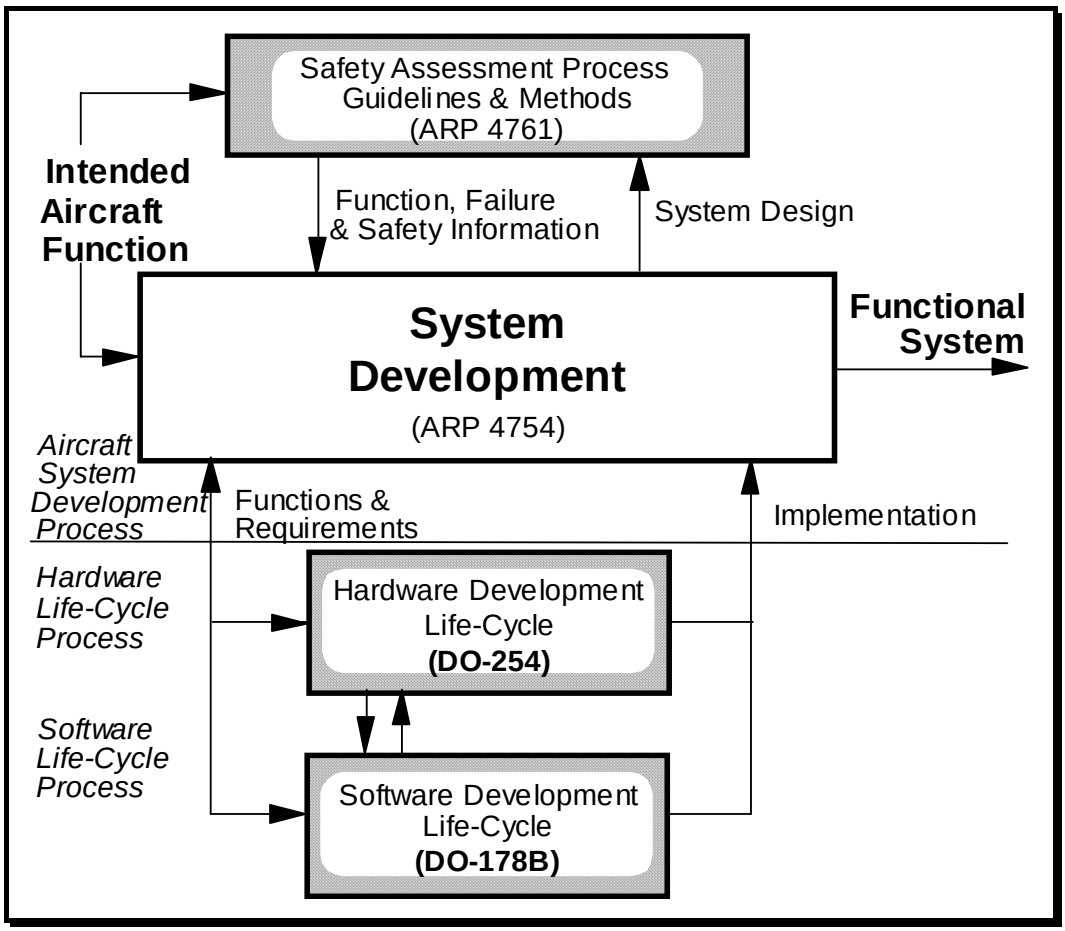


Figure 2.1: Certification guidance documents covering system, safety, hardware and software processes. Figure copied from [10]

Systems discussed in this thesis are safety-critical systems level A and B (See table 2.1). The focus is on communication within these systems and certification guidelines relevant are:

- ARP4754 [10] discusses the certification aspects of *highly-integrated* or *complex* systems installed on aircraft, defining requirements for the overall aircraft operating environment and functions. Highly-integrated system refers to systems that perform multiple aircraft-level functions. Complex refers to systems whose safety cannot be shown solely by test and whose logic is difficult to comprehend without the aid of analytical tools. ARP4754 addresses the total life cycle of systems that implement aircraft functions. It excludes information of detailed system, software and hardware design processes.
- ARP4761 [16] describes the safety-assessment process that include requirements generate and verification. The process provides a methodology to evaluate aircraft functions and to determine associated hazards.
- RTCA/DO-178B [11] deals with software life cycles. It specifies how to manage the design process and how to prove that the output meets the requirements. It is noticeable that RTCA/DO-248B is a clarification document for DO-178B to give a hint of the complexity of this process.
- RTCA/DO-254 [12] deals with hardware design assurance for developing complex hardware for safety-critical applications (see Figure 2.1).
- For environmental requirements the guidance document RTCA/DO-160D Environmental Conditions and Test Procedures for Airborne Equipment.

To classify the criticality of a part of a system five development assurance levels (see Table 2.1) have been introduced.

Table 2.1: System development assurance levels for civil aircraft [9] [11] [12]

System Development Assurance Level	Failure rate (failures/ hour)	Failure condition classification	Failure condition description
A	$\lambda <$ 10E-9	Catastrophic	Failure conditions that would prevent continued safe flight and landing.
B	10E-9 $< \lambda <$ 10E-7	Hazardous / Severe-Major	Failure conditions that would reduce the capability of the aircraft or the ability of the flight crew to cope with adverse operating conditions to the extent that there would be: a large reduction in safety margins or functional capabilities, physical distress or higher workload such that the flight crew could not be relied on to perform their tasks accurately or completely, or adverse effects on occupants including serious or potentially fatal injuries to a small number of those occupants.
C	10E-7 $< \lambda <$ 10E-5	Major	Failure conditions which would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be, for example, a significant reduction in safety margins or functional capabilities, a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to occupants, possibly including injuries.
D	10E-5 $< \lambda <$ 1	Minor	Failure conditions which would not significantly reduce aircraft safety, and which would involve crew actions that are well within their capabilities. Minor failure conditions may include, for example, a slight reduction in safety margins or functional capabilities, a slight increase in crew workload, such as, routine flight plan changes, or some inconvenience to occupants.
E	Any range	No safety effect	Failure conditions that do not affect the operational capability of the aircraft or increase crew workload.

3 Requirements on communication networks in avionics

This chapter contains requirements that are placed upon systems and components that are to be used in an aircraft. Note that many of the requirements stated below apply to a lot more than communication networks while some are more specific. The purpose of these requirements are to provide the certification authorities with proof that the each system and component put in an aircraft functions together and that the aircraft is airworthy and should be allowed to fly.

3.1 Functional requirements

According to [10] functional requirements are those necessary to obtain the desired performance of a system under the conditions specified. They are a combination of customer desires, operational constraints, regulatory restrictions and implementation realities. These requirements define all significant aspects of the system under consideration. Regardless of the original source, all functions should be evaluated for their safety related attributes.
Additionally there are several subcategories of functional requirements to consider. Customer requirement that vary with the type of aircraft and the usage intended such as route system. Operational requirements define the interface between i.e. flight crew and each functional system. Performance requirements define attributes that make it useful for the customer such as speed, accuracy and response time. Factors that affect performance requirements of a communication network are e.g. bandwidth, through put, number of nodes, overhead, jitter, delays and response time. Physical and installation requirements define attributes such as size, power, cooling and handling. Maintainability requirements include scheduled and unscheduled maintenance of equipment.

Interface requirements include the physical system and interconnections between subsystems.
For the systems discussed in this thesis many functional requirements such as usability have no importance since the purpose is to evaluate time-triggered broadcast buses. Focus is put upon the technology itself and what constraints and features it has and whether it is suitable for use in airborne systems or not. Some aspects of functional requirements are treated in chapter 7 and 10.

3.2 Environmental requirements

Environmental requirements state that equipment shall function normally and without degradation under the environmental conditions experienced by the equipment and aircraft through its service life unless otherwise stated in the specification. Temperature requirements include normal and severe conditions both on ground and when airborne. All equipment is covered in powered and unpowered condition. Vibration requirements state what kind of vibration that should be handled by all equipment.
Electromagnetic Compability (EMC) environmental requirements deal with Lightning effects; direct and indirect, high intensity radiated fields, EMC environment generated by on board equipment and Electrostatic discharge (ESD). Tests with transients and slow waveforms can be used to show that systems and item do not malfunction under the EMC conditions specified.

Environmental requirements are of big importance when designing a safety-critical communication system. A bus or a communication link that extends into the wings of an

aircraft is exposed to significant electromagnetic fields, temperatures and vibrations that might degrade the performance as well as the reliability. A big issue when introducing a "new" digital communication concept, such as TTP/C, is the amount of electronics needed out by each actuator where the environment can be very hostile. A big challenge is to prove that the equipments can survive in this hostile environment. Additional difficulties are to expect regarding temperature and EMI (Electromagnetic Interference) with the new lighter wings of composite material. Cable lengths are a concern due to signal distortion that increases with cable length. RTCA/DO-160D deals with environmental issues in airborne systems.

3.3 Test requirements

Testing is a verification method that provides repeatable evidence of correctness. There are two main objectives of testing.
- To show that a system performs its intended functions. Testing of an intended function involves evaluation against pass/fail criteria based on the system requirements.
- To show that a system do not perform unintended functions that affect safety. Ad hoc testing may be used to identify unintended system operation. Note that it is not possible to prove the complete absence of unintended function by test. According to DO-254 (certification issues) specific mitigation techniques are required for level A and B complex hardware assurance. The hardware is not complex if fully analyzable or testable.

Tests are conducted on all parts of a system. Documentation of the tests is crucial so a second party, i.e. the certification authorities, can reproduce and verify the test results.
For each test the required inputs, actions required and expected results with tolerance should be specified. Test result data should at least contain version of test and item being tested, version of tools and test equipment, test results, deviation from expected values (if any) and conclusions containing success or failure or the testing process.
Tests are often associated with the development phase of a system. Equally important are build in self-tests that are carried out every time the system is powered up and continuous self-tests during operation to detect and handle errors.

3.4 Safety requirements

Safety requirement include demands on that all unexpected events that can be identified as significant in a safety and reliability point of view must be handled. Each function is analysed and given a maximum failure rate per flight hour depending on its criticality. This is called quantitative safety requirements. Qualitative safety requirements include that no single point failure should lead to catastrophic or hazardous failure conditions.

In general when a new technique is introduced in the aviation industry a backup system is required for the technique to be proven reliable, this is also often the case when using an already proven technique. The possibility of design faults (described in section 4.1.1) must be assessed by service history, safety specific analysis or diversity in hard- and software.
The most common fault hypothesis is that no single point of failure is allowed to affect the performance of the system. Techniques to achieve this are to use redundancy both in software and hardware. Fault containment zones are also important to keep errors from propagating in the system.

A basic problem that is evident when it comes to a communication bus is that when nodes are connected to each other it is possible that a short circuit could prevent communication or destroy nodes connected to the bus. The situation is slightly different if a central hub or star is

used in star topology where the star can filter out errors. However, if the star coupler breaks all communication will be down.
The vicinity problem is another aspect that must be considered. In case of a fire a whole area of the aircraft might be destroyed. Even if the system has been designed with the most critical part replicated to tolerate a single point of failure this might destroy all replicas. Hence, it would be necessary to put the replicated nodes in different zones of the aircraft to create physical independence.

3.5 Broadcast bus requirements

This section contains a compilation of requirements that are especially important for a data bus that is to be used in avionics systems.

1. The bus shall provide bounded (predictable) latency.
2. Adequate bandwidth for the application should be provided by the bus. It is common to have at least 50% of spare bandwidth for future reconfigurations.
3. The probability of lost messages should be consistent with the probability classification of minor (see Table 2.1) under the interference of the specified environment.
4. The probability of undetected corrupted data should be consistent with the probability classification of hazardous (see Table 2.1), the probability of successive occurrences should be consistent with the probability classification catastrophic under the interference of the specified environment.
5. The bus standard should support broadcast or multicast messages.
6. The bus should provide fault isolation mechanisms that provide controlled access to the bus. It is also desirable that the physical bus driver (controller) has a low failure rate.
7. The bus should use components that remain in the market for at least 10 years.
8. The physical layer should allow communication on the bus even if a node is removed.
9. The physical layer should be able to accommodate at least 30 nodes and have a bus length of at least 100m.
10. The physical layer should be able to use transformer couplers to achieve galvanic isolation at each node.

4 Fault Management

There are two approaches to achieve reliability in a system. Fault-avoidance aims at preventing faults of occurring in the first place. This approach is implemented both in the design phase and in the planning of service and maintenance where components are replaced with certain time intervals to avoid faults. The second approach is to design the system with fault tolerance mechanisms such as redundancy so if one part fails another part can resume its function. The change in the system can either use a mechanism where the fault is recognized and some defined action can be taken or the fault can be masked using replicated hardware.

Section 4.1 describes fault types that occur in electrical systems. Mechanisms that are suitable to handle the specific fault types are briefly discussed. Section 4.2 describes fault tolerance and how to achieve it using error detection and fault tolerance mechanisms and Section 4.3 summarize fault management of different fault types

4.1 Fault types

Faults consist of a broad spectrum of events or states. There are several ways to divide or cluster different kind of faults. Common labels of faults are shown in Figure 4.1 below.

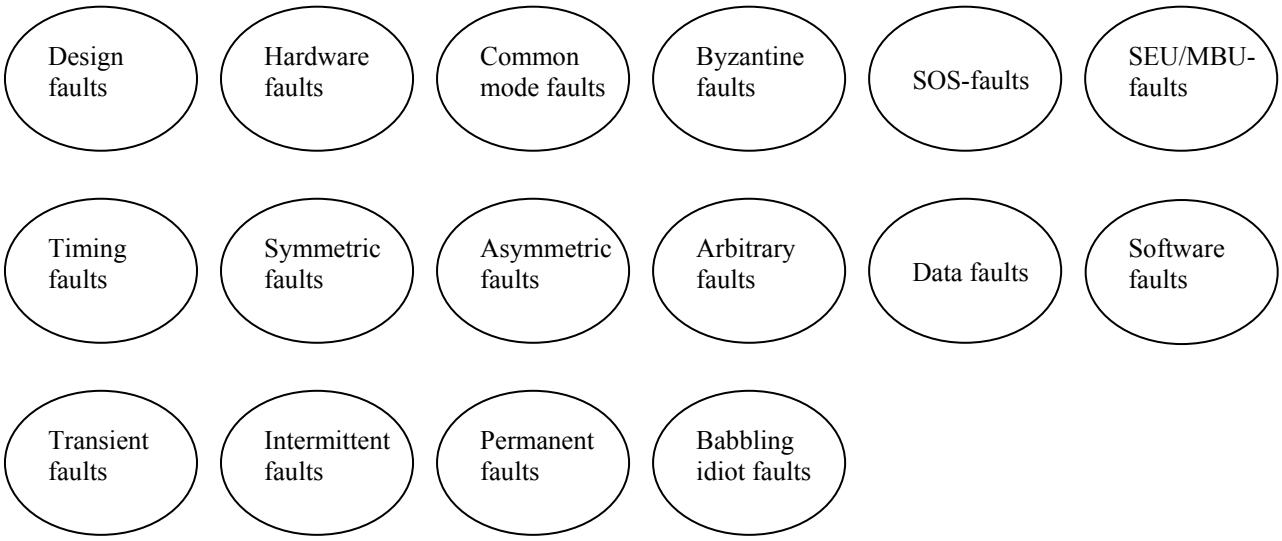


Figure 4.1: Fault types

This section gives a short description of these different faults and their appearance which is essential in order to chose appropriate and efficient fault tolerance mechanisms.

4.1.1 Design faults

Design faults are, as the name says, faults that are introduced in the design of the system, in software or in hardware. If a component contains a design fault then all copies or replicas of this component contains the same fault. Design faults are a subset of common mode faults. During the design process, fault prevention aims at preventing faults; especially design faults, from occurring and being introduced. Design faults must be proven to not exist or to be tolerated by e.g. diversity in components, software, specification etc.

Mechanisms to detect or avoid design faults are design reviews, rigid development processes, simulations, use of formal methods and testing (independent of design).

Hardware design faults can be mitigated through "service history" i.e. the complex hardware device can be proven to be free from design faults if it has been used successfully earlier or by safety specific analysis (e.g. element analysis, formal methods). Hardware design faults can be tolerated by hardware diversity. However software design faults are quite different and are best addressed with fault prevention. Software diversity adds cost and complexity and is not sufficient for tolerating "bugs" [19] .

4.1.2 Hardware and software faults

All software faults are design faults; all other fault types discussed here (in Figure 4.1) are hardware faults. Safety analysis techniques such as fault trees, Failure Mode Effect Analysis (FMEA), reliability prediction (estimated failure rates) are all related to hardware faults.

Hardware faults cause errors in the software, e.g. "stuck-high or stuck-low" faults in interfaces or memory cells cause *data faults*. These faults are tolerated using data redundancy (there are more than one source to the information). Detection and localisation is done through monitoring and/or self-tests.

4.1.3 Common mode faults

In general, a common mode fault occurs when two or more identical modules are affected exactly in the same way by a fault, at the same time. Common mode faults address the whole lifecycle of a system, developing through maintenance and service.
Mechanisms to detect these kinds of faults are common mode analysis. The common mode analysis researches systematically all causes, which can impair the independence, or segregation requirements of the design. The CMA process is described in [16] where the following common mode sources/failures are listed:

- Software development errors
- Hardware development errors
- Hardware failures
- Production/repair flaw
- Stress related events (e.g., abnormal flight conditions, abnormal system configurations)
- Installation errors
- Requirement errors
- Environmental factors (e.g., temperature, vibration, humidity, etc.)
- Cascading faults
- Common external source faults

Diversity is a way to avoid some common mode faults. In practise redundant system functions are implemented using different software and different hardware. Which, however, is a very expensive and strenuous approach since components have different life cycles, hence service and replacement might be hard to coordinate.

4.1.4 Specific fault types

This section describes Byzantine faults, SOS faults, Babbling idiot faults and Timing faults.

A *Byzantine fault* is when a component behaves in an arbitrary way. It might even send different information to different components.

Definitions of a Byzantine fault and a Byzantine failure are [4] :

> **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.

According to [4] , [13] for a system to exhibit a Byzantine failure there must be a system-level requirement for consensus. If there is no consensus requirement, a Byzantine fault will not result in a Byzantine failure. A class of systems that exhibit this requirement strongly are time-triggered systems where the failure of the global clock will lead to system failure.
This also applies to most asynchronous approaches as well since a coordinated system action will require consensus. Redundancy, which is widely used in safety-critical systems, is nearly impossible to create without consensus.

One possible Byzantine fault is when a digital signal is in between the voltage thresholds for a "0" and a "1" and may be interpreted differently by receiving nodes, see Figure 4.2. This kind of signal is called a 1/2 and can be the source of a fault that can propagate through several parts of a system since most systems allow all voltages that are within, for the system, specified range.

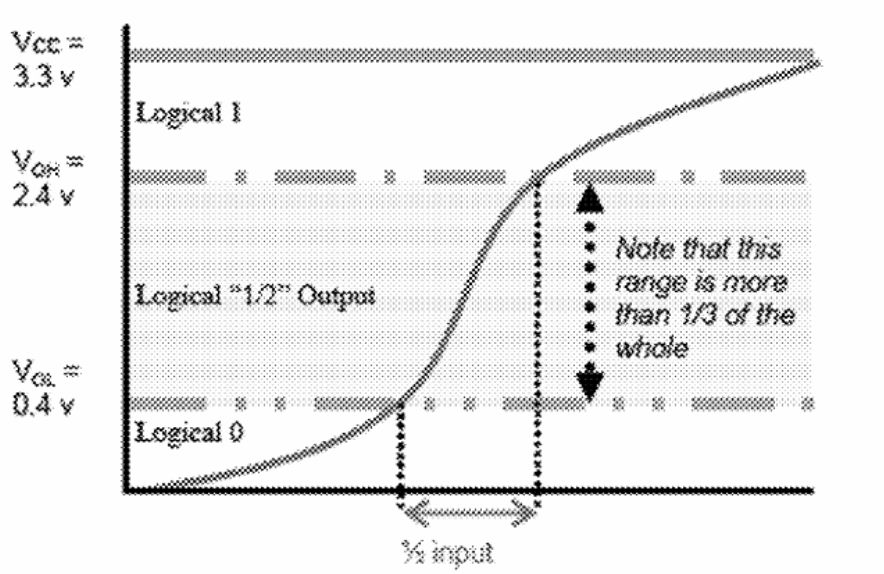


Figure 4.2: Gate transfer function with 1/2 area defined. Figure copied from [4]

Slightly off specification (SOS) faults, these are faults where components deviate from their specification in e.g. jitter and voltage range. SOS faults can appear as Byzantine faults. An example of a SOS-fault is if there is a corruption in a nodes time base that leads it to send messages at periods that are slightly outside specification, slightly too early or too late. Failure modes like clique formation might occur when this happens if receiving nodes with somewhat fast or slow clock (but within specification) would accept these messages but other correct nodes would not, which could create a disagreement whether to accept the sender as a functional or disregard it as a dysfunctional node.

These kinds of faults are very hard to detect and sometimes to account for in design. Creating fault containment regions in design is however the only way to prevent errors from propagating in an uncontrolled way.

Babbling idiot fault refer to a node which is sending/talking in an uncontrolled way. This is particular dangerous in a bus topology, where such a fault is a single point of failure for the bus. These faults are prevented with bus guardians.

Timing faults appear when the timing requirements are not met. In a communication network this can be due to bottlenecks or jamming in systems using heuristic scheduling techniques or due to insufficient timeslots using TDMA scheduling. Fault prevention (analysis, simulations, testing, experience etc…) is the only way to address these faults.

4.1.5 SEU and MBU

Bit flips due to cosmic radiation are called SEU (Single Event Upset) and MBU (Multiple Bit Upset). Civilian aircraft, cruising at an altitude of ~10 000 m, are very likely to experience several SEU and MBU during flight hence the design must be tolerant to these transient faults.

Design considerations includes: a) limit the use of RAM (EPROM is preferred), b) assess the risk of registers in microprocessors, programmable circuits, etc. For memory protection error detection and auto correction (EDAC) code can be used and also new memories are being developed using a distributed storage of data where MBU only cause *one* bit to flip hence common ECC protection (Error Correction Code for one bit) is sufficient to tolerate both SEU and MBU.

4.1.6 Divided by occurrence

Physical hardware faults can be *permanent*, *intermittent* or *transient*. Permanent faults are stable fault modes that once they occur will stay in that mode for a long time. Transient are fault modes of limited duration in time, caused by temporary malfunction of the system or due to some external interference. An example of how these are related is connection faults because of a poor connection i.e. between connectors. The first time a signal loss in a connection occur it is manifested as a transient fault, as time goes the connection disconnect more frequently which then becomes an intermittent fault, and if the connector is disconnected all time the fault is permanent. Transient faults can not be treated since they, by definition, disappear after a short time. Errors caused by transients are tolerated using time redundancy, filters, or recovery mechanisms. Permanent fault are tolerated by hardware redundancy. Intermittent faults will fall under the transient or the permanent fault treatment.

4.1.7 Divided by appearance

Symmetric, *Asymmetric* , and *Arbitrary* faults. A fault which appear the same to all observers or all nodes in a communication network is called symmetric, while faults that are not symmetric are called asymmetric or arbitrary. Hardware faults, as SOS faults, are “easy” to tolerate if they are symmetric with e.g. TMR (Triple Modular Redundancy, see Figure 4.3) and voting. However hardware faults that are asymmetric, could be SOS faults or Byzantine faults, are more “costly” in terms of replicated units to tolerate. To distinct one arbitrary faulty node requires minimum four nodes. Asymmetric faults should primary be addressed with fault prevention.

4.2 Fault tolerance

When designing a fault-tolerant system an explicit fault hypothesis must be used that describes the number, type and arrival rate of the faults it is intended to tolerate.
Systems that address some kind of dependability use at least the *Single fault hypothesis* in which the designer must ensure that no single point of failure can cause a catastrophic failure. A common approach to prevent this is to use hardware redundancy in i.e. nodes, power supplies and clocks to prevent that a failure in one part affects the performance of other parts of the system.
Since it is absolutely unacceptable for a safety-critical system to experience degradation or loss of service due to a single point of failure all subsystem must be designed to operate in the presence of a single failure. In the following subsection fault tolerance mechanisms and services are described. In order to achieve reliability in a system it is necessary to address fault tolerance at different levels of abstraction.

4.2.1 Redundancy

Static redundancy
In static redundancy (also called active replication) the application is executed at N redundant nodes in parallel and a majority vote is performed to prevent faulty data to propagate to other parts of the system. (See Figure 4.3)

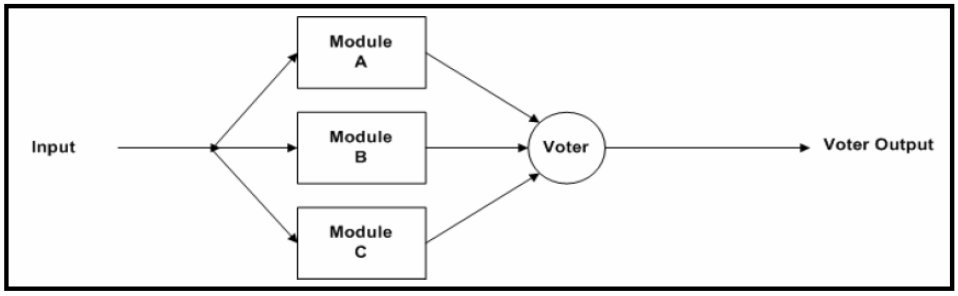


Figure 4.3: Fault masking in a triple redundant system

Dynamic redundancy
In dynamic redundancy (also called passive replication) some nodes are active and some are used as stand-by spares, which are activated in case of failure. The stand-by spares can either be cold (unpowered) or hot (powered). A shadow node is hot but does not deliver anything to the bus except for in case of failure.
A hot spare generally uses less time to replace the faulty node than a cold spare. However, the cold spare generally have lower failure rate.

Replicated communication channels
Replicated communication channels are used to tolerate faults on the communication channels. Faults that can occur are short-circuit, cut off wires and disturbances such as radiation and lightning.

Software and data redundancy
These are systematic error handling functions implemented at system-level and component level. Common techniques are: replicated messages, message checksums, error detection and correction for memory, assertion checks, N-version programming and test programs (i.e. BIT).

4.2.2 Recovery

Roll back recovery
Roll back recovery is the simplest scheme where calculations are allowed to be re-executed in additional time when a failure is detected.

The recovery block approach
Is similar to active and passive replication but uses a simpler backup spare to reduce the time to provide a correct output.

4.2.3 Monitoring and filters

Health monitoring
Monitoring nodes at a higher system level monitors subordinate nodes and performs periodical life sign controls. Through this system it is possible at aircraft level of a system to have complete information regarding the status of the aircraft.

Self-tests
Check of components by running a program with known input and comparing output with known result.

4.3 Fault management summary

Fault types and fault tolerance mechanisms from 4.1 and 4.2 are summarised in Table 4.1. In order to handle an error the system has to be able to detect it. Table 4.1 includes fault types and fault tolerance mechanisms together with several techniques to detect errors. It is common to use more than one of these techniques in a safety-critical system.

Table 4.1: Fault management summary

Fault types	Fault tolerance mechanism	Error detection mechanism
Design faults	For SW design faults see below, HW design faults are tolerated through hardware diversity.	For detection it is sufficient with two diverse replicas. For masking (no functional degradation) at least three diverse replicas are needed.
Software faults	Fault avoidance, SW diversity, backup system (different control laws)	For diversity see above.
Hardware faults **Transient,** **Permanent** **Symmetric** **Asymmetric/Arbitrary (Byzantine, SOS)**	Hardware replication	Hardware replication (Two, three or more similar components running the same task with comparison of the results)
Common mode faults	Fault avoidance	CMA during design phase.
Timing faults	Fault avoidance	Watchdog timers
Data faults	Bit flips are tolerated through EDAC (Error detection auto correction), transient data fault is corrected using recovery, and permanent data faults are tolerated through data redundancy.	Assertion checks (Dynamically adjusted checks of signal values), Coding (Extra bits in a data word or message, used to disclose error), Data redundancy with comparison, Double execution (Run the same process twice and compare the results)

A reliable communication system is designed using several of the mechanisms described here. For hardware faults, the single fault hypothesis duplicates the network, design fault mitigation (for broadcast bus) adds a backup network (hardware and software diversity), coding (CRC)

is used for error detection of messages, bus guardians to prevent babbling idiot faults. If the nodes are FCUs (see below) SOS and Byzantines faults can not occur.

A fault containment unit (FCU) is created with the purpose of preventing faults to propagate to other parts of the system after occurring. The main goal is to minimize dependence between parts of a system so a faulty component cannot affect the operation in other FCUs. A system design using FCUs has the additional benefit that it simplifies the safety assessment work.

5 Protocols

Three time-triggered protocols are included here TTP/C is chosen since this is the most mature time-triggered protocol available today (2006) addressing fault tolerance and predictability. AFDX is a protocol that is in consideration for safety-critical applications and is chosen because it could, in some cases, be an alternative to TTP/C. FlexRay is included since it is probable to be the next big standard in the automotive industry and can also use both bus and star topology. MIL-STD-1553 is widely used in both military and civil aircraft i.e. in Gripen. This protocol is included for comparison reasons. If a digital bus architecture shall replace a traditional point-to-point architecture it must be proven to be as reliable as the former one.

The choice of protocol is, of course, an application dependant process. Protocols that communicate over a joint medium such as a bus have to coordinate the communication to avoid collisions. In a point-to-point system every communication link is dedicated to the communication between two nodes and collisions is not a problem. Medium access method is one of the most critical parts of a communication protocol. An access attempt can either be triggered by an event or scheduled in advance.
An Event-triggered protocol does not guarantee a deterministic behaviour since on high loads more collision will occur which will increase delays. It is possible to analyze a specific schedule to guarantee i.e. response times. A protocol that uses Time-triggered (TT) medium access provides deterministic behaviour since all nodes know which node that is next to transmit at all times. This is well suited for safety-critical tasks like control loops that periodically reads sensors and updates actuators because of its deterministic behaviour and periodic nature.

TDMA is the medium access method used by TTP/C, FlexRay and MIL-STD-1553. It assigns all nodes a time window to transmit according to a predefined scheme (MEDL in TTP/C). Within the specified time window a node is granted exclusive access to transmit on the bus. The communication schedule is cyclic and divided into TDMA rounds where a node is allowed to transmit once. A number of TDMA rounds form a cluster cycle that is repeated continuously. One trade off when using time-triggered communication is loss of efficiency when nodes do only send sporadic messages but are still assigned a time-window. The effect of this can be reduced by using different schemes in different modes of operation i.e. take-off, flight and landing in an aircraft. Some protocols, i.e. FlexRay, can combine static TT-frames with dynamic ET-frames for sporadic messages.
This chapter describes protocols that could be interesting in safety-critical applications in avionics. Sections 5.1-5-4 give a brief description of each protocol.

5.1 MIL-STD-1553

The aircraft internal time division command/response multiplex data bus is a military standard with the designation MIL-STD-1553b. This bus was published in 1978 and is one of first data communication buses to send digital data between parts of a system over a common set of wires (a bus). Some of the applications are in the F-16 and the AH-64A Apache helicopter. It is also used in satellites, space shuttles and the International Space Station.
MIL-STD-1553 defines a redundant serial communication bus that interconnects nodes on a network. The medium is twisted pair and the maximum length of the bus is not defined, it is however recommended to test an implementation before deploying it.

The medium access is TDMA and a Bus Controller (BC) controls access to the bus. The BC contains a clock and commands nodes when to communicate, which removes the requirement for a global clock. The BC is replicated, since a malfunctioning BC would mean that there could be no communication on the bus.

The communication is normally not configured for redundant communication on both buses but rather with the secondary bus on hot backup in case of failure such as babbling idiot failure on the primary bus. One of the drawbacks with MIL-STD-1553 is that it is limited to 1 Mbit/s. There are implementations that are extended to 10 Mbit/s or faster that require star or hub coupling [14]

5.2 AFDX

Avionics Full-Duplex Switched Ethernet (AFDX) is a trademark of Airbus and was developed for the A380 passenger plane. It is a standard that defines the electrical and protocol specifications for the exchange of data between avionic subsystems using IEEE 802.3 (100 Base-TX) for the communication architecture. AFDX has been derived from Ethernet adding deterministic timing and redundancy management to the widely used protocol.

The deterministic communication is achieved by virtual links (VLs) that specifies virtual connections between parts of the system though the shared 100Mbit/s physical link.
Due to queues at switches that might introduce jitter and message latency there is a requirement that the delay from sender to receiver must be less than $500\,\mu s$, which does not include jitter at switches and the receiver [14] . Messages are numbered by the sender and checked at the receiver to ensure that the order of packets is correct.

The protocol does not support bus topology since the physical layer requires switches. The routing information is contained in tables in the switches. A redundant set of communication links and switches is required and messages are sent redundantly on both channels. The message that arrives first is used. AFDX has a slightly different area of application then FlexRay and TTP/C, such as less critical applications with higher requirements on bandwidth, but is an upcoming interesting technology.

5.3 FlexRay

According to [14] the FlexRay protocol is specifically designed to address the needs of a dependable automotive network for applications like drive-by-wire, brake-by-wire, and power train control. It is designed to support communication over single or redundant communication channels. It includes synchronous frames and asynchronous communication frames in a single communication cycle. The synchronous communication frames are transmitted during the static segment of a communication cycle. All slots are the same length and are repeated in the same order every communication cycle. Each node is provided one or more slots whose position in the order is determined at design time. Every node interface is provided only with the information concerning its time to send messages in this segment and must count slots on each communication channel. After this segment, the dynamic segment begins with the time divided into minislots. At the beginning of each minislot there is the opportunity to send a message, if one is sent the minislot expands into the message frame. If a message is not sent the minislot elapses as a short idle period. Messages are arbitrated in this segment by sending the message with the lowest message ID. It is not required that messages are sent over both communication channels when a redundant channel exists.

The current version of FlexRay lacks some of the fault tolerating mechanism that TTP/C features. No membership services are provided to detect faulty nodes. There are no bus guardian specification currently published and no published fault hypothesis.
The FlexRay consortium, consisting of many major automotive companies, has indicated it has no current interest in any field of application other than the automotive industry. The hardware that has been developed is only available to the consortium members and cannot be purchased by non-members. The protocol and physical layer specification was not publicly available until recently.
However, FlexRay is an interesting protocol because it is considered to be the next big standard, as CAN is today, in the automotive industry. This means that a lot of benefits can be drawn from large volumes that contribute to large scale development and testing, availability of tools and test equipment to low cost.

5.4 TTP/C

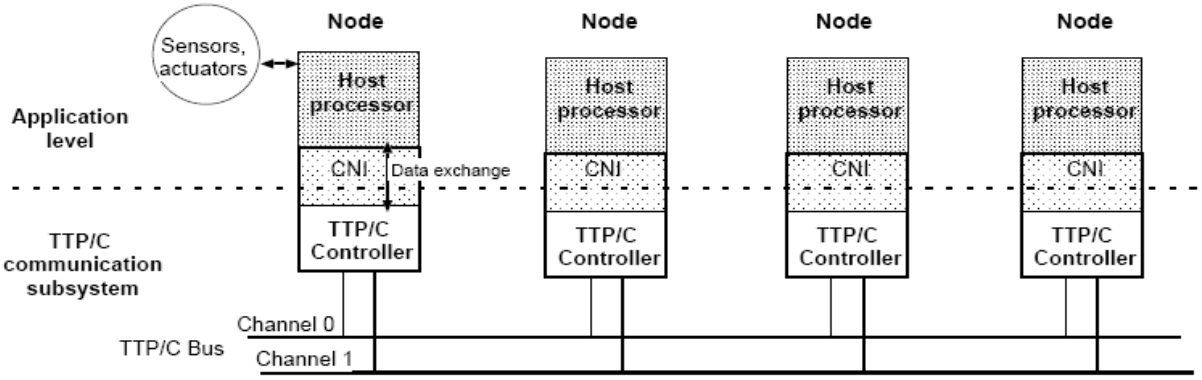


Figure 5.1: Cluster using TTP/C in a bus topology. Figure copied from [1]

The Time-Triggered Protocol (TTP) was developed by Hermann Kopetz and colleagues at the University of Vienna and is commercially developed by TTTech.
The protocol is a real-time communication protocol for the interconnection of nodes in distributed fault-tolerant real-time systems. TTP/C is designed to meet both the strong requirement of safety, availability and composability in the fields of aerospace and automotive electronics [1] , [7] . For a typical bus layout when using TTP/C, see Figure 5.1.
TTP/C can be implemented using a bus topology (see Figure 6.2), star topology (see Figure 6.3) or a combination of the two [1] .
The communication in TTP/C is redundant on two buses. Each communication interface has a bus guardian that prevents it from transmitting on the bus outside the predefined time-window. A faulty node that transmits on the bus outside of the defined time-window is called a babbling idiot. It has a fault tolerant time synchronization mechanism to establish a global time base and membership service to notify all nodes in the cluster of which nodes that are operational and if any failure has been detected.
The protocol is master less which allows communication to continue on the bus even if some nodes have failed.
TTP/C follows the single point of failure fault hypothesis meaning that it should tolerate a failure in any part of the communication system without degradation of performance. The term TTA (the Time-Triggered Architecture) is commonly used together with TTP/C. TTA is a concept where tasks are executed synchronously to the global time in TTP/C [3].

5.4.1 Membership

The membership service informs all nodes in a cluster about the operational state of each node with a latency of one TDMA round. A node is operational if the node has updated its life-sign in the membership vector within the last TDMA round. Requirements to update the membership are that controller is operating and is synchronized with the rest of the cluster.
If the node fails to update its life-sign it will be considered non-operational. It then stays synchronized to receive frames but not to send.

The membership vector is sent by each node in every slot and is check by all nodes to maintain an updated view of the cluster. The membership vector is also used for implicit acknowledgement. Implicit acknowledgement means that the receiver does not send an explicit acknowledgement that is has received data correctly. Instead the sender will check its own life-sign bit in the membership vector that the receiver transmits in the next transmission. If the sender is flagged as a correct node the transmission was correct and if not then the transmission failed. In this scenario the sender (A) might be the faulty node, but since nodes initially always consider them selves to be correct the sender will consider the receiver (B) to be faulty. However, if the next node to transmit (C) tells A that the B was correct the sender A sets it life-sign bit to zero until the problem is resolved and the sender can be reintegrated.

5.4.2 Communication Schedule

The communication pattern in TTP/C is described in the Message Descriptor List (MEDL) (see Figure 5.2). Each node has its own MEDL that contain detailed information about all communication between nodes in a cluster. The part that describes the cluster-wide communication is static and exactly the same in all nodes. Local node-information is however specific to each node.

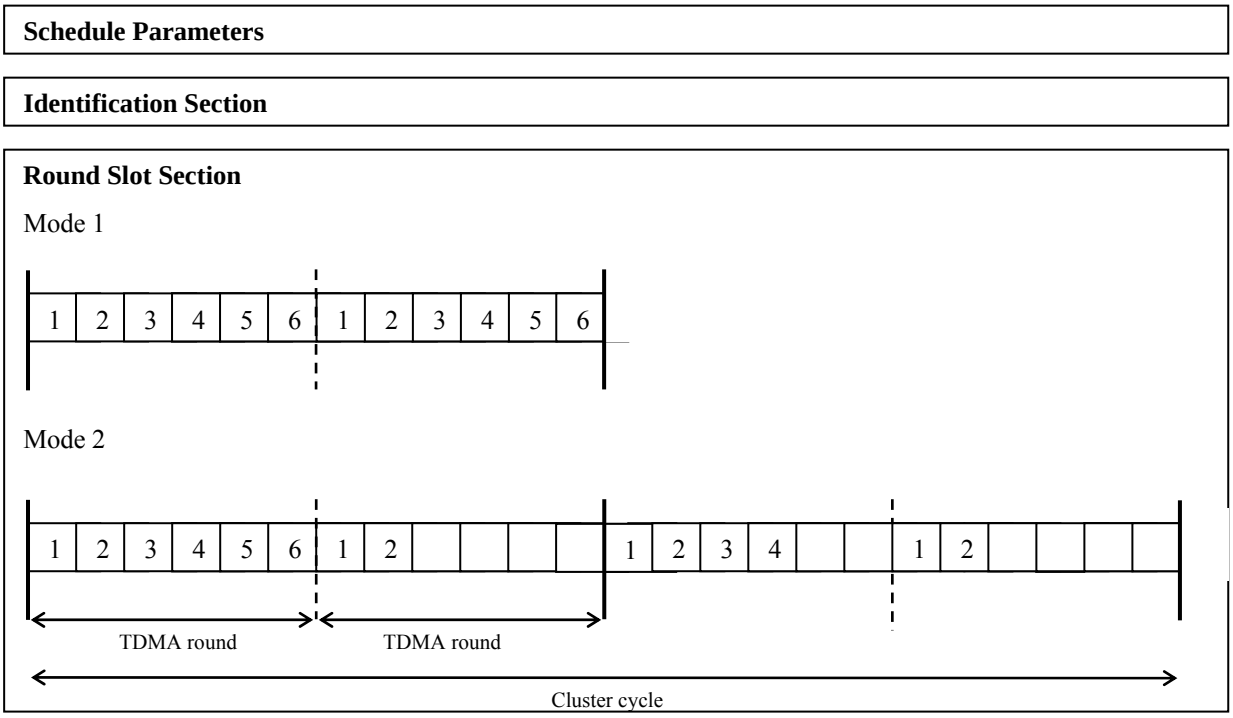


Figure 5.2: MEDL Layout Example.

A MEDL contains a few key elements that determine the communication on the bus (see Figure 5.2). Schedule parameters that describe the basic communication behaviour of the node and are necessary to start up or integrate a cluster.
Identification section contains data that is used when controlling that the MEDL is compatible with the cluster. Round slot section holds information about the rounds used in different operating modes. The example in Figure 5.2 is a visualisation of a MEDL containing two modes of operation; Mode 1 with two identical TDMA rounds and Mode 2 with four TDMA rounds that form a cluster cycle. In Mode 2 node 1 and 2 are assigned slots in each round, node 3 and 4 transmits every second round and 5 and 6 only once in every cluster cycle. This scheduling is suitable for control loops where some values and commands are needed more frequently than others. The empty slots are used to send I-frames with synchronization information etc.

5.4.3 Clock synchronization

One of the most important parts of time-triggered communication is the global time base. Since the whole concept relies upon that all nodes have synchronized clocks and thus knows that messages are transmitted at the correct instant according to schedule.
This is accomplished with a set of rules for synchronization and a clock synchronization algorithm. The TTP/C controller uses the Fault Tolerant Average (FTA) algorithm for clock synchronization.
The synchronization of clocks includes the following steps [1] :

1. Every clock reads the time values of a well-defined ensemble of clocks
2. Every clock calculates a correction term for its clock using the FTA algorithm
3. Every clock applies the correction term to its local clock to bring the clock into better agreement with the ensemble.

Reading the time values
The action time (AT) occurs in the start of every TDMA round and is used for exchanging clock information between nodes. The sender s starts transmission at time AT' which is at the start time plus a send delay that is long enough so no correctly synchronized node (receiver) receives the transmission before AT. AT' is called delayed action time of the sender.

$$t_{AT'_s} = t_{AT_s} + \Delta_{delay_s} \tag{1.1}$$

If the sender is not a master clock or the frame is corrupt, the transmission is disregarded.
The time difference between the expected arrival time from the MEDL and the actual arrival time is measured locally in microticks at the receiver.
The MEDL also contains a correction term $\Delta_{corr_{s,r}}$ that holds the number of microticks that a transmission takes between sender and receiver. This is added to the expected time of arrival. The clock values received are stored in a four-value push down stack. New values replace old values that get pushed down off the stack upon a correct arrival of a new clock value.

Calculating the correction term
The slot in which the correction term is calculated is marked in the MEDL with the ClkSyn flag. This ensures that all nodes synchronize their clocks at the same time. If the sender of this slot is a master clock node the clock value is stored on the stack before the internal correction term is calculated since master clock nodes have more accurate clock with less probability for drift. The largest and smallest value on the stack is discarded and the average is calculated

from the two remaining values. This forms the clock state correction term. If the stack does not contain four values the average is taken from the available values.
It is also possible to make use of an external clock correction term from e.g. a GPS. This term is calculated at the host of the node that is connected to the GPS and sent to all the other nodes as normal data. It is extracted and added as external rate correction field in the CNI of the controller.
If the absolute value of the external clock correction term or the absolute value of the total correction term is larger than $\Pi / 2$ (where Π is the precision and has to be smaller the one macrotick) the node raises a synchronization error and freezes.

Correcting the local clock
A defined amount of microticks is corrected after ftMTs (free-running macroticks count) macroticks. After the correction term is exhausted, the clock runs free until the next synchronization instant marked in the MEDL.

This clock synchronization procedure is only for one channel. For two channels both channels need to be done differently because of different propagation delays.

6 Topologies

There are two types of data communication between processors and between processors and peripherals: channels and networks. A channel provides a direct or switched point-to-point connection between communicating devices. Channels are usually hardware intensive and provide high bandwidth with low overhead. A network is a collection of processors and peripherals that interact with each other using a protocol. Networks require more software to handle the communication but can be used to a larger variety of tasks. Each communication technique has its advantages and disadvantages and within aircraft different topologies, protocols and media are used.

The most common topologies are point-to-point, bus, star, ring and combinations of these. In avionics point-to-point topologies is dominant because of superior reliability. Over the last decades the focus on environmental issues of the aviation industry has grown. Due to this weight reduction is a big issue today. Large savings can be made if a sparse (partly connected) mesh (see section 6.1) or bus can be used instead of a fully connected mesh topology.

6.1 Point-to-point interconnect

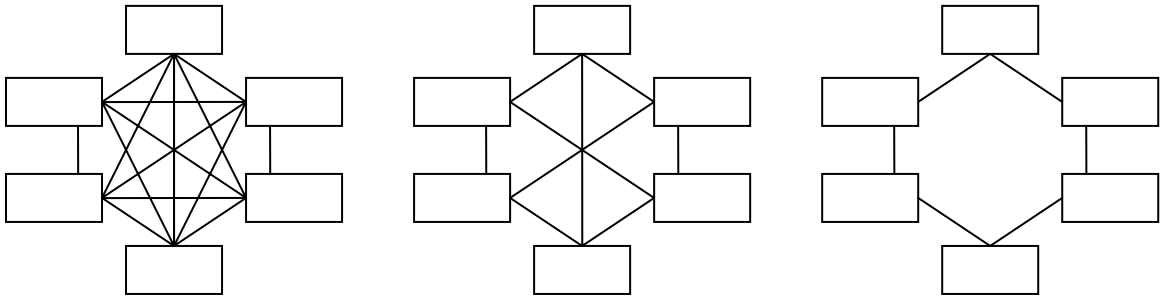


Figure 6.1: Point-to-point topologies

Point-to-point interconnects can be configured in many different ways. A fully connected point-to-point communication system (fully connected mesh) has a topology where all nodes have point-to-point interconnections between them selves and all the other nodes in the network. The communication capability of this topology is very high because messages can be transmitted in parallel on all communication paths. It is a very robust topology that has extensive redundancy to the cost of being very hardware-intensive. The complexity increases fast with the number of nodes and a reconfiguration is difficult to achieve. To reduce weight and complexity to the cost of reliability it is possible to use a mesh that is similar but not fully connected (partly connected mesh).

A ring topology is point-to-point connection where every node has two connections to other nodes so that all nodes form a ring. Messages are sent in one direction and repeated by all nodes until it reaches the sender again. In order to tolerate a single fault on the ring it is possible to exchange communication direction. A system with redundant rings and redirectional transmission is very robust.

A big advantage with a point-to-point topology is that it creates physical separation between communication channels. Separation between the channels creates natural fault containment zones.
A star interconnect is also a kind of point-to-point topology that will be further discusses in section 6.3.

6.2 Broadcast bus interconnect

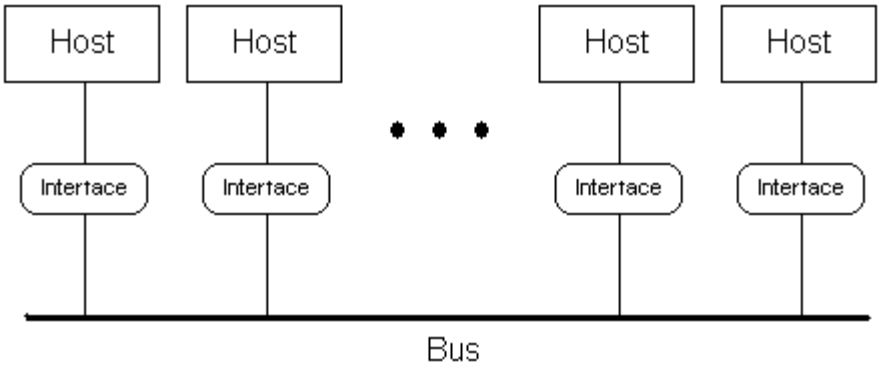


Figure 6.2: Bus Interconnect. Figure copied from [5]

The broadcast bus topology consists of a medium (which in principle can be an electric wire, optical fibre or radio link) to which all nodes are connected. Buses always transmit broadcast messages to all members on the bus. Because of this only one member can send at a time despite the fact that there usually exist several paths within the bus. Since all nodes will see the signal at, virtually, the same time on the bus timing analysis becomes easy compared to making an analysis of a complete point-to-point communication system.

The bus topology is flexible and composable and can easily be reconfigured without any major hardware redesign. Buses generally utilize less wiring and interfaces than i.e. point-to-point and star topologies. Since all nodes are connected to the bus there is a single point of failure issue which must be handled. Fault containment need to be created using mechanisms such as membership to prevent that faulty members disrupt the communications between other nodes on the bus.

The protocols supporting bus topologies in safety-critical communication systems available today are limited in bandwidth compared to point-to-point topologies. This comes from the fact that on a bus all nodes share the bandwidth while every point-to-point is dedicated to the communication between two nodes. A bus topology is considered very reliable since it is a passive component with few fault modes. Following are single points of failure in a bus topology:

- *Short-circuit:* At a short circuit between the wires in a twisted pair connection the whole bus fails to transfer information. This is a rarely occurring event with very low probability and the only way to be able to handle this fault is to use static replication if the form of dual or more redundant bus. A more frequent fault mode is short circuit to ground in the node bus interface or in the connector. It is usually possible to still maintain communication with a lower capacity in this fault mode if the system is designed for “graceful degradation”. The influence of short circuit fault modes can be reduce by using galvanic isolation at the stub of each node on the bus.

- *Circuit cut-off:* A cut off of wire is a low probability fault, which will divide the system in two parts that might or might not be recoverable. Redundancy in communication buses located separately handles this fault mode. Circuit cut off in a connector is the most common connector fault mode and it will only affect the node but not the bus function.

- *SOS-faults:* When a node sends messages on the limits or slightly out-of-specification of the assigned time window such that some of the receiving nodes will receive the messages correctly and some will not. In this case the nodes membership opinion will not be consistent if not an atomic broadcast mechanism is provided.

- *Babbling idiot:* This is when a faulty node sends outside its pre-assigned time-window which destroys the communication that was scheduled at the time on the bus. Only one message must be sent at a time and depending on the chosen protocol different mechanisms prevent nodes from sending at the wrong time. All nodes in a safety-critical bus communication system therefore must be designed with a fail-silent behaviour in the time domain or at least it must not violate the protocol rules. Bus guardians are one way of preventing babbling idiot faults.

6.3 Star interconnect

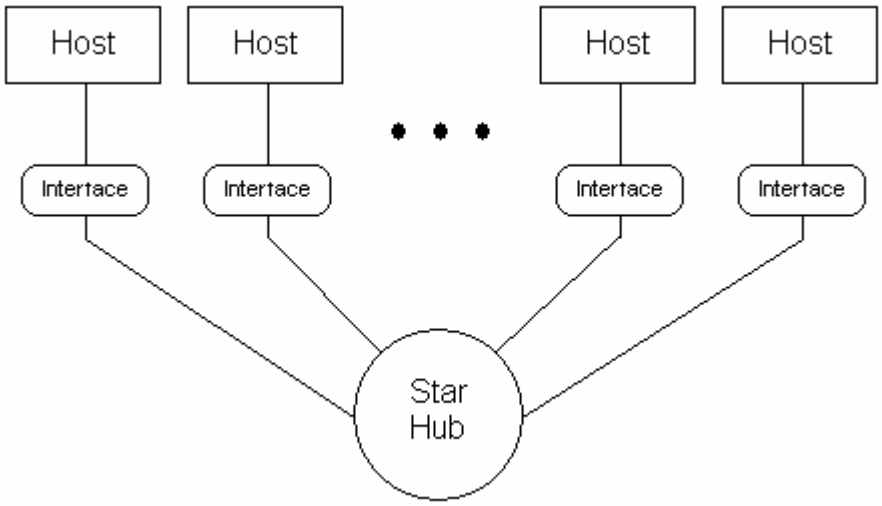


Figure 6.3: Star interconnect. Figure copied from [5]

A star-coupled system relies on one central node, a star or a hub, that is connected to all other nodes. The star is the obvious master of the communication since there are no direct connections between the other nodes. The central position puts the star in charge of all communication and faulty nodes can be excluded since it controls the access to the communication medium. In a star topology it is natural to implement the guardians in the star coupler since this makes them physically independent from the node.

However, if the star breaks down or becomes faulty the whole network will experience loss of communication. The probability of that this fault mode can cause total communication black out can be significantly reduced by using redundant stars.
Star topology is a hardware intensive topology that requires more wiring than most other topologies; an exception is the fully connected mesh. Additionally the star coupler is much more complex and has larger failure rate than a passive component like a cable in a mesh or

bus interconnect. Delays are increased when using a star coupler since the routing at the switch is not instant.

Fault tolerance mechanisms placed in the star (filters, pulse shaping, etc) can handle asymmetric faults as Byzantine, SOS. However, the star is an active and rather complex component which might have many fault modes. As a worst case the star coupler becomes Byzantine; it will tell different things to nodes in the system. That is a very hard fault mode to detect and resolve.
The star topology is inflexible to hardware reconfigurations since the star couplers need to be reconfigured. One advantage a star topology has over a bus is that the physical layers that are available for star topologies today supports higher bandwidth. TTP/C using 100Base-TX has the maximum bandwidth of 25Mbit/s and AFDX 100Mbit/s.

7 Architecture comparison

This chapter contains a conceptual study of communication architectures where the protocols investigated in this thesis are theoretically evaluated for use as a primary communication system in a Flight Control System (FCS).
The flight control system that has been used as a template for this study is a fictitious flight control system based on JAS 39 Gripen. To avoid confidentiality issues the details of this flight control system has been entirely collected from [15] .

Important issues which are compared are:
- The amount of hardware required by each architecture.
- The reliability presented as the probability of system failures due to loss of communication network.
- Maturity, specific features and limitations.

Note that power supplies and extra connections for i.e. emergency shutdown are not included in this study.
Section 7.1 describes the flight control system that has been used as a template for this chapter. Sections 7.2- 7.4 describe the communication architectures and discuss strengths and weaknesses. Section 7.5 contains reliability calculations and failure rate estimates of the architectures. Section 7.6 contain estimated data specific to the architectures and a comparison of features of the protocols included.

7.1 Analogue point to point

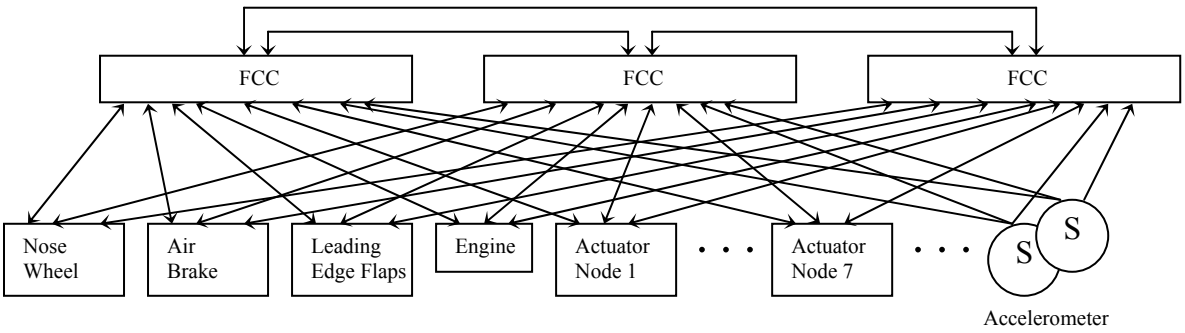


Figure 7.1: Distributed flight control architecture with analogue point to point connections

The FCS in JAS 39 Gripen has one system core (flight control computer) with triple redundant channels. In order to simplify the "porting" and make it more understandable to the reader the FCC is described as one function distributed over three FCC-nodes (see Figure 7.1). The FCCs use the sensors(s) data as input to the control laws to calculate control commands that are sent to the actuators. Data are compared between the FCCs and the control commands go trough a voter at the actuator in order to prevent a faulty control command to reach the actuator.

The flight control system contains a total of 20 nodes that are connected to the FCC nodes using point-to-point connections. The FCCs are connected to each other to make data exchange possible e.g. for state comparison, data exchange etc.This is a traditional point-to-point flight control system used here as benchmark.

7.2 Bus interconnect using TTP/C

The architecture using TTP/C consist of two redundant communication buses that transmit all messages on four channels (two buses with two redundant channels each, see Figure 7.2). The choice to use fours parallel channels is based on that one controller has two channels and in case of a permanent failure of the controller the node loose the two channels connected to that controller.
Redundant transmission of messages on different channels introduce fault tolerance since every message is transmitted on four channels in parallel.

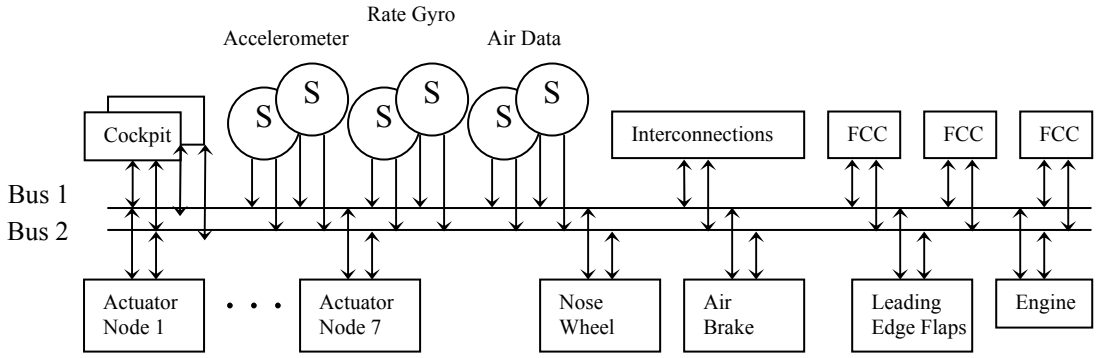


Figure 7.2: Distributed flight control architecture with dual redundant broadcast bus connections using TTP/C

TTP/C is specifically designed for safety-critical hard realtime systems and has several features to provide dependable communication built in. These mechanism are crucial in a safe-critical communication system because of the fault modes that are introduced when using a bus described in 6.2.

The protocol is not specified for a certain physical layer which makes it flexible. It has been tested with buslengths up to 100 m using using MFM/Manchester coding over a RS485 interface which would make implementation feasible even in a large aircraft.
The maximum bandwith using RS485 with MFM/Manchester coding is 5 Mbit/s. MII 100Base-TX ,that provides bitrates up to 25 Mbit/s, can also be used if changing the archtecture to use star couplers which uses a topology similar to the one in section 7.3.

In order to protect the nodes from a lightning effects, short circuit and other electrical impulses that might be distributed by the bus galvanic isolation is needed at each node. This will reduce the transmission rate on the network to somewhere in the range 1-2Mbit/s. As can be seen in the cluster example in APPENDIX C; 2Mbit/s or lower is sufficient to schedule a system of this magnitude on a TTP/C bus.

A small CPU is required at each node to handle the FT-COM layer that is either included in TTP-OS or implemented by the user at the host CPU. It is possible to implement the TTP controller in an FPGA or ASIC in order to reduce the amount of hardware at critical parts of a system.

The conclusion of this architecture is that is would be implementable in a flight control system. Issues that need to be investigated closely include finding a physical layer that can provide atleast twice the bandwidth that is needed by the FCS using galvanic isolation at each

node. Enivironmental testing of the controller and the host CPU also need to be performed to ensure that the severe enviroment by an sensor / actuator can be handled.
It is highly probable that for a flight control system using this technique to pass certification it would have to have a backup system that has been fully certified for operating without support from the main FCS.

7.3 Star interconnect using AFDX

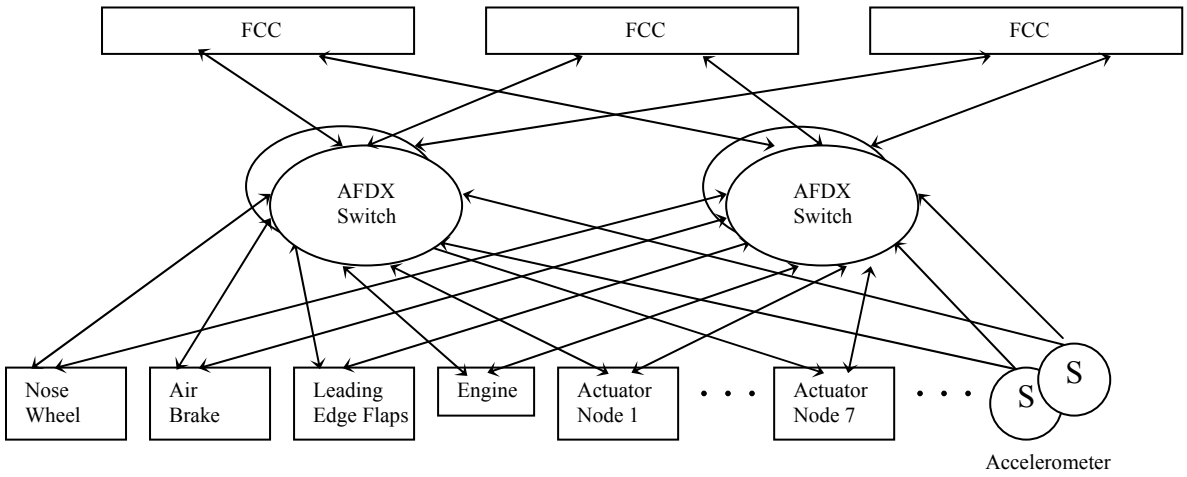


Figure 7.3: Distributed flight control architecture with dual redundant star couplers using AFDX

The architecture using AFDX uses ethernet as physical layer and consequently need four AFDX-switches The placement of these switches in the body of the aircraft will affect the amount of cables that is needed. Since the switches are complex hardware and the most critical part of the system they will have to be placed inside the aircraft-body.

Each node in the system (including the FCCs) has four redundant links; one to each of the redundant switches.

AFDX requires one switch for each channel which makes it a hardware intensive and complex compared to a bus or point-to-point architecture. The introduction of additional complex hardware increases the failure rate significantly which can be seen in the realiability calculations in 7.5.
According to the ethernet specification a shielded link can be up to 100 m which would be sufficient for implementation even in a large civil aircraft.

My conclsion of using AFDX in a flight control system are that there are two major disadvantages compared to the two previous architectures that make that archtecture unsuited for that application. First, the protocol require a powerful CPU in each node to process the packing and unpacking of frames of the datapackets from the 100Mbit/s datastream. Putting such a computer by every primary control surface is not feasible. Secondly, the delays introduced at the switches combined with the significatly higher latency jitter bound 500µs are not acceptable in a communcation system that transport highly critical data used for flight control.

7.4 Bus interconnect using FlexRay

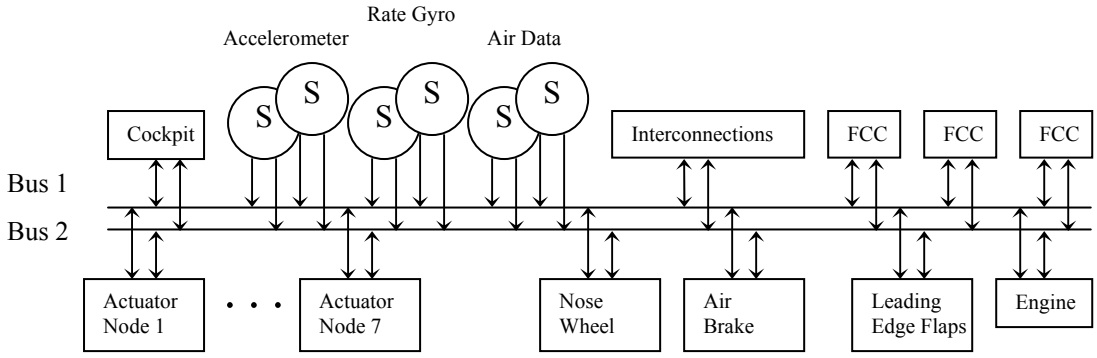


Figure 7.4: Distributed flight control architecture with dual redundant broadcast bus connections using FlexRay

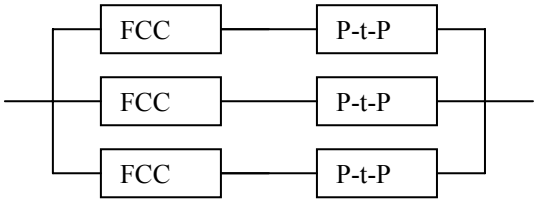
An architecture using FlexRay as primary communication system has many similarities to the TTP/C architecture in 7.2. However, as previously mentioned FlexRay lacks fault tolerance such as bus guardians and membership service that are needed to handle fault modes such as babbling idiot faults. Exclusion of fault tolerant mechanisms such as membership in the protocol layer means that such mechanisms must be implemented at application level. A result of this is that FlexRay would need a more powerful CPU at each node than a protocol with these services built in at protocol level.
The physical layer of FlexRay only allows bus lengths up to 24 m which is a major disadvantage considering that a civil aircraft usually have a wing span larger than 50m.
This is clearly a protocol that is designed for the automotive industry. However, it is not impossible that in a future a FlexRay that has a broader market targeted will be published.
The physical layer of FlexRay is specified for up to 10 Mbit/s and allows event-triggered communication to be fitted in the time-triggered schedule.

The conclusion is that FlexRay is still an immature technology that is only available to members of the FlexRay consortium. As long as the maximum bus length is 24 meters an implementation in a FCS would not be feasible.

7.5 Reliability calculations

These reliability calculations include the communication channels only. The triple redundant FCCs are included in the pictures for understanding the layout of the different communication architectures. Failure rates used for the calculation come from [20].

Point-to-Point



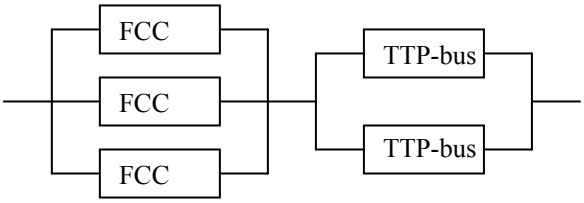
$p_1 = e^{-\lambda_1} \qquad p_2 = e^{-\lambda_2}$

$P = (1-(1-p_1)^3)(1-(1-p_2)^3)$

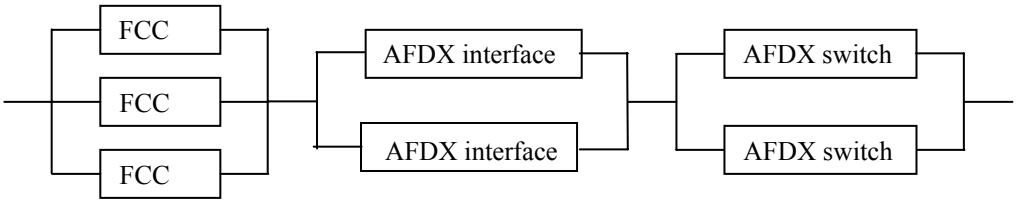
p_1 is reliability of a single FCC

p_2 is reliability of a Point-to-Point subsystem
P is reliability for the complete system
$P_{Systemfailure} = 1 - P_{PtP} = 29 \cdot 10^{-12}$

TTP/C



$p_1 = e^{-\lambda_1}$ $p_2 = e^{-\lambda_2}$
$P = (1-(1-p_1)^3)(2p_2 - {p_2}^2)$
p_1 is reliability of a single FCC
p_2 is reliability of a TTP-bus
P is reliability for the complete system
$P_{Systemfailure} = 1 - P_{TTP/C} = 76 \cdot 10^{-12}$



$p_1 = e^{-\lambda_1}$ $p_2 = e^{-\lambda_2}$
$P_{AFDX} = (1-(1-p_1)^3)(2p_2 - {p_2}^2)(2p_3 - {p_3}^2)$
p_1 is reliability of a single FCC
p_2 is reliability of a AFDX interface
p_3 is reliability of a AFDX switch
P is reliability for the complete system

$P_{Systemfailure} = 1 - P_{AFDX} = 10 \cdot 10^{-9}$

7.6 Comparison

Table 7.1: Architecture summary

Communication Architecture	TTP/C (bus)	TTP/C (star)	Point to Point	AFDX	FlexRay (bus)
Number network interfaces	92	4*23*2 = 184	120 + 6 = 126	4*23*2 = 184	92
Cable length (m)	4*50 = 200[1]	23*4*7 = 644[2]	23*3*7 = 483	23*4*7 = 644	4*50 = 200
Star coupler / Switch	None	4	None	4	None
Bandwidth	5 Mbit/s using RS485	25 Mbit/s using 100Base-TX	Depending on physical layer	100Mbit/s using 100Base-TX	10 Mbit/s using twisted pair
P(System failure)	$76 \cdot 10^{-12}$	NA	$29 \cdot 10^{-12}$	$10 \cdot 10^{-9}$	NA

Table 7.2: Protocol comparison

	TTP/C	FlexRay	AFDX	MIL-1553 (point-to-point)
General				
Previously used in avionics	Yes, Airbus A380	No	Yes, Airbus A380	Yes, Jas 39 Gripen among many other
Backup system required for certification	Yes	Yes	Yes	No
Global time base	Yes	Yes	No	No
Availability	Available as COTS HW. Tests equipment available.	Available as COTS HW for consortium only. Tests equipment available.	Available as COTS HW. Tests equipment available.	Available as COTS HW. Tests equipment available.
Performance				
Bandwidth	Bus 5Mbit/s using RS485 (MFM), Star 25Mbit/s using MII 100Base-TX	10Mbit/s using twisted pair	100Mbit/s using Ethernet	1Mbit/s using MIL-1553
Latency jitter	Programmable 1-10 µs	Programmable 1-6 µs	<= 500 µs	<12 µs
Duplex	Half	Half	Half	Full

[1] A bus length of 50 meters is used.
[2] An average link length from star/hub coupler of 7 meters is used.

Physical Layer				
Medium access method	TDMA	TDMA	CSMA/CD	Event-triggered
Topology	Bus / Star	Bus / Star	Star	Point-to-point
Physical layer independent	Yes	No, has a twisted pair PL in specification	No, Traffic control using VL maintained by end system	No
Cable length	Up to 100m using bus. Depending on number of nodes.	24m bus specified	100m cable length (100m from node to star)	Not specified > 100 m
Fault tolerance				
Inherent Redundancy	Dual-redundant bus	Dual-redundant bus	Dual-redundant bus	Dual-redundant links and switches
Redundancy management	Yes, Node and task replication in both HW and SW	No, must be implemented at application level	No, must be implemented at application level	No, must be implemented at application level
Membership service	Yes	No	No	No
Fault containment	Yes, in hardware, membership, message status, dual redundant bus	No, must be implemented at application level	Yes, if fault is limited to one of the redundant switch networks	Yes, if the secondary bus can be used to remove or tolerate the fault

8 Experimental setup

The TTTech equipment used in the lab includes four nodes. Below is the work, problems experienced described when trying to setup a four node cluster. The intention was to setup a cluster where we could measure/test

- Delays
- Start-up time
- Resynchronization
- Fault tolerance mechanisms (replication, reliable message passing, etc)
- Physical layer issues

8.1 TTTech TTP-IP module

The hardware platform consists of four TTP-IP modules from TTTech mounted on two Tews TCP212 CPCI carrier cards. The TTP-IP modules are equipped with a Freescale MPC555 PowerPC® and the TTP controller AS8202NF. For more information about the TTP-IP module see [2]

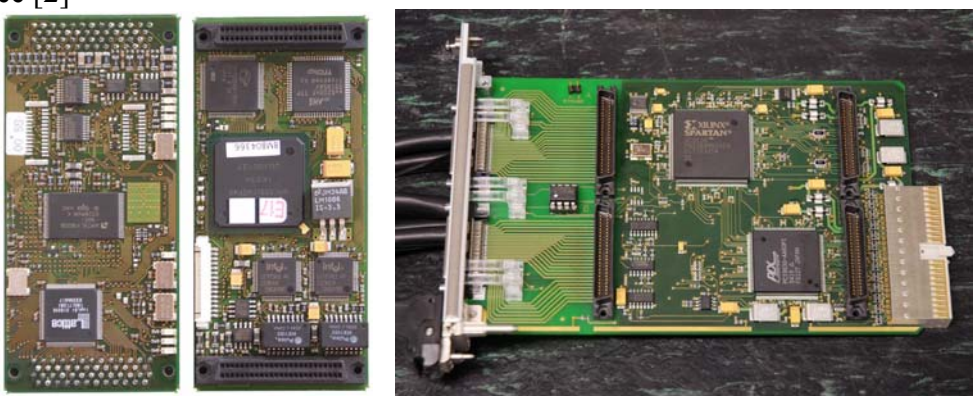


Figure 8.1: TTP-IP module (left), CPCI IP-carrier (right)

The start-up phase, involving acquiring cables and connectors in order to power and connect the TTP-IP modules, was very time consuming. For more details on hardware problems (see Section 8.4).

8.2 Hardware setup

The hardware setup that has been used to the tests carried out is schematically described in Figure 8.2.

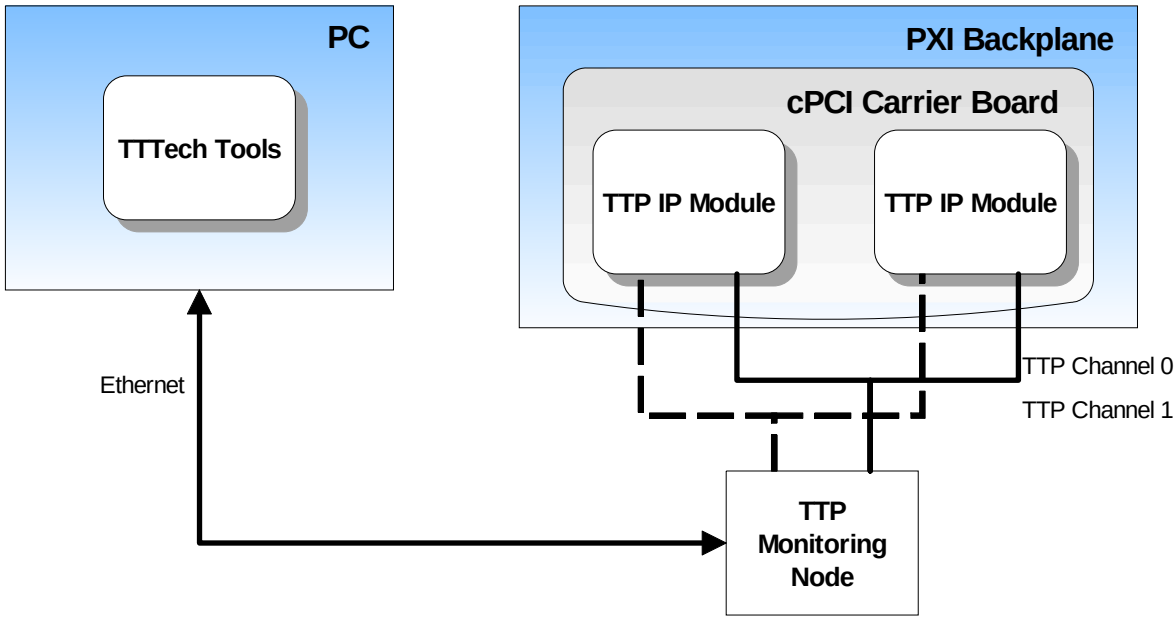


Figure 8.2: Schematic lab setup using two TTP-IP modules and a monitor node

Due to one broken CPCI IP-carrier board (see Section 8.4) the connection in Figure 8.2 was the only possible. A VME IP-carrier card was ordered to be able to run a four node cluster but the delivery time was outside the time line of the thesis work. The actual lab connection is shown in Figure 8.3.

All tests described here use RS485 as physical layer and 2Mbit/s bandwidth. In order to have a properly working bus it is necessary to have termination resistors at the end of the bus, either in the last node on the bus or at the end of the bus past the last stub.
The TTP-IP modules do, by default, have RS485 termination resistors mounted on them. In the connection below there is no need to change the configuration since the TTP Monitoring node is not terminated by default.

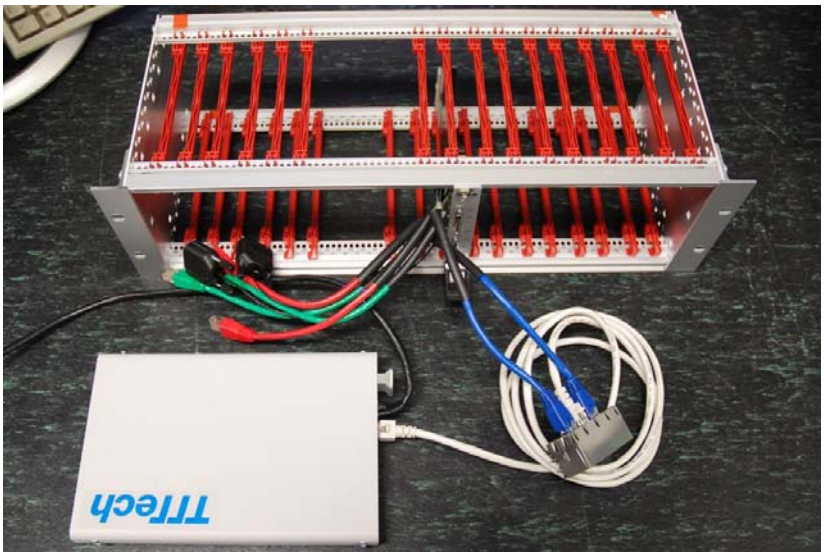


Figure 8.3: Laboratory setup using two TTP-IP modules and a monitor node

The connectors used to connect the IP-modules to the bus are standard RJ45 connectors where pin 3-6 is used. For simplicity both channels of the TTP/C bus are connected to the bus using one cable. The bus here is a RJ45 connection block that has all pin 3 connected, all pin 4 connected etc.

8.3 Cluster design – TTP-Tools

The design of a TTP/C cluster can be done by using the TTP-Tools tool-chain from TTTech. An overview of the development process is given in Figure 8.4.
The design at cluster level is done in TTP-Plan. Objects like hosts, subsystems, messages and properties like TDMA round length are defined here. When the cluster design is done a complete communication schedule is generated by TTP-Plan.
The schedule, objects and other settings are written to a cluster database that is used by subsequent tools in the tool-chain. TTP-Build is used to configure every single node. Tasks are defined and connected to the messages from TTP-Plan. TTP-build creates the individual MEDL that is loaded into every node. TTP-Load is used to upload the compiled application and the MEDL to the host. TTP-View is used to monitor the traffic on the network using a Monitor node from TTTech.

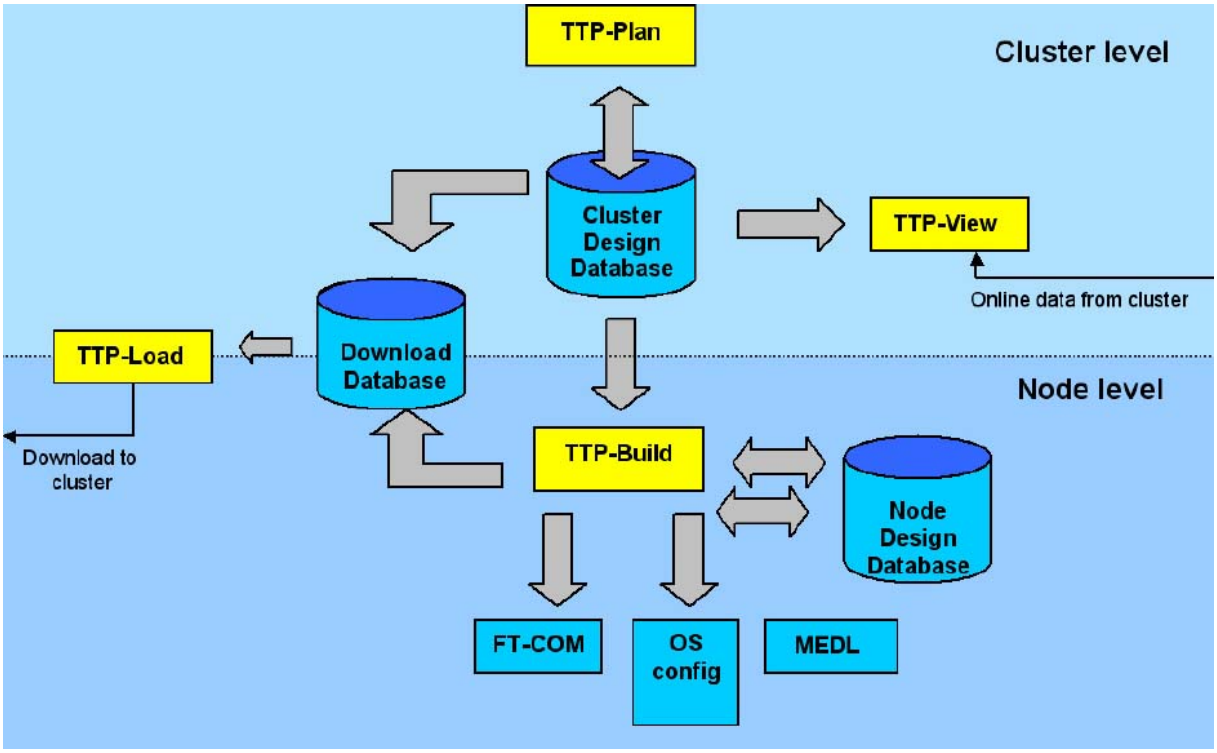


Figure 8.4: The TTP software development tool-chain. Figure copied from [17]

The design of the communication system discussed in 7.2 is described in APPENDIX C. TTP-Matlink is a prototyping tool that is included in the TTP-Tools suite. It is a Matlab Simulink toolbox that uses Real-time workshop and Real-time workshop embedded coder to generate C-code for each node. TTP-Matlink is very useful for rapid prototyping when experimenting with different configurations since design and implementation of a TTP/C communication system is quite time consuming.
I have designed several clusters using this tool chain, including one cluster illustrating the FCS using TTP/C (see Figure 7.2). The results and work can be found in APPENDIX C.

8.4 Hardware problems

When starting up with the hardware purchased from TTTech (described in 8.1) the following problems was encountered:
Initial powering error of carrier boards due to invalid backplane. Since the only available CPCI-backplane was invalid a power adapter was constructed. The adapter have the format of a CPCI backplane connector but only 20 power supply pins and 13 ground pins was connected since the I/O to the backplane was not needed in the experiments and tests that was planned (see Figure 8.5).

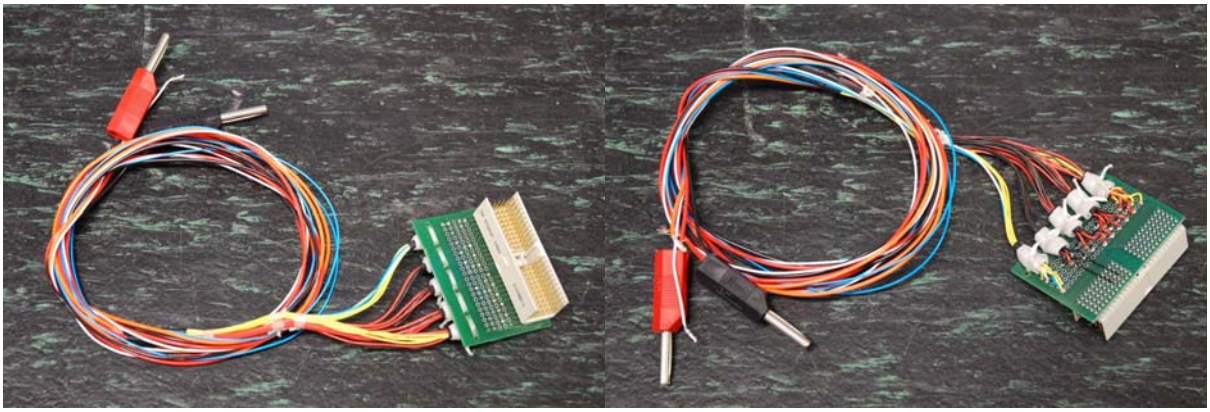


Figure 8.5: Power adapter for CPCI IP-carrier card

Unfortunately the Compact PCI standard contains some pull-down reset signals that have to be "1" except for when a reset is requested from the backplane. This fact caused the MPC555, that works as host CPU on the TTP-IP modules (mounted on the Compact PCI carrier), to always be in reset when using the power adapters. An attempt to simulate that the reset was pulled up was made by putting 3.3V on it permanently but for some reason this did not work.
However, after some weeks a backplane that could be borrowed was found in Saab's calibration lab. The use of this backplane solved the problem from attempt one.

The connectors on the Tews Compact PCI carrier boards is of the type Champ 50 which is a type of high density shielded connector similar to the a high density SCSI connector. Some search for this connector resulted in that it had to be ordered from the US and the delivery time would be at least 8 weeks which was too much time. The purpose of these connectors was to connect the TTP/C bus.
The problem was solved by soldering wires directly from the Compact PCI board (see Figure 8.6).

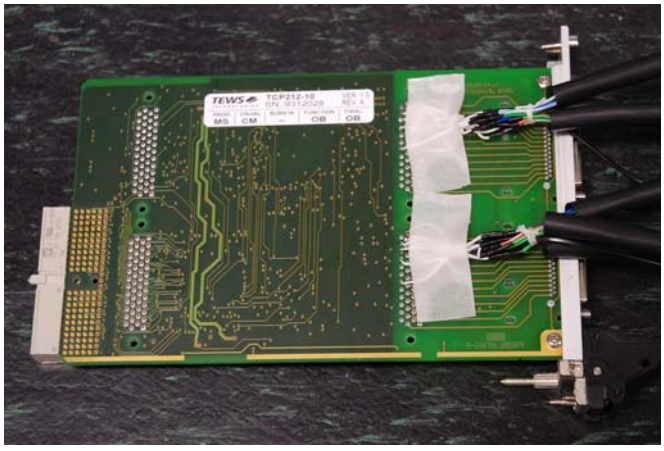


Figure 8.6: Modified CPCI IP-carrier

One of the Compact PCI carrier boards was found to be broken when testing the wires soldered on to it. It is probable that the invalid backplane was the cause of this since the error in the backplane initially used put 12V on a 3.3V pin.

The fact that one of the carrier boards was broken was a major draw back in the testing since this meant that only two TTP-IP modules could be used at the same time.

9 Tests performed

This chapter describes the performed tests and results of these. For more information about the test cases see APPENDIX B.

Test case 1 - Basic functionality test - start-up: Pass

TTP/C communication has been established using the test setup described in 8.2. The cluster used is configured to include four nodes. The TDMA round is 1200 µs and consist of four slots (see Figure 9.1). The operation of the cluster was inspected using TTP-View; the tool to monitor TTP/C networks from TTTech.

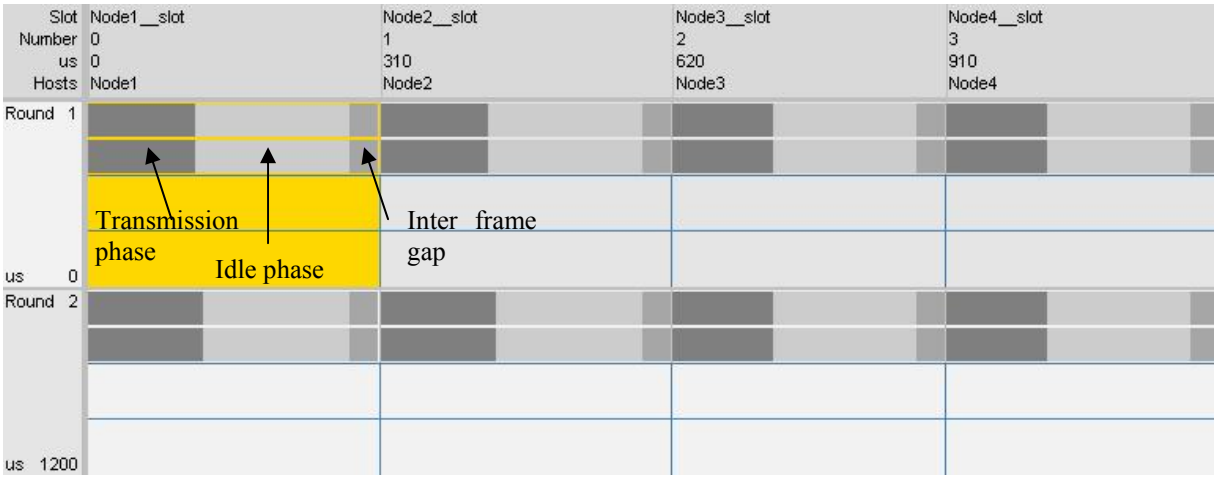


Figure 9.1: Schedule of the test application used in TC1 and TC2

Since TTP/C is master less nodes a node will synchronize and establish communication if any other participant on the network is alive. The signals on the bus can be seen in Figure 9.2 where it is visible the node number 1 and 2 in the cluster is present and transmits on the bus, while nodes 3 and 4 are absent these slots are empty.

Test case 2 - Basic functionality test – timing: Pass

The cluster described above is also used in this test.

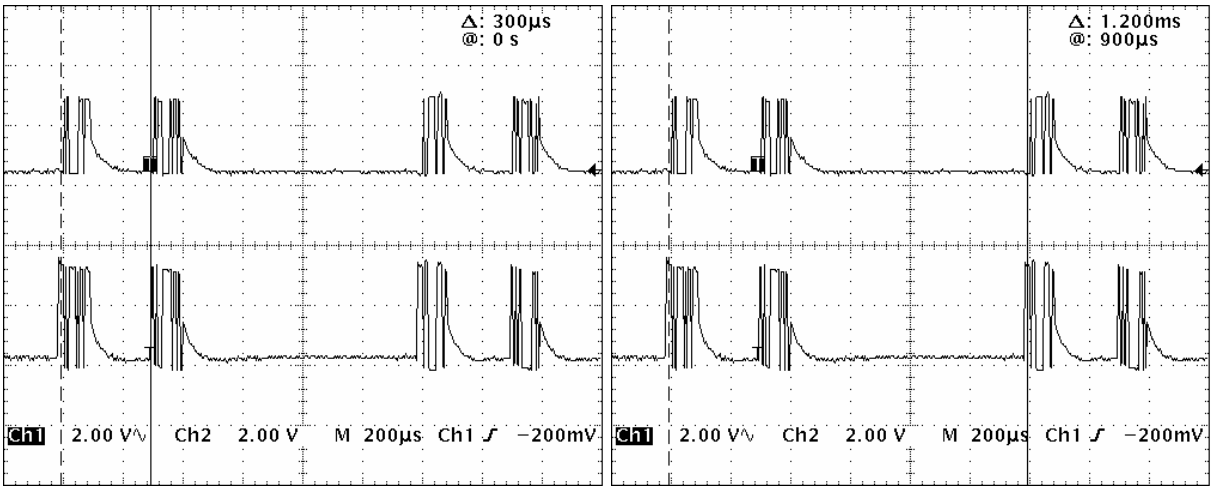


Figure 9.2: Timing measurements on TTP/C-bus

This test illustrates the physical signalling on the bus and shows conformance to the generated schedule. By visual inspection and comparison of Figure 9.1 and 9.2 it is illustrated that schedule and real signalling correspond to each other.

Test case 3 - Fault injection: Did not finish
To be able to complete this test either a degradation of the controller such as the one performed in [18] or fault injection methods would be necessary.

Test case 4 - Reintegration: Did not finish
Reintegration would require at least three “normal” (not including the passive monitor node) nodes on the bus and since only two nodes have been operational during testing (discussed in Section 8.4) these tests could not be carried out.

Test case 5 - Shared responsibility between nodes (replication): Pass
This test case is partially shown by the cluster in test case 1 but with node 1 and 3 operational. The configuration of the cluster is that node 1 and 2 runs a subsystem with a counter that sends messages with the value to node 3 and 4. Since the subsystem runs on both nodes 1 and 2 the TTP/C built in redundancy handling will coordinate the operation if either node 1 or 2 goes offline or is faulty. The test shows that this works since counter values are received at node 3 even though node 2 is offline. It remains to test the functionality when all four nodes are connected to the bus.

Test case 6 - Time measurement of start up, reintegration and restart: Did not finish
This test case did not finish. The start up time of the cluster was not measure because it is dependent of how many nodes that is operational. The start-up time of two nodes is not considered interesting.
Reintegration and restart would require at least three “normal” nodes on the bus and since only two nodes have been operational during testing (discussed in Section 8.4) these tests could not be carried out.

10 Result and Analysis

Are time-triggered broadcast buses (TTP/C) suitable for a flight control system?

Certification:
TTTech that develops TTP/C commercially has put a lot of effort in making TTP/C certifiable. Since the protocol is used in the new Airbus A380 it will, eventually, be certified when the final certification of the aircraft is completed.
The AS8202NF controller has been designed to meet the criticality level A of RTCA/DO-254. The TTP driver, TTP-OS, TTP-Verify (tool to verify communication schedules) and the TTP loading library has been designed to meet level A of RTCA/DO-178B.
This proves that certification of TTP/C itself will most likely not be a problem if it would be used in a safety-critical system in avionics. However, when using a new technology in the avionics industry a backup system is needed for certification.

Functional requirements:
Bandwidth: Yes. The 5 Mbit/s that is supported by RS485 using Manchester/MFM coding is sufficient for a flight control system (2 Mbit/s has been found sufficient with proper scheduling and communication running on 80-100 Hz). Bandwidth using transformer coupling is yet to be investigated.
Latency Jitter: Yes. (Is configurable in TTP/C 1-10 us)
Maintenance: Yes. Nodes can easily be added but the communication schedule needs to be rebuilt if empty slots have not been included for future additions in previous design. TTP/C even supports hot-swap: a node could be replaced without disrupting the other communication.
Number of nodes: Yes. TTP/C supports up to 64 nodes in one cluster which is considered to be more than enough for a flight control system.

Environmental requirements:
Temperature: Not shown. The TTP chip AS8202NF is specified for -40°C to 125°C. For electrical components that are to be placed in hazardous areas like the wings an operating range of -60°C to 70°C. Considering that when powered components will generate heat it might very well show that the AS8202NF chip would survive in such an environment. However, further investigation and testing need to be conducted.

EMC: Not shown. Among other disturbances in the wings of an airplane electro magnetic fields will induce disturbances in wires. It is probable that a multi drop bus will suffer more from this than point-to-point connections since the bus has a greater area to absorb disturbances.

TTP/C is independent of the physical layer which makes it possible to choose a physical layer that will meet the requirements in exposed environments.
The nodes need to have galvanic isolation from the communication bus to be robust against severe electromagnetic disturbances and lightning effects. It is not considered to be a problem to implement since TTP/C is physical layer independent.
A short-circuit that prevents all communication on a channel can be handled by having double redundant channels.

Implementation in an aircraft would in some cases mean that a bus would be up to 100 m long which can be accomplished using RS485.

Safety requirements:
The predefined communication schedule makes the communication predictable and hence easy to analyze. It guarantees latency and jitter which is a strong requirement for distributed control functions to be able to send and receive data in time. TTP/C is master less and uses fail silence which means that faulty nodes will not affect the rest of the communication on the network. This is ensured using bus guardians and membership service. Data consistency in the form of CRC is performed by the TTP-controller which means that the data consistency check is performed at controller level which saves host CPU power.

Bus-specific fault modes (presented in Section 6.2)
Short-circuit can be handled by using transformer coupler (galvanic isolation) at each stub of the bus. Redundancy in buses will tolerate short-circuit on the bus side of the transformers or a circuit cut-off. SOS-faults can be handled by TTP/C by forcing nodes that are considered faulty to go into a fail silent mode which means that the node cannot affect the communication on the bus, which has been shown in [18] .
Babbling idiot faults are handled by the bus guardians implemented in the TTP/C controller. The bus guardians have been formally verified as far as to the algorithms managing the slot timing. However, to achieve true fault tolerance the bus guardians would have to be implemented physically independent from the TTP/C controller. This has been done in the TTP/C star architecture where the bus guardian is centralized in the star coupler. It still remains an issue that has to be analyzed more in detail for bus.

The bus specific requirements in 3.5

1. The bus shall provide bounded (predictable) latency.
2. Adequate bandwidth for the application should be provided by the bus. It is common to have at least 50% of the bandwidth unused for future reconfigurations.
3. The probability of lost messages should be consistent with the probability classification of minor (see Table 2.1) under the interference of the specified environment.
4. The probability of undetected corrupted data should be consistent with the probability classification of hazardous (see Table 2.1), and the probability of successive occurrences should be consistent with the probability classification catastrophic under the interference of the specified environment.
5. The bus standard should support broadcast or multicast messages.
6. The bus should provide fault isolation mechanisms that provide controlled access to the bus. It is also desirable that the physical bus driver (controller) has a low failure rate.
7. The bus should use components that remain in the market for at least 10 years.
8. The physical layer should allow communication on the bus even if a node is removed.
9. The physical layer should be able to accommodate at least 30 nodes and have a bus length of at least 100m.

Table 10.1: TTP/C evaluated against the bus requirements

Requirement number	Pass	Fail	Need further analysis
1	X		
2			X
3			X
4			X
5	X		
6	X		
7	X		
8	X		
9	X		
10			X

10. The physical layer should be able to use transformer couplers to achieve galvanic isolation at each node.

Comments to results in Table 10.1
2, 3, 4 and 11 need to be evaluated for a specific system by building a demonstrator that has all nodes and wiring. Since this is rather large investment that would need a longer time budget it is not feasible to fit into this work.
6 see section on bus specific fault modes above.
7 TTTech has signed an agreement with their supplies Austria Microsystems that the AS8202NF controller should be available at least until 2030.
9 and 10 would need to be analyzed together with 2 to see that a satisfactory system configuration can be achieved.

11 Conclusions and future work

The main question that this thesis was aimed to answer is whether a TT broadcast bus communication (exemplified by TTP/C) is suitable for usage in safety-critical communication system in avionics (exemplified by a flight control system).

Based on the results and analysis (chapter 10) the conclusion is drawn that a broadcast bus using TTP/C would be suitable for use in a flight control system.

Regarding the protocols included for comparison; AFDX and FlexRay the conclusions is drawn that AFDX is not really suitable for a flight control system but interesting to show a different approach to TT communication. FlexRay would be an interesting alternative to TTP/C if an updated version with a more complete set of fault tolerance mechanism where to be released and the consortium would allow usage outside of the automotive industry.

Unfortunately the practical part of the evaluation was delayed due to hardware issue (described in chapter 11) which resulted in that not merely as many measurements as planned could be carried out. The contact with the hardware suppliers concerning availability and delivery time is extremely important in any project since the delays fundamentally affect the project and its outcome.

As a result of this thesis a lot of thoughts and ideas have been born. Some of the questions in this thesis remain unanswered and will need a significant amount of analysis and testing to be answered.

Future work for further evaluation of the bus architecture discussed in section 7.2 involves testing in a full scale network. Environmental stress testing of the components chosen is absolutely necessary for certification. To conducting safety-analysis of the bus guardian implementation in TTP/C to ensure that the system can be proved to meet a sufficient level of reliability for certification. Testing to decide whether RS485 can provide sufficient bandwidth in the bus configuration discussed in chapter 7 using transformer coupling.

Other interesting questions include:
- Large clusters divided in to multiple TTP/C networks
- TTP/C networks in parallel
- How a bus is affected differently by disturbances than a point-to-point i.e. by EMC – important aspect when there are severe conditions in a wing.

12 References

[1] TTTech, Time-Triggered Protocol TTP/C High-Level Specification Document Protocol Version 1.1, version 1.4.3 19 Nov 2003

[2] TTTech Computertechnik AG, Homepage of TTTech at www.tttech.com

[3] H. Kopetz, G. Bauer, The Time-Triggered Architecture, In *Proceedings of the IEEE Special Issue on modelling and Design of Embedded Software*, Oct 2002

[4] K. Driscoll, B. Hall, M. Paulitsch, P. Zumsteg, H. Sivencrona, The Real Byzantine Generals, In *Proceedings of the 23rd DASC*, Oct 24-28, 2004

[5] J. Rushby, A Comparison of Bus Architectures for Safety-Critical Embedded Systems. Technical report, Computer Science Laboratory, SRI 2003

[6] GAST project, The GAST project homepage at www.chl.chalmers.se/gast

[7] H Kopetz, Real-time Systems: Design Principles for Distributed Embedded Applications, Kluwer International Series in Engineering and Computer Science. Real-time Systems. 1997

[8] Jean-Claude Laprie, "Dependable Computing: Concepts Limits and Challenges", In *Proceedings of the 25th IEEE International Symposium on Fault-Tolerant Computing*, Pasadena, California, June 27-30, 1995, pp 42-54.

[9] K. Forsberg, NFFP3+ WP2 Certification guidance documents, Saab technical report NFFP-2006:002, 2006-04-12, (Not public)

[10] SAE ARP 4754, Certification Considerations for Highly Integrated or Complex Aircraft Systems

[11] DO-178B, Software consideration in Airborne Systems and Equipment Certification, RTCA inc, 1140 Connecticut Avenue, N.W. Suite 1020, Washington D.C. 20036

[12] DO-254, Design Assurance Guidance for Airborne Electronic Hardware, RTCA inc, 1140 Connecticut Avenue, N.W. Suite 1020, Washington D.C. 20036

[13] K. Forsberg, Analysis and Calculations for Dependable DIMA Architectures, Report Safety Critical issues (WP5) SD25101, Saab technical report, MOEL-2004:046, 2004-11-01, (Not public)

[14] D.A. Gwaltney, J.M. Briscoe, Comparison of Communication Architectures for Spacecraft Modular Avionics Systems, NASA Technical report, NASA/TM-2006-214431, June 2006

[15] K. Forsberg, Design Principles for Fly-By-Wire Architectures, PhD Thesis, Department of Computer Engineering

[16] SAE ARP 4761, Guidelines and methods for conduction the safety assessment process on civil airborne systems and equipment.

[17] TTTech, TTP-Plan manual, manual edition 5.3.8 11-Nov-2005

[18] H. Sivencrona, Heavy-Ion Fault Injection in TTP-C2 Implementation. Report of the SP Swedish National Testing and Research Institute, September 2003.

[19] John C. Knight and Nancy G. Leveson. An Experimental Evaluation of the Assumptions of Independence in Multi version programming. *IEEE Transactions on Software Engineering*, SE-12(1):96-109, January 1986.

[20] K Forsberg, B Habberstad, J Torin. FoT25 Systems Architecture WP6, Detailed DIMA and DTT UAV architectures. Saab technical report (Not public)

[21] IEEE, IEEE Recommended Practice for Architecture Description of Software-Intensive Systems, IEEE Recommended practice for architectural description of software-intensive systems, E-ISBN 0-7381-2519-9, ISBN 0-7381-2518-0

APPENDIX A. Definitions and terminology

A.1 Dependability terminology

The terminology and concepts of dependability used through out this thesis are adopted from [8] which are widely used among computer science researchers in the field of dependable systems. Laprie defines dependability as "that property of a computer system such that reliance can justifiably be placed on the service it delivers".
There are three main classes, impairments, means and attributes that are shown in the dependability tree in figure 2.1.

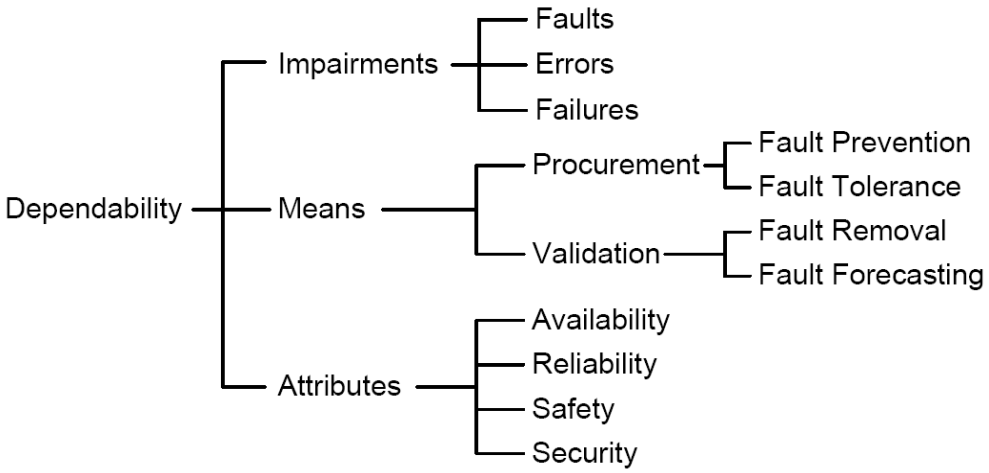


Figure A.1 Laprie's dependability tree

Attributes

The dependability attributes characterize the dependability of a given system.

Availability is a measurement of how probable it is that the system is operational and able to provide service at any given time. Higher availability means that the probability that the system can provide the requested service is higher.

Reliability is the measurement of whether a system can provide the intended service within the specified time, which makes the given response time accurate.

Safety is a measurement of how a system can provide service to its users without being a threat to its environment e.g. when performing services that it was not originally intended to.

Security is prevention of unauthorized access or handling of information.

Impairments

The impairments of a system are divided into faults, errors and failures. Although none of these are wanted in a system they are unavoidable. Fortunately there are ways to prevent and deal with these problems. A fault may if the part of the system where it occurred is activated lead to an error. An error might lead to other errors and if an error prevents the system from providing the intended service it leads to a failure. An error or a failure that stops all operation within a system is called system failure.

The chain can be illustrated as follows:

… → failure → [fault → error → failure] → fault → …

Means

The means for dependability are methods to increase the trustworthiness of a system. During the design process fault prevention aims at preventing faults from occurring and being introduced. It is however impossible to prevent all faults from occurring so the design must be made is such a way that faults can be tolerated and prevented from propagating to failures. This is accomplished through fault tolerance. Fault removal tries to deal with and minimized the effects of faults while fault forecasting estimates the probability and severity of faults. For more information about means see Chapter 4.

A.2 Definitions

A.2.1 General

Term	Definition
100BASE-TX	100BASE-TX is the predominant form of Fast Ethernet, providing 100 Mbit/s Ethernet.
Architecture	The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution [21].
Avionics	Aviation electronics
Certification	Is used here as short for the certification process that is applied to an aircraft before given permission to be used by the certification authorities.
Complex system	Complex refers to systems whose safety cannot be shown solely by test and whose logic is difficult to comprehend without the aid of analytical tools.
Control surface	Parts of an aircraft such as rudder, flaps and air brake that are used to affect the air stream to be able to control the manoeuvring of the aircraft.
Composability	The ability to build new systems from existing pieces, to run different applications on a platform composed from a pool of reusable system components.
Distributed system	A system where functions are distributed to several nodes.
Error	An error is a deviation from the required operation of the system due to a fault.
Failure	A system failure occurs when the system fails to perform its required function due to an error.
Fault	A fault is a defect within the system.
Fault Forecasting	Fault forecasting is conducted by performing an evaluation of system behaviour with respect to fault occurrence and activation.
Fault Prevention	Fault prevention is a technique that aims to prevent faults from entering the system in the design stage. It includes structured programming, information hiding and modularisation for software and rigorous design rules for hardware.

Fault Removal	Fault removal both tries to track and locate faults in a system before it enters service and after the system have been taken into service. This includes both hardware and software testing techniques.
Fault Tolerance	Fault tolerance is a method that aims to design a system so faults can be tolerated. If fault tolerance is not implemented a single fault may lead to global system failure.
Federated system	A system that is built to isolated from other systems to only carry out the system specific function. This is the opposite of an integrated system.
Hard real-time system	A real-time system where failure to complete its tasks before deadline expiration will have catastrophic consequences, i.e. a control loop.
Integrated system	A system where resources like a computational node are shared
Node	A node is a device connected to a communication network. It consists of a communication interface, a host computer and a buffer interface connecting the two.
RJ45	A standard connector with 8 pins used for 100BASE-TX among other Ethernet standards.
RS485	RS-485 (also known as EIA-485) is an OSI Model physical layer electrical specification of a two-wire, half-duplex, multipoint serial connection.
Predictability	Different methods to provide fault-tolerant services by making implicit or explicit assumptions about the behaviour of the system, so called failure modes.

A.2.2 Communication terminology

Term	Definition
Broadcast bus	A shared communication medium where all nodes can receive every transmission. All nodes are somehow physically connected to be able to receive the broadcast.
Channel	A channel provides a direct or switched point-to-point connection between communicating devices.
Collision	When two or more nodes on a network transmit simultaneously destroying all transmissions on that instant.
Event-triggered communication	In event triggered communication a node transmits on the occurrence of an event. If many nodes try to transmit at the same time there will be delays since only one can utilize the medium a time. When a collision occurs one node will have to retransmit later which renders in that delays become probabilistic. This is a big disadvantage since delays will vary with network load.
Jitter	Jitter is variation in delay between transmissions on a communication channel.
Latency Jitter	Variation in Latency
Latency	Latency is the time from the start of a transmission until the start of the transmission being received at the receiver.
Network	A network is a collection of processors and peripherals that interact with each other using a protocol. It can physically consist of a number of channels or a bus.

Overhead	Amount of data in a transmission that is not a part of the data sent but is needed by i.e. the communication protocol such as CRC, routing information etc.
Time-triggered communication	In time-triggered communication each node is allowed to send according to a predefined cyclic schedule. The communication is deterministic since unexpected events cannot occur as in event-triggered communication. The deterministic characteristics are crucial when i.e. performing a safety analysis. Since a node only is allowed to send in its time-slot network load will not affect the delays.
Clique formation	Part of a cluster i.e. the nodes in a cluster that interprets a message in a different way then the rest of the cluster due to some (TTP/C)
Macrotick	A periodic signal that delimits a granule of the global time. (TTP/C)
Microtick	A periodic signal that is generated by the oscillator of the controller. Each macrotick is made up of a number of microticks. (TTP/C)

A.3 Abbreviations

Abbreviation	**Description**
AFDX	Avionics Full-Duplex Switched Ethernet
ARINC	Aeronautical Radio Incorporated
ARP	Aerospace Recommended Practise
ASIC	Application-specific integrated circuit
AT	Action Time (TTP)
BC	Bus Controller (MIL-STD-1553)
BIT	Built in test
CAN	Controller Area Network
CMA	Common mode analysis
CNI	Communication Network Interface, TTP
CPCI	Compact Peripheral Component Interconnect
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
ECAC	European Civil Aviation Conference
ECC	Error Correction Code
EDAC	Error Detection Automatic Correction
EMC	Electromagnetic Compability
EMI	Electromagnetic Interference
EPROM	Erasable Programmable Read Only Memory
ESD	Electrostatic Discharge
ET	Event-triggered
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulations
FCC	Flight Control Computer
FCS	Flight Control System
FCU	Fault Containment Unit
FMEA	Failure Mode Effect Analysis
FT-COM	Fault Tolerant Communication (TTP)
FTA	Fault Tolerant Average algorithm, TTP

FTM	Fault Tolerant Midpoint algorithm
FTU	Fault Tolerant Unit, TTP
GAST	General Application Development Boards for Safety Critical Time-Triggered Systems
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IP	Industry Pack
JAA	Joint Aviation Authorities
JAR	Joint Airworthiness Requirements
IMA	Integrated Modular Avionics
MAC	Media Access Control
MBU	Multiple Bit Upset
MEDL	Message Descriptor List, TTP
MFM	The Modified Frequency Modulation (MFM) code is used for the bit synchronization and bit encoding/decoding
MIL-STD	Military Standard
MII	The Media Independent Interface is a standard interface used to connect a Fast Ethernet MAC-block to a physical layer
PL	Physical Layer
RAM	Random Access Memory
RTCA	Radio Technical Commission for Aeronautics
SEU	Single Event Upset
SOS	Slightly Out of Specification
TDMA	Time Division Multiple Access
TMR	Triple Modular Redundancy
TT	Time-triggered
TTCAN	Time-Triggered Controller Area Network
TTP/C	Time-Triggered Protocol automotive class C
TTP-OS	Time-Triggered Protocol Operation System
TDMA	Time Division Multiple Access
VL	Virtual Link
VME	Versa Module Europa / VERSAbus-E

APPENDIX B. Test cases

Test 1:	Basic functionality test – start-up
Goal:	To get the bus to start up and go into normal operation.
Purpose:	To evaluate that the configuration is correct and to prepare for subsequent tests.
Requirements:	All nodes should be able to send and/or receive messages.

Test 2:	Basic functionality test – timing
Goal:	To measure the timing physically on the bus to compare with the timing set in the schedule.
Purpose:	To examine the operation of TTP/C.
Requirement:	The physical communication on the bus should correspond with the communication schedule.

Test 3:	Fault injection
Goal:	To document the behaviour of the bus when faults like bit flips are introduced.
Purpose:	To evaluate the fault tolerating mechanisms of TTP/C.
Requirement:	The communication should no be affected by faults in one node.

Test 4:	Reintegration
Goal:	Get successful reintegration of a node after power off during operation.
Purpose:	To evaluate the fault tolerating mechanisms of TTP/C.
Requirement:	A node should resume normal operation without disrupting the other communication on the bus in case of i.e. a power failure

Test 5:	Shared responsibility between nodes (replication)
Goal:	Two nodes should have shared responsibility of a function. There should be no influence of one of the nodes go offline.
Purpose:	To test the redundancy management of TTP/C.
Requirement:	If one node in the pair goes offline the other should continue operation without any degradation of performance.

Test 6:	Time measurement of start up, reintegration and restart.
Goal:	To measure times for start up, reintegration and restart.
Purpose:	This data is important information when doing a safety analysis of a system.
Requirement:	To acquire reliable time values of start up, reintegration and restart.

APPENDIX C. Example cluster

The TTP/C architecture in Section 7.2 is used as a template for this example cluster. The TTTech tool-chain (TTP-Tools) is used to create bus scheduling and node software. Only one of the two redundant buses is implemented in this example. Every function has been designed in a subsystem to be easily distributed over a set of redundant nodes where needed. The FCC is triple-redundant (runs on three nodes) and sensors are double-redundant (runs on two nodes).
All the sensors send measurements to the FCCs. The actuators receive commands from the FCC. The actual output is sent back to the FCC as feedback to ensure proper operation.

A TDMA round of 6000 μs is used which gives a cluster cycle of 12000 μs; hence the messages will be sent with ~83 Hz. Message sizes in Table C.1 are from [15]
A short analysis of the schedule gives the conclusion that 6208 bits are sent every round and with 83.3 rounds per second the effective data send over the bus is 517 Kbit/s. From the schedule data we get that the total transmission rate on the bus is 1010 Kbit/s and transmission time is 78.9 % of that, 797 Kbit/s. This means that the synchronization information and overhead on the bus is approximately 35%. The reason for this is the high number of nodes on the bus and the fact that the FCC sends a lot more data than then other nodes (see Figure C.2). If the functionality of the FCC where to be distributed over a few small subsystems the communication is believed to be more efficient. This experiment however is left to future work.

Table C.1 contain message size, if feedback is used and the number of replicated nodes. Table C.2 contain schedule data. Figures C.3 and C.4 gives a graphical overview of the cluster communication schedule.

Table C.1: FCC cluster data

Function (Subsystem)	Message size (bit)	Feedback	Number of nodes
Sensors			
Air data	128 * 2	No	2
Rate gyro	128 * 2	No	2
Accelerometer	96 * 2	No	2
Cockpit	144 * 2	No	2
Interconnections	160	No	1
Actuators			
Actuator 1	144	Yes	1
Actuator 2	144	Yes	1
Actuator 3	144	Yes	1
Actuator 4	144	Yes	1
Actuator 5	144	Yes	1
Actuator 6	144	Yes	1
Actuator 7	144	Yes	1
Nose wheel	64	Yes	1
Air brake	64	Yes	1
Leading edge flaps	64	Yes	1
Engine	64	Yes	1
Flight control computer			
FCC	1264 * 3	No	3
Total	**6208**	**-**	**23**

Table C.2 FCC cluster: Schedule data

Number of nodes	23	**transmission time**	4735 µs (78.92%)
TDMA rounds per cluster cycle	2	**kilobits/second**	1010
TDMA round duration	6000 µs	**messages/second**	4500
Cluster cycle	12000 µs	**n-frames/second**	4333
stretch	575 µs (9.58%)	**i-frames/second**	7666
inter-frame gaps	690 µs (11.50%)		

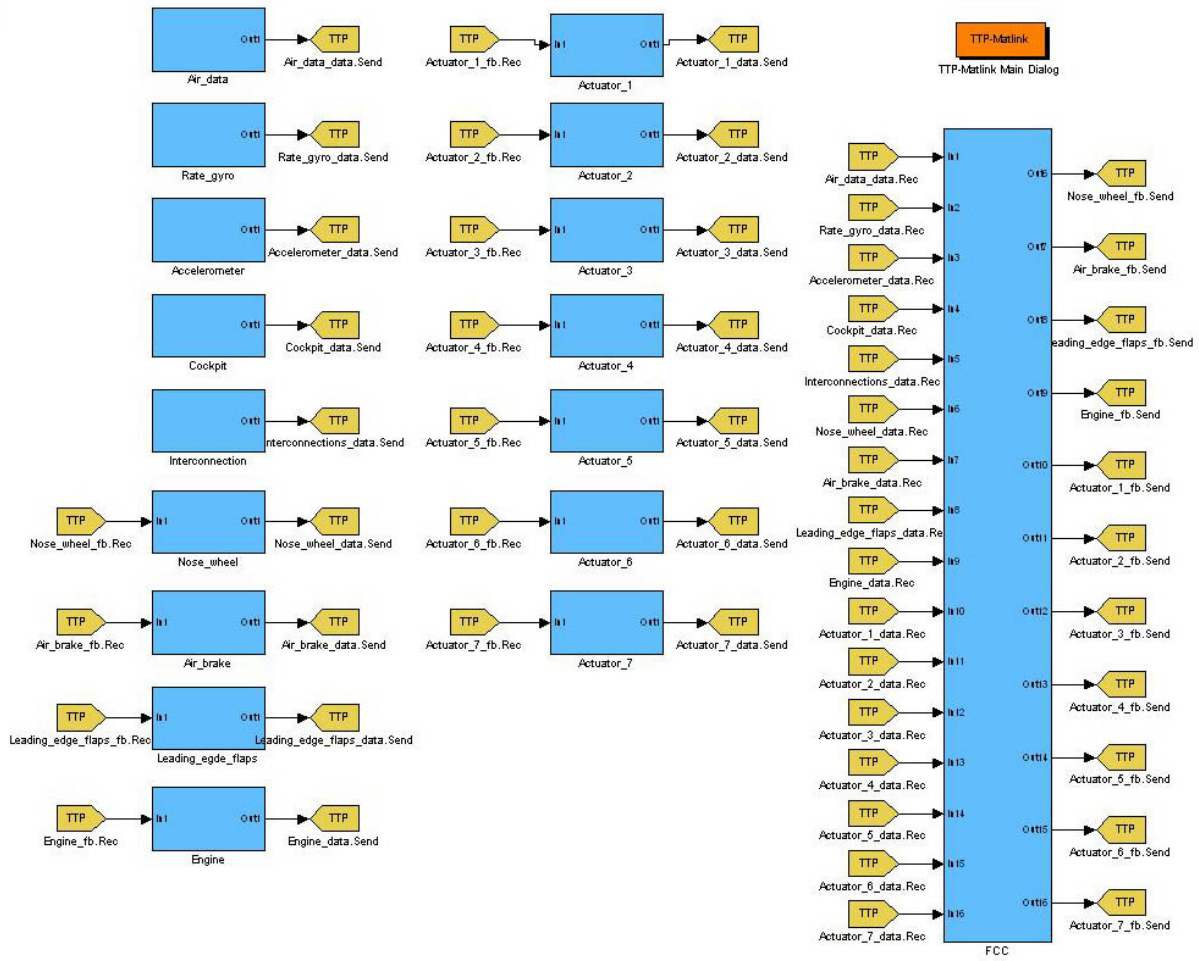


Figure C.1: The FCC cluster in TTP-Matlink

The Simulink model of the cluster is displayed in Figure C.1. Each block connected to a TTP-block represents a subsystem and the TTP-blocks represent messages, or arrays of messages in this case, that are sent over the communication bus. A subsystem can be distributed over several nodes and contain one or more tasks. TTP-Matlink is used to generate MEDL and node specific code for each node without the need for prewritten tasks which is very useful for prototyping.

The design steps in TTP-Matlink are (see Figure C.2):

1. Create a basic cluster design in Simulink including subsystems, tasks, messages and the mapping between subsystems and hosts, TDMA round and what kind of hardware target is used.
2. Run TTP-Plan to generate a cluster database and a schedule
3. Import the schedule parameter from TTP-Plan to TTP-Matlink
4. Generate node database, MEDL and TTP-OS code for each node with TTP-Build
5. Import data such as the task sample times generated by TTP-Build
6. Generate application code using Real-time workshop embedded coder

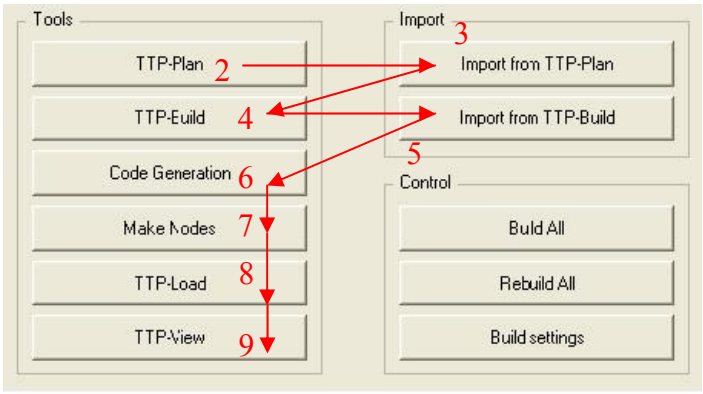


Figure C.2: Design steps in TTP-Matlink

7. Make nodes using the compiler of choice (I used Diab C compiler from Wind river)
8. Load the compiled code on to your embedded target over the TTP-bus using TTP-Load.
9. Use TTP-View to monitor the communication on the bus

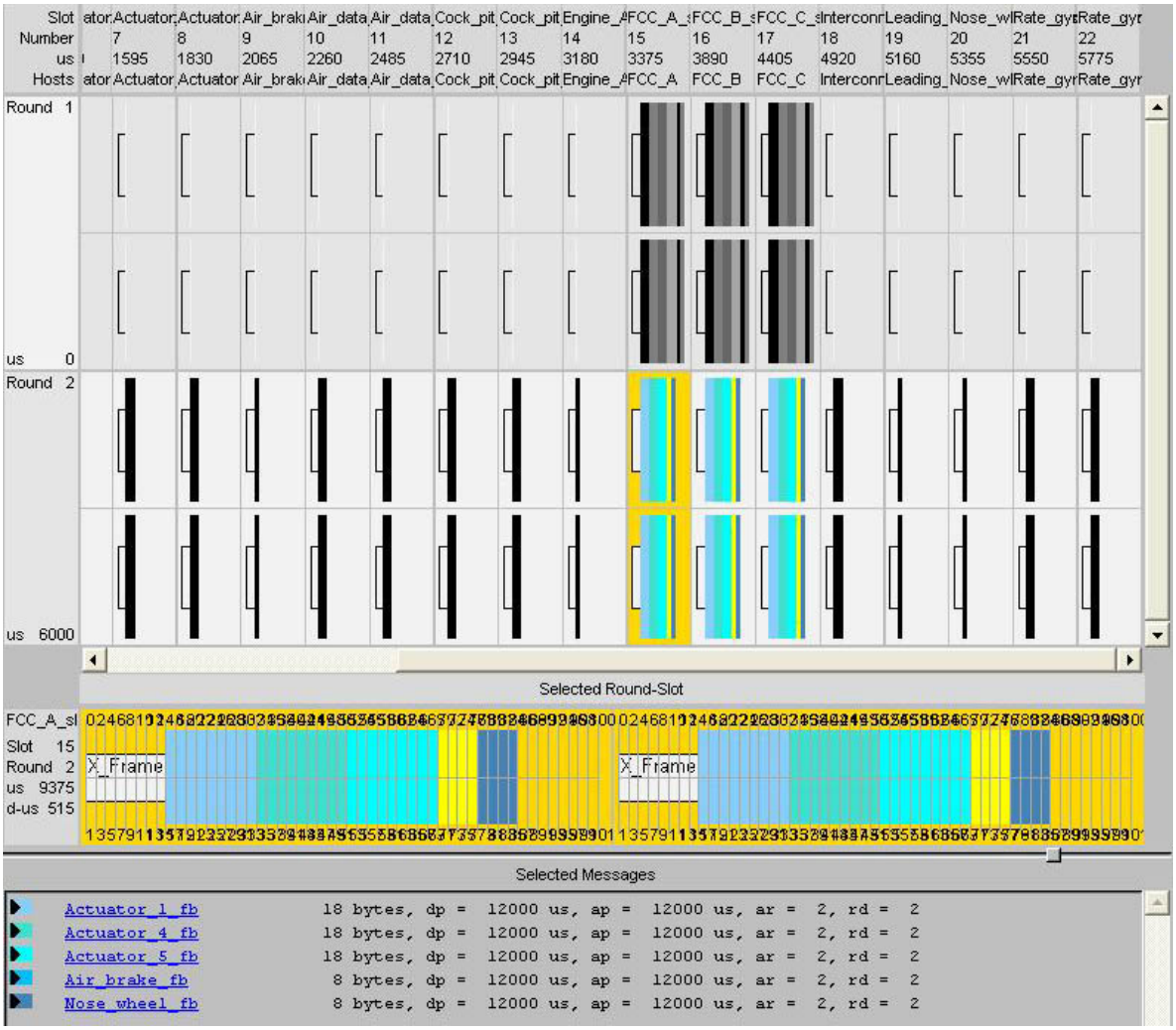


Figure C.3: Part of the schedule high lighting data transmitted by FCC_A in one TDMA round

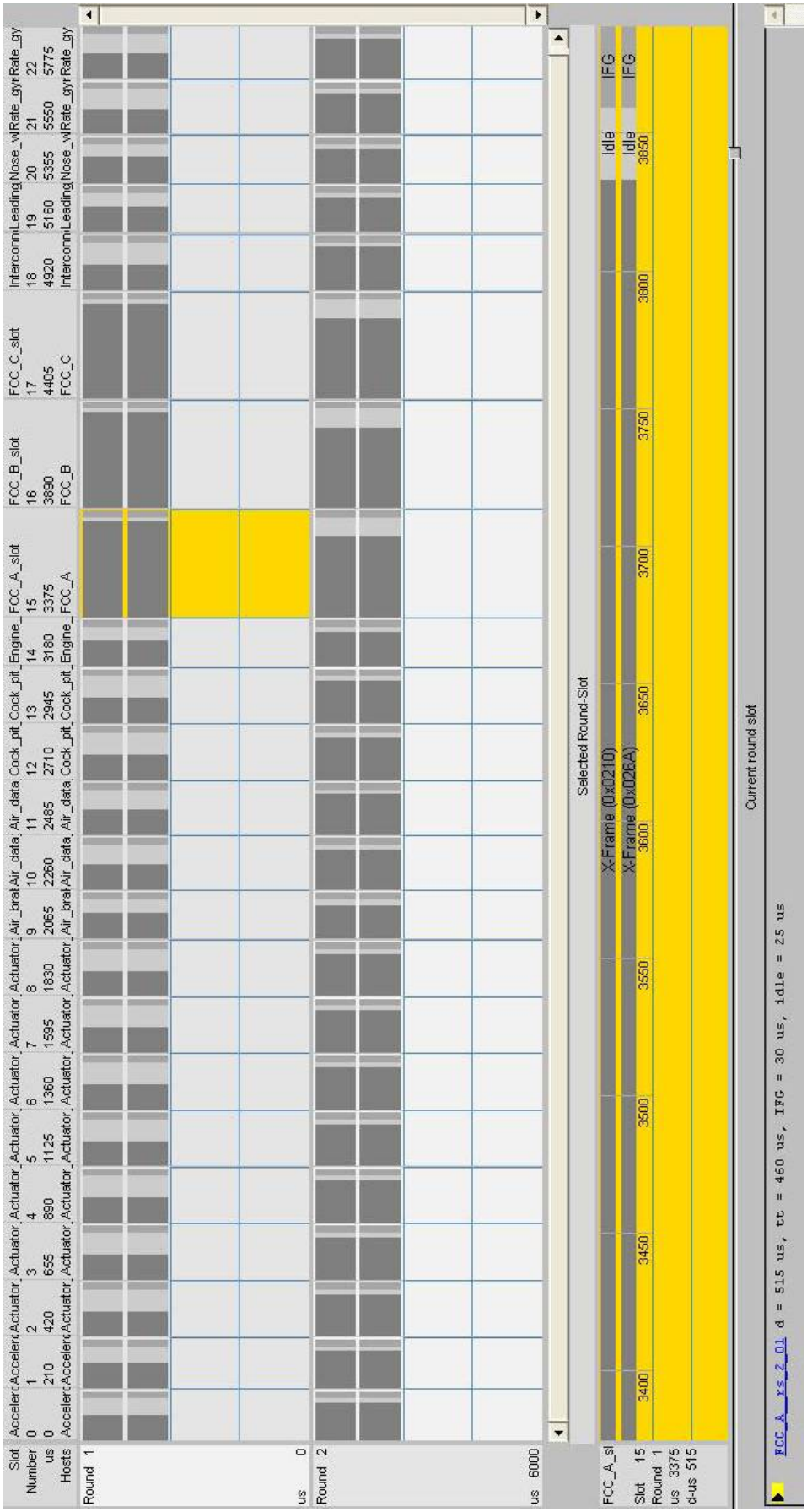


Figure C.4: Graphical overview of FCC communication schedule