MAINTAINING CONSISTENCY AMONG DISTRIBUTED CONTROL NODES

Kristina Forsberg, SaabTech AB, Jönköping, Sweden
Simin Nadjm-Tehrani, Linköping University, Linköping, Sweden
Jan Torin, Chalmers University of Technology, Göteborg, Sweden
Rikard Johansson, Saab Aerosystems, Linköping, Sweden

Abstract

This paper presents how state consistency among distributed control nodes is maintained in the presence of faults. We analyze a fault-tolerant semi-synchronous architecture concept of a Distributed Flight Control System (DFCS). This architecture has been shown robust against transient faults of continuous signals through inherent replica consistency [1]. This approach necessitates neither atomic broadcast nor replica determinism. Here, we extend the analysis of replica consistency property to confirm robustness against transient faults in discrete signals in presence of a single permanent fault in the DFCS components. The paper is based on a case study on JAS 39 Gripen, a modern fourth generation multi purpose combat aircraft, presently operating with a centralized FCS. Our goal is to design the DFCS fault management mechanisms so that the distributed treatment of faults corresponds to the existing non-distributed FCS.

1. Introduction

The consistency problem in distributed replicas is a well-known problem in aerospace control systems. Already the SIFT project [2] recognized and solved the problem with exact voting. In this paper we revisit the consistency issue in the context of a very different hardware architecture. The theoretical approach for redundancy management of fault-tolerant (FT) systems often calls for exact bit-wise consensus [3]. To achieve this the distributed nodes need to be strictly synchronized, and important primitives, such as membership agreement and atomic broadcast, are needed. For example the MARS system [4], with similar underlying hardware architecture, relies on membership service. Many algorithms have been developed to realize these primitives. However, protocols supporting membership agreement designed to increase dependability can exhibit brittleness against transient faults, and for example, increase the risk of excluding a correct node [5].

Synchronous communication ensures predictability in the time domain and enforces real-time requirements, but strict synchrony works against tolerating different views of the system state in the distributed nodes. In the distributed architecture, we have looked into a semi–synchronous approach where nodes can be temporarily inconsistent during short periods, but converge to the same view within a bounded time. With inherent replica consistency we mean that the nodes might not be exact replicas, continuous signals can be slightly different in the value domain and the mode status or discrete signals can be inconsistent during short, well defined, time intervals. The synchronization demands can in this way be relaxed and more robust because inconsistency among the actuator nodes will be tolerated during short periods. This conceptual solution has a great impact on the DFCS fault handling mechanisms. The adopted approach reduces overhead due to consensus at communication level, and allows well-known scheduling techniques for centralized nodes to be applied to the distributed nodes.

From early simulations presented in [1] it is found that the inherent replica consistency approach works well with continuous signals. However, the challenge is manifested when decisions due to discrete signals are to be taken. Hence, we must carefully analyze that the consistency property of system status is upheld among the distributed nodes.

A partial analysis was performed earlier [6]. In particular, we showed that whenever one control surface is disengaged due to a major fault, the other actuators reconfigure simultaneously within a

maximum time represented as a well-defined number of periodic cycles. This paper extends and details the included fault classes, in particular for the interface component, and completes the consistency analysis of DFCSs' fault handling mechanisms. Faults and fault combinations presented in [6] are included here for completeness of the picture.

The analysis is built up in two stages. First, overall safety requirements are stated in terms of desired properties of a distributed flight controller, with focus on discrete mode changes. For example, prescribing that a distributed controller acts in a similar way to the centralized one, when a major fault causes a flight control surface to disengage. This results in (informal) formulation of safety and bounded response properties. In the second stage, a careful analysis and listing of possible transient and permanent faults in every component of the architecture shows that no potential combination of single permanent and single transient faults violates the stated requirements.

Note that design faults are excluded from the class of permanent faults studied here. Several methods to reduce design faults, including formal verification are incorporated in the development process of safety-critical software and electronics under consideration here [7].

The paper is divided into 6 sectionss. Next section presents the hardware architecture. Section 3 outlines the system structure and fault model followed by the DFCS fault management mechanisms in section 4. Maintaining consistency in presence of faults is analyzed in section 5, and section 6 concludes the paper.

2. The Distributed FCS Architecture

JAS 39 Gripen has seven primary and three secondary control surfaces, all controlled by the FCS. In the distributed architecture, the critical sensor nodes and the bus are duplicated, while the seven actuator nodes are simplex, one at each primary control surface, see Figure 1. The reasons for studying a distributed solution compared to a centralized one are beyond the scope of this paper, but among the reasons one can mention: less weight, use of new technology in intelligent sensor and actuator nodes giving rise to redundant computational resources that can be used up this way, and finally fewer components leading to lower risk of breakdowns.

Each primary control surface can operate in one of two modes, the normal mode (fault free) and the streamlining mode (in presence of permanent faults). During normal mode the FCS controls the surface. In streamlining mode the surface is free to follow the aerodynamic forces affecting it. In this mode the surface will not add any lift force and will therefore have minor impact on the movement of the aircraft. The aircraft is still controllable and able to perform safe landing even when one primary control surface is streamlining.

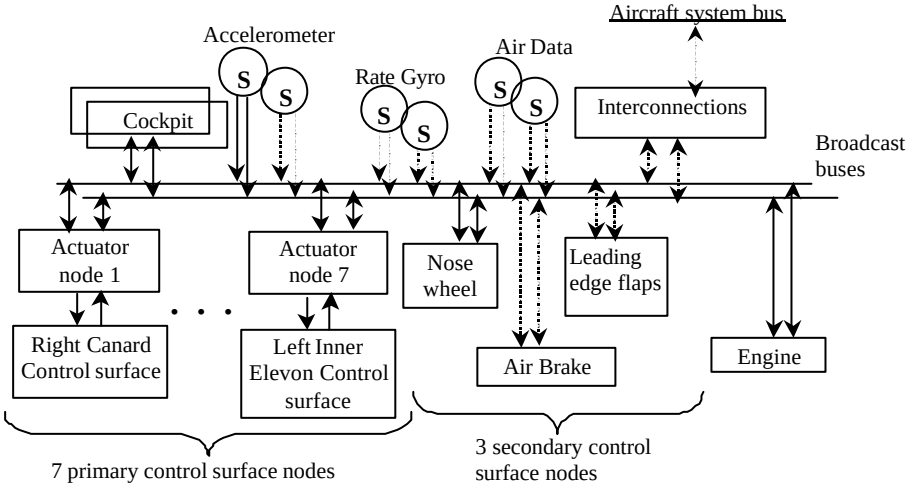

Figure 1. Sensor and Actuator Nodes of the Distributed FCS

Hardware replication (sensor and cockpit nodes, and bus) is required for the system to meet the safety requirements despite single permanent faults. This implies that no transient faults should lead to hardware losses. We will come back to this issue when considering the requirements imposed on the DFCS.

All control software of today's centralized flight control system is replicated at all seven actuator nodes in the distributed configuration, hence achieving a massive redundancy (seven redundant control computers compared to three of today's). The actuator nodes redundantly calculate all control commands and exchange them over the broadcast bus. Hence, each actuator has its own result plus the other actuators' results for comparison. Our thesis on distributed control nodes is as follows: Distributed control systems can be designed to be inherently consistent, meaning that transient faults can be recovered from within a bounded time limit.

The communication between the distributed nodes is synchronized using Time Division Multiple Access (TDMA) according to protocols as e.g. TTP/C, FlexRay or TT-CAN, while the actuator nodes are semi-synchronous, i.e. the nodes are synchronized every cycle by the bus but running asynchronously in the meantime. Thus, the nodes will not be strictly replica consistent but inherently replica consistent, where errors caused by transient faults can be tolerated and inherently recovered from. Below we will show how the consistency among distributed control nodes in presence of various faults can and will be maintained, thereby making them replica consistent.

3. System Structure and Fault Model

In the distributed FCS illustrated in Figure 1, all control and logic as well as fault handling mechanisms is allocated to the actuator nodes. The sensors can be viewed as data sources. Consequently, the following reasoning concerns the actuator nodes and their functions.

3.1 The Actuator Functions

The actuator, depicted in Figure 2, has one digital part (a computer) and one electro-mechanical part including servo and control surface. The digital part can experience both transient and permanent faults, whereas the electro-mechanical part experiences only permanent ones.

The digital part is divided into six functions that will be further discussed below: Interface, Sensor Adaptation and Fault Handling (AFH), Control Law Computation (CLC), Voter, Monitoring, and Loop Closure. In the distributed case, the CLC is identical to the present central FCS, and Monitoring and AFH are similar (but not identical). Hence, our goal is to verify that changes to the design, including the inherent replica consistency concept, lead to adequate fault handling with specific emphasis on the added Voter component.

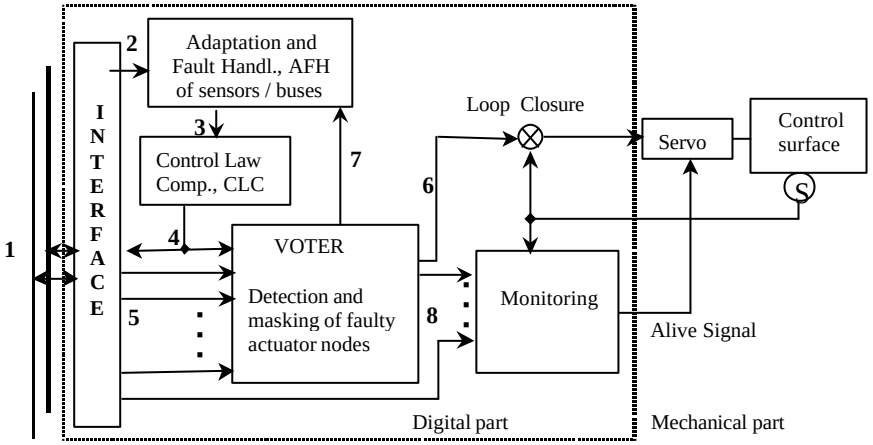

Figure 2. Simplified View of An Actuator

The interface component is of course important and will be analyzed in detail. Potential design changes in the loop closure and the electro-mechanical part will not be discussed below.

We now describe the digital functions in each box in more detail. The dataflow denoted by numbered arrows is described in next section.

Interface. The main purpose for the interface is to deal with the incoming and outgoing messages on the bus, in effect implementing the TDMA protocol. Particular for this analysis is the incoming sensor signals and the exchange of actuator messages.

Adaptation and Fault Handling (AFH). Here, adaptation of sensor signals are performed as well as detection and handling of faulty sensors. Knowledge of system and sensor's behavior is used to pinpoint a faulty sensor with high coverage.

Control Law Computation (CLC). This unit implements algorithms that perform stability and control computations. They change depending on which flight phase (e.g. landing, start etc.) the aircraft is currently performing. The JAS 39 Gripen aircraft can operate in nine different phases, one at a time. Depending on actual operating phase the CLC can operate in different modes, e.g. the pilot can choose to engage modes for holding the aircraft at a certain altitude, automatic aiming etc. In this paper we focus on the fault handling modes. –In particular, the reconfigured modes that might be selected in the CLC to compensate a streamlining surface.

Voter. The voter is a key element for the fault handling mechanisms of the DFCS and its purpose is twofold. First, for the continuous signals the voter algorithm selects one out of seven command words in each TDMA round by taking the mean value. In this way, faulty values are detected and masked, and erroneous command words are prevented from propagating to a control surface. Second, for the discrete signals, i.e. mode status, the algorithms will assure mode changes to be synchronous and the actuators' states consistent via exact (majority) voting and deferring the decision one cycle.

Monitoring. This component monitors the behavior of both the digital part and the electro mechanical part (using the control surface's position sensor, S in Fig. 2). It emits the Alive Signal that prevents the control surface from streamlining. As long as the monitoring qualifies the node as being healthy it issues the Alive Signal but if the node is not qualified the Alive Signal is not issued and the servo streamlines the control surface. The monitor function is as important as the Voter to achieve required FT, but is left out at this stage where focus is on consistency of the distributed Voters.

3.2 Communication and Data Flow

Next we explain the data flow into, within, and out of each actuator node. Messages from nodes not relevant for the analysis are left out.

In the beginning of each TDMA round, the duplicated sensors broadcast their messages, [S_1^1, …, S_n^1, S_1^2, …, S_n^2] on both buses (Figure 3). Message S_i^1 from sensors can hold values from continuous signals. Additionally the cockpit sensor also contains discrete signals such as selected mode as mentioned earlier. Actual mode is denoted as $mode_v^w$, where $v \in$ 1..Number_of_modes, and $w \in \{\eta, \xi\}$ to indicate normal or reconfigured CLC operation.

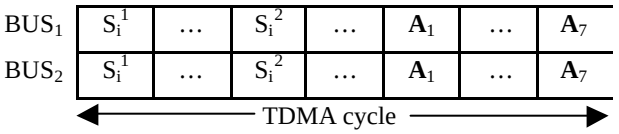

Figure 3. Messages Broadcast Every TDMA Round under Fault Free Condition

Each actuator node receives all sensor values and computes AHF and CLC, and subsequently exchanges information by broadcasting the messages, $\mathbf{A}_p$, p = 1..7 (Figure 3). Messages from an actuator node include continuous signals, the computed command words, υ_1-υ_7, from CLC and some discrete signals, in particular actual flight mode, $mode_v^w$, the Alive Signal, α, and the streamlining signal, ξ. For example a message from actuator node *p* in normal operation mode issuing streamlining is denoted by: $\mathbf{A}_p$: [υ_1- υ_7, $mode_v^\eta$, α, ξ_p, …].

In the paper we use $\varnothing$ to indicate a missing value, x^{faulty} for a faulty value of the variable in position *x*, and $\neg\xi$ for an unset streamlining signal.

The numbered arrows in Figure 2 show the data flow within a node and we limit the details on data flow to those signals that are important for the FT analysis later on. In Figure 2, *Arrow #1* illustrates all incoming and outgoing messages of the interface **M**:[…, m_k, …], k ∈ 1..Number_of_nodes. *Arrow #2*, input to AFH block, illustrates the duplicated set of received sensor values **S**: [S_1^1, …, S_n^1, S_1^2, …, S_n^2] and *arrow #3* is the computed sensor vector to the control law computation block, $\mathbf{S}_{CLC}$: [S_1, ..., S_n]. The double *arrow #4* coming out from the CLC block is this actuator's message, $\mathbf{A}_p$, which is both input to the own voter and broadcasted to other actuators' voters, typically carrying normal mode control commands, $\mathbf{A}_p$: [υ_1- υ_7, $mode_v^\eta$, α, $\neg\xi$, …]. The one-way groups of *arrows #5* into the voter represent the other six actuator messages. The result from the voting process, *arrow #6*, is a control surface's specific command word, $\mathbf{V}_{p\,out}$: [υ_p] and feedback, *arrow #7*, to AFH for actuator fault detection. *Arrows #8* into the monitoring block carries information for detection of permanent faults that must lead to streamlining, in particular faults in the Voter and the Interface components.

3.3 Fault Classes

Faults in the DFCS components are divided into following fault classes, a)-h). Sensor faults manifest as:

a) Permanently faulty sensor node results in wrong or missing value(s) in all actuators during all sequential TDMA rounds.

b) Transient fault in sensor node results in wrong or missing value(s) in all actuators during one TDMA round.

Faults at the communication bus can manifest as:

c) Permanently faulty bus, no messages can be transmitted.

d) Transient fault, e.g. EMI disturbance, affecting an ongoing message results in incorrectly received message in all or in a subset of actuators.

Faults in an actuator's communication interface can manifest as:

e) Permanently corrupted or no message / values (e.g. permanent fault in the sequence handler, timing and voltage faults can appear as Byzantine faults [5]).

f) Transiently corrupted message / value (e.g. bit flip in communication memory buffer, might cause inconsistency between affected and correct actuator nodes).

Faults in the actuator processor / memory can manifest as:

g) Permanent (all TDMA rounds): crash, omitted or value error.

h) Transient (one TDMA round): omitted or value error.

Faults in the actuator's electro mechanical parts lead to permanent disengagement of its control surface.

4. Fault Handling and Redundancy Management

We begin this chapter by stating the fault handling requirements on the DFCS as well as the assumptions used in the analysis.

4.1 The Fault Handling Requirements of DFCS

The analysis of adequacy of fault management in the DFCS, i.e. in the actuator nodes, depends on the high level requirements placed on the control system. In this section we list a number of major requirements that should be ensured by the distributed design.

Requirement 1: Neither a single nor a combination of two transient faults must lead to streamlining[1].

As indicated in the introduction, this requirement reflects the decision that no extra hardware components should be included for dealing with transient faults. Thus, transient faults should be tolerated by the fault handling mechanisms.

[1] Exception: A transient fault causing a state machine to behave as permanently faulty is not included in the requirement.

Requirement 2: The distributed actuator nodes behave as one with respect to the discrete signals, in particular the mode status.

This is an obvious requirement for any distributed control system (a kind of correctness requirement). However, to make it more concrete, we consider it in terms of a combination of the two following properties.

Requirement 2.1: Mode changes will be reflected in the control decisions taken by all actuator nodes. The decision will be taken simultaneously in all nodes within a given bound.

To be more precise, some permanent physical faults will cause a control surface to streamline. If this happens, it will be reflected in the control decisions taken by all remaining 6 actuator nodes. Moreover, this change takes place within a predefined number of cycles, and all actuators change control law in the very same TDMA cycle.

Requirement 2.2: If none of the control surfaces are streamlining, then none of the actuator node computations are carried out in the streamlining mode.

At the heart of these requirements lies the inherent consistency property (Req. 2). As mentioned earlier, one primary control surface streamlining is not a critical situation, the aircraft can still be well controlled and perform safe landing. From real flight experience it is known that reconfiguration in the case of a control surface streamlining is performed in a safe and correct way by today's centralized FCS. Additionally, for the distributed case we must ensure the inherent consistency property (Req.2). First the correct working actuator nodes must agree upon which surface is streamlining and secondly they must change mode simultaneously, within some time limit small enough not to jeopardize the stabilization of the aircraft, i.e. they reconfigure synchronously.

In Chapter 5 the FT mechanisms are analyzed to ensure that the distributed design does not violate the above identified requirements. The analysis is based on the detailed fault handling mechanisms in Section 4.2 and the following three assumptions.

1 No "babbling idiots": The nodes are fail-silent in the temporal domain.

2 Independent buses.

3 Very high fault/error detection coverage is assured through message synchronization mechanisms and CRC at all messages.

4.2 Actuator Fault Handling Mechanisms

In this section we present detection and handling mechanisms for the faults described in Section 3.3. All detection and handling is performed simultaneously within each actuator. Table 1 gives an overview of the mechanisms, and implicitly presents some dependencies that will appear in the analysis of distributed fault tolerance.

The DFCS cannot recover from permanent faults during runtime, instead the infected area or node is lost, giving a redundancy loss of the system. Permanent faults of sensors and buses are tolerated by hardware redundancy and the system impact of such faults is redundancy loss. A fault in a sensor node can result in either a) the node's fault detection mechanisms discover the fault and report this in its next broadcast message or b) the fault is not detected in the sensor and an erroneous value is then broadcasted on the bus. Erroneous input sensor values will be detected and isolated in the AFH by comparison e.g., assertion checks [8], range, min / max derivate etc.

Most permanent actuator faults, e.g. interface, voter, monitoring, servo, control surface, (monitor, voter and interface programs are checked by checksum calculation and coded variables.) must lead to streamlining of the affected control surface followed by a reconfiguration, by which the remaining six control surfaces must compensate for the missing surface.

Table 1 summarizes the fault-handling mechanisms of the DFCS. As the table shows, there are 5 faults (rows) in which the system will resort to reconfigured mode to compensate for a streamlining control surface. These fault-handling scenarios will be further considered in next chapter.

Table 1. DFCS Actuator Fault Handling

DETECTION MECHANISM	FAULT HANDLING MECHANISM	SYSTEM EFFECT
Sensor, fault class a) and b)		
Wrong sensor value – detected in the AFH.	Exclude faulty sensor value in CLC	If transient fault: None
Missing sensor value – detected by bus protocol	Use redundant sensor value	If permanent fault: Redundancy loss
Bus, fault class c) and d)		
Destroyed messages detected by CRC	Use messages from redundant bus	If transient fault: None
Nothing or noise on one of the buses detected by bus protocol	Switch to duplicate bus	If permanent fault: Redundancy loss
Actuator node, fault class e)-h)		
Communication Interface, fault class e) and f)		
Corrupted message(s) detected by CRC	Masked by voter	None
No messages – detected by bus protocol	Streamlining	Reconfiguration
Processor, fault class g) and h)		
Computing fault (faulty operating calculation units) detected by comparison in voter	Reuse previous states in CLC, masked by voter	None
No results (crash) – detected by bus protocol	Streamlining	Reconfiguration
CLC (memory), fault class g) and h)		
Wrong result (bit flip in program or data) detected by comparison in voter	Reuse previous states in CLC, masked by voter	None
Program crash detected by exception in CLC	Masked by voter	None
Voter (memory), fault class g)		
Wrong result (permanent bit flip in program) detected by monitoring	Streamlining	Reconfiguration
Monitoring (memory), fault class g)		
Wrong result (permanent bit flip in program or states) detected by existing mechanisms (not public)	Streamlining	Reconfiguration
Control Surface (Electro mechanical part)		
Not correct working control surface is detected by monitoring and comparison with position sensor	Streamlining	Reconfiguration

5. Analysis of the FT Mechanisms of Actuator Nodes

The system is not designed to cover two arbitrary permanent faults. Hence, in the current analysis we exclude combinations of permanent faults, but consider the possibility of transient and permanent faults appearing together. Before covering the separate cases, we give an abstract description of the distributed algorithm, to clarify how faults may affect its computations in different phases. DFCS Algorithm behaves as follows in each TDMA round:

Communication Phase 1: All sensor nodes broadcast their values, which are received by all actuator nodes.

Processing Phase 1: Actuator nodes apply received/stored sensor values on current states and perform AFH and CLC.

Communication Phase 2: Actuator nodes exchange results by broadcasting their messages.

Processing Phase 2: Actuator nodes perform voting on everybody's results.

In presence of faults, this algorithm does not include any explicit consensus procedures. Instead, we will show that the distributed voters will decide on the same mode in the presence of one or two faults. In the following discussions we will go through single transient and permanent faults (as presented in Section 3.3), as well as combination of faults that might cause the seven voters to come to different states followed by the treatment in order to re-establish the consistency.

5.1 Single Faults

We begin our analysis by considering single faults. Table 2 and 3 embrace all faults, a)-h), listed in Section 3.3, where Table 3 explains reconfiguration for single faults leading to streamlining. The first column in Table 2 describes the faults. The second column shows which vector/vectors that are affected, and especially the voters' input vectors, $\mathbf{A}_p$, p = 1..7.

Table 2. Single Faults

FAULT DESCRIPTION	ERROR MANIFESTATION	COMMENTS
Wrong or missing value from sensor S_i^1 Fault class a) and b) in §3.3	BUS_1 and BUS_2 broadcast: $\mathbf{S}$: […, S_{i-1}^1, ∅, S_{i+1}^1…, S_i^2, …,] or $\mathbf{S}$: […, S_{i-1}^1, S_i^{faulty}, S_{i+1}^1…, S_i^2,…,]	Inconsistency cannot happen since all actuator nodes receive the same sensor values.
Permanent faulty bus, no messages or noise from the faulty bus (here exemplified with BUS_2) Fault class c) in §3.3	$\mathbf{M}_{BUS1}$: […,m_{k-1}, m_k, m_{k+1}, …] $\mathbf{M}_{BUS2}$: [∅] or for all k; [m_k^{faulty}]	Inconsistency cannot happen since all actuator nodes receive the same messages.
Corrupted message, m_k, from transient disturbance in one bus (here BUS_2) during *Comm. Phase 1* or *Comm. Phase 2* Fault class d) in §3.3	$\mathbf{M}_{BUS1}$: […,m_{k-1}, m_k, m_{k+1}, …] $\mathbf{M}_{BUS2}$: […,m_{k-1}, m_k^{faulty}, m_{k+1}, …]	Inconsistency cannot happen since all actuator nodes receive messages on the duplicated correct bus.
Faulty interface in actuator node p affecting incoming messages, $\mathbf{S}$ or $\mathbf{A}$ Fault class e) and f) in §3.3	p receives $\mathbf{S}'$, $\forall\, q \neq p$ q receives $\mathbf{S}$ or p receives $\mathbf{A}_q'$, $\forall\, q \neq p$ q receives $\mathbf{A}_q$	Actuator p will perform the CLC on a different set of sensor values ($\mathbf{S'}_{CLC}$) Inconsistency cannot happen since all actuator n odes receive the same actuator messages, or p vote on different set of actuator messages. Masked by voter.
Faulty interface in actuator node p affecting outgoing message, $\mathbf{A}_p$ Fault class e) in §3.3	$\mathbf{A}_p$ is received arbitrary as [m_p] or [m_p^{faulty}]	Weak bus-drivers might result in a transmitted "1" or "0" are received differently, A_p is seen faulty by a subset of the actuator nodes. Masked by voter.
Transient fault during *Proc. Phase 1* in actuator node p Fault class h) in §3.3	$\mathbf{A}_p$:[u^{faulty}, …,$mode_v^{\eta}$, α, ¬ξ, …] $\forall\, q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ¬ξ, …]	All voters vote on 6 correct vectors and one faulty (p's).
Transient fault during *Proc. Phase 2* affecting the Voter in actuator node p Fault class h) in §3.3	$\mathbf{V}_{p\,out}$: [υ_p^{faulty}] $\forall\, q \neq p$ $\mathbf{V}_{q\,out}$: [υ_q]	The faulty command word in A_p propagates to the loop closure, this is OK for shorter periods due to the inertia of the aircraft, and does not influence the voter's decisions.
Transient fault during *Proc. Phase 2* affecting the Monitoring in actuator node p, Fault class h) in §3.3	As in the present FCS (Saab restricted information).	Has no impact on decisions in the voters.

Permanent sensor and bus faults are treated by hardware replication while permanent actuator faults lead to a degraded operating mode and CLC reconfiguration, see Table 1. The question is how does the DFCS deal with single permanent actuator faults? That is, how do we ensure that the actuator nodes reflect the streamlining decision by one node in the future behavior of all remaining nodes (Req. 2.1). Decision on streamlining a control surface is taken locally by the affected actuator node. Table 3 analysis permanent actuator faults in which the system resorts to streamlining. Two types are recognized:

Type A: the actuator node does not detect it is faulty, e.g. faulty interface and message omission (or processor crash)

Type B: the actuator node detects it is faulty and issues streamlining.

From this first assessment we can recognize that single transient or permanent fault will not violate the consistency requirement (Req.2) and a single transient fault will not lead to streamlining.

However, a Byzantine behavior from one of the nodes, causing inconsistency in identifying a faulty node is not resolved by the DFCS algorithm nor by the fault handling mechanisms. The fault is tolerated by majority voting. (With seven nodes, two such faults can be tolerated.)

For permanent faults that lead to streamlining the analysis shows that all seven voters take the decision on reconfiguration simultaneously and within a certain number of cycles. In next sections, we show that this property (reconfiguration in the same cycle) will not be affected by a concurrent fault.

Table 3. Single Permanent Faults Enforcing Streamlining

SINGLE PERMANENT ACTUATOR FAULTS ENFORCING A CONTROL SURFACE TO STREAMLINE		
Type A. No message from actuator node p due to permanently faulty communication interface. Fault class e) in §3.3 Same error manifestation (silent) as processor crash, Fault class g) in §3.3		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{M}_{\mathbf{A}p}$: [∅]	$\mathbf{M}_{\mathbf{A}p}$: [∅]	$\mathbf{M}_{\mathbf{A}p}$: [∅]
$\forall\, q$	$\mathbf{A}_p$: [υ_1-υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]	$\mathbf{A}_p$: [υ_1-υ_7, $mode_v^{\xi}$, ∅, ξ_p, …]
$\mathbf{A}_q$: [υ_1-υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]	$\forall\, q \neq p$ $\mathbf{A}_q$: [υ_1-υ_7, $mode_v^{\eta}$, α, ξ_p, …]	$\forall\, q \neq p$ $\mathbf{A}_q$: [υ_1-υ_7, $mode_v^{\xi}$, α, ξ_p, …]
Cycle j: The control law computation is in normal mode, $mode_v^{\eta}$, in all actuator nodes, $A_{1\text{-}7}$. A_p cannot communicate on either one of the buses. **Cycle j+1**: The DFCS is still operating in normal mode. Since A_p is silent the correct working actuators issue the streamlining flag of actuator p, ξ_p. A_p finds out that the other actuators believe it is faulty and must disengage its control surface. **Cycle j+2**: CLC is computed in reconfigured mode, $mode_v^{\xi}$, in the six correct working actuators. Hence, reconfiguration takes place in the same TDMA cycle i.e. streamlining and synchronous reconfiguration.		
Type B. In Cycle j a permanent fault in actuator node p causes its control surface to streamline Fault class g) in §3.3		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{A}_p$: [υ_1-υ_7, $mode_v^{\eta}$, ∅, ξ_p, …]	$\mathbf{A}_p$: [υ_1-υ_7, $mode_v^{\eta}$, ∅, ξ_p, …]	$\mathbf{A}_p$: [υ_1-υ_7, $mode_v^{\xi}$, ∅, ξ_p, …]
$\forall\, q \neq p$	$\forall\, q \neq p$	$\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1-υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]	$\mathbf{A}_q$: [υ_1-υ_7, $mode_v^{\eta}$, α, ξ_p, …]	$\mathbf{A}_q$: [υ_1-υ_7, $mode_v^{\xi}$, α, ξ_p, …]
Cycle j: The control law computation is in normal mode, $mode_v^{\eta}$, in all actuator nodes, $A_{1\text{-}7}$, and A_p flags for streamlining, ξ_p thus, stopped issuing the Alive Signal, α, (seen by ∅). **Cycle j+1**: The DFCS is still operating in normal mode, but all actuators have now recognized that control surface p is streamlining, ξ_p, and due to majority decision in the voter the next computation will be in reconfigured mode. **Cycle j+2**: CLC is computed in reconfigured mode, $mode_v^{\xi}$, in all actuators. Hence, reconfiguration takes place in the same TDMA cycle i.e. streamlining and synchronous reconfiguration.		

5.2 Concurrent Transient Faults

In this section we show (Table 4) that combinations of two transient faults alone can neither lead to streamlining (Req. 1) nor inconsistency with respect to discrete signals (Req. 2.1).

Table 4. Combination of Transient Faults

CONCURRENT FAULTS	ERROR MANIFESTATION	COMMENTS
Double sensor faults, fault b): Wrong or missing sensor value in sensor, S_i^1, and in sensor, S_j^2	$\mathbf{S}$: […, S_{i-1}^1, S_i^{faulty}, S_{i+1}^1, …, S_{j-1}^2, S_j^{faulty}, S_{j+1}^2…,]	Inconsistency cannot happen since all actuator nodes receive the same sensor values. However, if i = j a mode change might be missed during this TDMA cycle if affected signal was to enforce one.
Double bus faults, fault d): Corrupted message, m_k, on Bus 1 and, m_l, on Bus 2	$\mathbf{M}_{BUS1}$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...] $\mathbf{M}_{BUS2}$: […, m_{l-1}, m_l^{faulty}, m_{l+1}, ...]	k≠l, correct message taken from the unaffected bus. k=l, not possible, see assumption under Section 4.1
Double actuator faults, fault f): Faulty interface units in node *p* and *q* affecting incoming and / or outgoing messages.	*p* receives $\mathbf{S}'$ and/or $\mathbf{A}_q'$, q≠p or transmits [m_p^{faulty}] *q* receives $\mathbf{S}''$ and/or $\mathbf{A}_p''$, p≠q or transmits [m_q^{faulty}]	Most cases: All voters vote on 5 correct vectors and two faulty, no inconsistency. Worst case: *p* and *q* have different numbers of vectors into their voters, inconsistent (this TDMA cycle) to the five correct nodes with 7 vectors.
Double actuator faults, fault h): Faulty operating calculation units in actuator node *p* and *q*	$\mathbf{A}_p$, $\mathbf{A}_q$: […, υ^{faulty}, …], $p \neq q$	All voters vote on 5 correct vectors and two faulty.
Concurrent sensor and bus faults: S_i^1 sends faulty value and m_k is corrupted on bus one	$\mathbf{S}$: […, S_i^{faulty}, …, S_i^2, …] $\mathbf{M}_{BUS1}$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...]	Handled by duplication in sensors and buses (see Table 1).
Concurrent sensor and actuator faults: S_i^1 and $\mathbf{A}_p$ contains faulty values on both buses	$\mathbf{S}$: […, S_i^{faulty}, …, S_i^2, …] $\mathbf{A}_p$: […, υ^{faulty}, …] or [$m_{actuator\,p}^{faulty}$]	Single sensor faults are handled by AFH. All voters vote on 6 correct vectors and one faulty.
Concurrent actuator and bus faults: $\mathbf{A}_p$ contains faulty values and m_k is corrupted on bus one	$\mathbf{A}_p$: […, υ^{faulty}, …] or [$m_{actuator\,p}^{faulty}$] $\mathbf{M}_{BUS1}$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...]	Single bus faults are handled by duplication. All voters vote on 6 correct vectors and one faulty.

This analysis shows that Req 1 (No combinations of transient faults lead to streamlining) is satisfied by the DFCS architecture.

5.3 Combined Permanent and Transient Faults

Here we show that the actuator nodes of the DFCS behave as one with respect to the discrete signals also for combination of transient and permanent faults (Req. 2.1). The analysis leaves out permanent faults that result in redundancy loss only and focus on those treated with mode change, i.e. reconfiguration. The single permanent actuator faults in Table 3 are combined with the transient faults from Table 2 and analyzed in Table 5.

The combined cases are:

Case I: Actuator node *p* permanently faulty and in the same TDMA cycle is sensor, S_i^1, affected by a transient.

Case II: Actuator node *p* permanently faulty and in the same TDMA cycle message, m_k, is corrupted during *Communication Phase 1* or *2*.

Case III: Actuator node *p* permanently faulty and in the same TDMA cycle a transient fault affect in or out going messages of an interface.

Case IV: Actuator node *p* permanently faulty and in the same TDMA cycle occurs transient computation faults during *Processing Phase 1* in actuator *q*. Here we analyze a computation fault affecting the discrete signal ξ.

Table 5. Combined Permanent & Transient faults

<table>
<tr><td colspan="3">Case I In Cycle j: Actuator p issues streamlining (or is silent) and Sensor fault in S_i^1</td></tr>
<tr><td>TDMA cycle j</td><td>TDMA cycle j+1</td><td>TDMA cycle j+2</td></tr>
<tr><td>$\mathbf{S}$: […, S_{i-1}^1, S_i^{faulty}, S_{i+1}^1, …, S_i^2, …]
$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$: $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\xi}$, α, ξ_p, …]</td></tr>
<tr><td colspan="3">Cycle j to j+2: Sensor faults does not affect the ξ signal (only υ values corrected by AFH) hence, Case I is reduced to the single permanent fault case in Table 3.
For Type A faults ξ_p is set in cycle j+1 because actuator node p is silent instead of is suing streamlining.</td></tr>
<tr><td colspan="3">Case II In Cycle j: Actuator p issues streamlining (or is silent) and message m_k, is corrupted on Bus 1</td></tr>
<tr><td>TDMA cycle j</td><td>TDMA cycle j+1</td><td>TDMA cycle j+2</td></tr>
<tr><td>$\mathbf{M}_{BUS1}$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...]
$\mathbf{M}_{BUS2}$: […, m_{k-1}, m_k, m_{k+1}, ...]
$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\xi}$, α, ξ_p, …]</td></tr>
<tr><td colspan="3">Cycle j to j+2 : Bus faults are corrected by the duplicated bus and Case II is reduced to the single permanent fault case in Table 3.
Again, for Type A faults ξ_p is set in cycle j+1 because actuator node p is silent instead of issuing streamlining.</td></tr>
<tr><td colspan="3">Case III In Cycle j: Actuator p issues streamlining (or is silent) and actuator q's interface is faulty, with following three consequences: i) Node q receives $\mathbf{S}'$ ii) q receives $\mathbf{A}_p'$, p≠q, and iii) q transmits [m_q^{faulty}]</td></tr>
<tr><td>TDMA cycle j</td><td>TDMA cycle j+1</td><td>TDMA cycle j+2</td></tr>
<tr><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
i) and ii) $\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]
iii) $\mathbf{A}_q$: [m_q^{faulty}]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
i) $\mathbf{A}_q$: [υ'_1- υ'_7, $mode_v^{\eta}$, α, ξ_p, …]
ii) $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, $\neg\xi$, …]
iii) $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, $\varnothing$, ξ_p, …]
i) $\mathbf{A}_q$: [υ'_1- υ'_7, $mode_v^{\xi}$, α, ξ_p, …]
ii) and iii) $\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\xi}$, α, ξ_p, …]</td></tr>
<tr><td colspan="3">Cycle j to j+2 : i) the computed command words will be slightly different, u'$_q$, this will sustain in the system
ii) node q has a reduced number of actuator vectors to vote on in cycle j+1, might miss p issuing streamlining
iii) node q votes on 7 correct vectors, the others on 6 in TDMA cycle j
All versions are reduced to the single permanent fault case in Table 3,
Type A iii) node q votes on 6 correct vectors, the others on 5 in cycle j</td></tr>
<tr><td colspan="3">Case IV Actuator p issues streamlining (or is silent) and actuator q has faulty discrete signal i.e. q flags for streamlining, ξ_q.</td></tr>
<tr><td>TDMA cycle j</td><td>TDMA cycle j+1</td><td>TDMA cycle j+2</td></tr>
<tr><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
$\exists$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_q, …]</td><td>$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, $\varnothing$, ξ_p, …]
$\forall\, q \neq p$
$\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]</td><td>$\forall\, p$
$\mathbf{A}_p$: […, $mode_v^{\xi}$, …]</td></tr>
<tr><td colspan="3">Cycle j to j+2: In this case two actuators flag for streamlining simultaneously, however only one control surface is actually streamlining. The voter logic will mask this case by ignoring the ξ_q since the Alive Signal, α, (also exchange by the actuators) is present in A_q. Thus, also this case will resort to the single permanent one and reconfigure in three cycles. Same scenario (the voter logic ignore ξ_q since α is present) for the Type A faults.</td></tr>
</table>

The permanent faulty actuator node *p* in *Case I-IV* can be of type *A* – cannot communicate with other actuator nodes, or type *B* – issuing streamlining of its control surface. Both Type A and Type B are included in the analysis in Table 5, where $\mathbf{A}_p$ under TDMA cycle j to j+2 belongs to Type B faults, and dissimilarities to Type A are explained in the commentary field under the three cycles.

This analysis shows that Req. 2.1 is met even in presence of combination of permanent and transient faults. As a corollary we get the satisfaction of Req. 2.2 as the final case of the analysis, *Case IV*.

6. Concluding Remarks

The analysis of the DFCS even as an informal reasoning process has not been a trivial task. Having done this analysis, we have completed the replica consistency property that was initiated in [6] for discrete signals, and in [1] covering continuous signals. In this paper, we have concentrated on presenting the likely fault scenarios and the essential fault handling mechanisms that ensure a correct distribution of the flight control function. In particular, we showed that the semi-synchronous approach with inherent replica consistency is robust against transient faults and distributed treatment of faults corresponds to the centralized FCS, i.e. the DFCS behaves as one. The correctness of the implemented distribution with respect to well-defined combinations of transient and permanent faults (that might affect continuous or discrete values) has been shown and is the major contribution of this paper. Thus, a valuable input to the system safety and reliability analysis has been rigorously documented. Future works include detailing the design, especially the voter, at a formal level where the analysis can be checked by employment of formal verification tools.

7. Acknowledgement

This article is partially supported by projects in the Swedish national aerospace program NFFP 428, 436 and the project SAVE, supported by the Swedish foundation for Strategic Research (SSF).

References

[1] Ahlström Kristina, Jan Torin, Krister Fersán, Per Nobrant, 2002, "Redundancy Management in Distributed Flight Control Systems; Experience & Simulations", in proceedings of AIAA and IEEE 21th Digital Avionics Systems Conference, Irvine, CA.

[2] Wensley J.H. *et al.*, 1978, "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control", In proceedings of IEEE, Vol. 66, pp 1240-1255.

[3] Lala J. H., R. E. Harper, 1994, "Architectural Principles for Safety-Critical Real-Time Applications", In proceedings of IEEE, vol. 82, pp 25-40.

[4] Kopetz Herman, *et al.*, 1990, "Tolerating Transient Faults in MARS", In proceedings of 20th Symposium on Fault-Tolerant Computing, pp 466-473.

[5] Sivencrona Håkan, Per Johannessen, Jan Torin, 2003, "Protocol Membership in Dependable Distributed Communication Systems – A Question of Brittleness", SAE World Congress paper No. 2003-01-0108, Detroit.

[6] Forsberg Kristina, Simin Nadjm-Tehrani, Jan Torin, 2005, "Fault analysis of a distributed flight control system", Submitted to HICSS38, Hawaii, USA.

[7] Hammarberg J. and S. Nadjm-Tehrani, 2004, "Formal Verification of Fault Tolerance in Safety-critical Modules", Software Tools for Technology Transfer Journal, Springer Verlag. To appear.

[8] Hiller Martin, 2000, "Executable Assertions for Detecting Data Errors in Embedded Control Systems", In proceedings of IEEE International Conference on Dependable Systems and Networks, pp. 24-33.

.