Fault Analysis of a Distributed Flight Control System

Kristina Forsberg
SaabTech AB
kristina.forsberg@saabtech.se

Simin Nadjm-Tehrani
Linköping University
simin@ida.liu.se

Jan Torin
Chalmers University of Technology
torin@ce.chalmers.se

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 a control node. 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. In particular, fault management mechanisms not existing in the present centralized system but only in the distributed system are considered.

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 a 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 is excellent to ensure predictability in the time domain and enforce 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 the system made 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. Similar treatment of faults in presence of partial synchrony can also be implemented as a middleware service [6]. But this paper concentrates on application level fault tolerance that was desired in the given aerospace context.

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. Additionally, our semi-synchronous approach opens up for software diversity.

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.

In this paper we present an analysis of the consistency of the distributed control system in presence of faults and discrete mode changes. Using properties of the selected architecture, we present arguments that the system can reconfigure and keep its desired control properties in presence of faults. In particular, 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. These arguments form the sketch of a proof that can be formalized in future works.

Note that the architecture for realizing a distributed flight control system has been defined earlier and is not motivated in this paper. This paper takes the architecture as given and studies its inherent replica consistency in relation to a class of faults and discrete mode changes.

The analysis is built up in two stages. First, we consider the overall safety requirements of the aircraft.

These are then stated in terms of desired properties of a distributed flight controller, especially upon 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. The requirement on the DFCS is (informally) formulated in terms of safety and bounded response properties after this first stage of study.

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 these faults violates the requirements stated above.

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 sections. Next section presents the hardware architecture, and 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

The multi-role aircraft 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 well controllable and able to perform safe landing even when one primary control surface is streamlining.

Hardware replication (sensor and cockpit nodes, bus) is added to the system in order to meet the safety requirements with regards to 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.

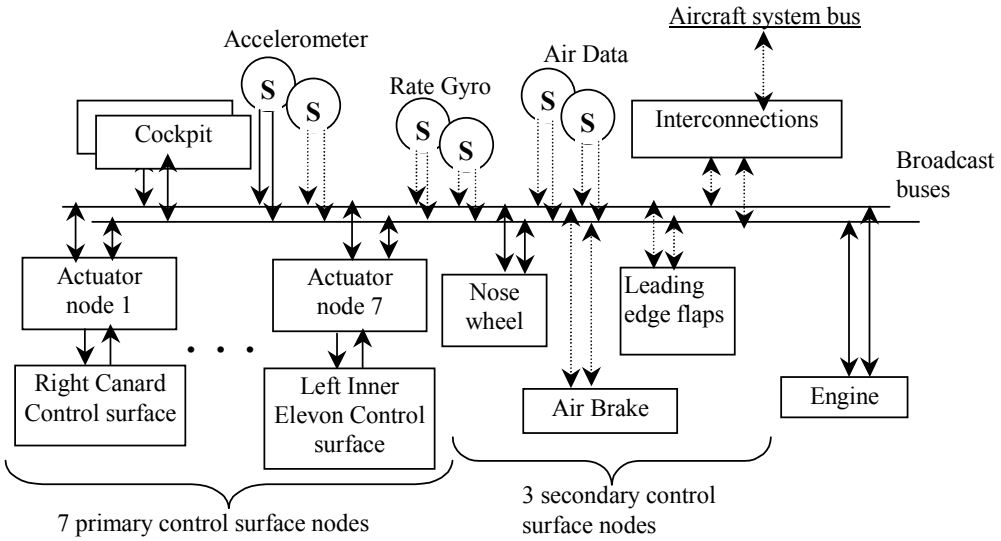

Figure 1 Sensor and Actuator Nodes of the Distributed FCS.

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 but inherently replica consistent. Below we will show how the consistency among distributed control nodes in presence of various faults can and will be maintained using inherent replica consistency.

3. System Structure and Fault Model

In the distributed FCS illustrated in Figure 1, all control and logic is allocated to the actuator nodes as well as fault handling mechanisms. 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 input 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. 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 will be described in next section.

Interface. The main purpose of the interface is to deal with the incoming and outgoing messages on the bus, in effect implementing the TDMA protocol. Messages of particular interest in a TDMA round are 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, to compensate a streamlining surface, might be selected in the CLC.

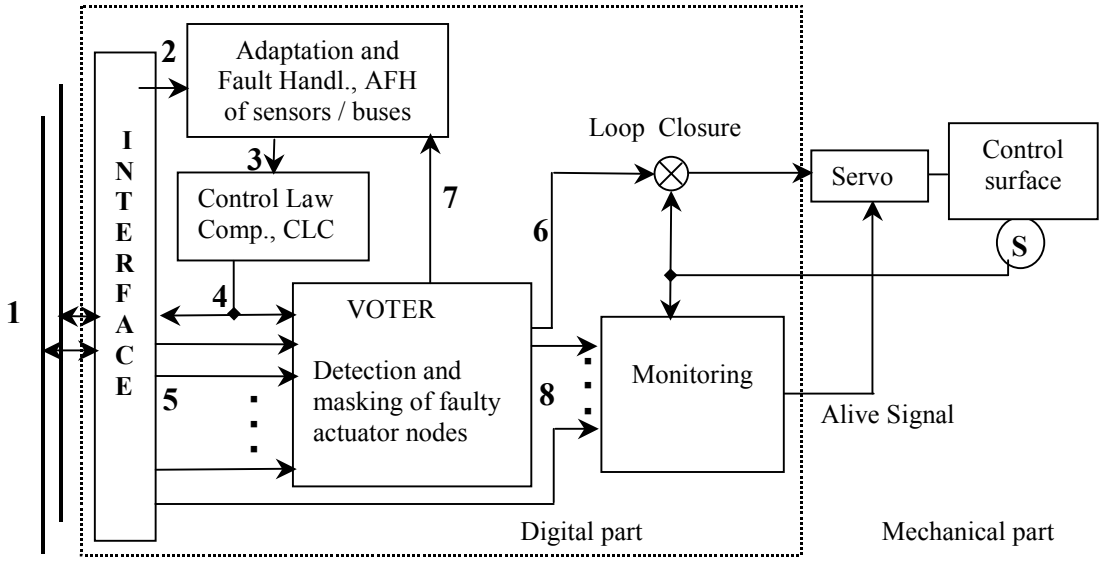

Figure 2 Simplified View of an Actuator.

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 equally important, as the Voter to achieve required FT, but is left out at this stage where focus is on consistency of the distributed Voters. (Monitor, Voter and Interface programs are checked by checksum calculation and coded variables for fault detection, due to the fact that components do not keep history between the cycles.)

3.2. Communication and data flow

Next we explain the data flow into, within, and out of each actuator node. Figure 3 pictures sensor and actuator messages sent each TDMA round (messages from nodes not relevant for the analysis are left out).

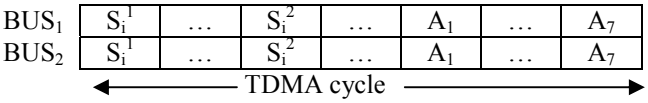

Figure 3 Messages broadcast every TDMA round under fault free condition.

In the beginning of each TDMA round, the duplicated sensors broadcast their messages, $[S_1^1, \ldots, S_n^1]$ and $[S_1^2, \ldots, S_n^2]$ on both buses. 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 = 1..Number_of_modes, and w = {η, ξ} to indicate normal or reconfigured CLC operation. Each actuator node receives all sensor values and computes AHF and CLC, and subsequently exchanges information by broadcasting the messages, A_1 to A_7 in 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: $\mathbf{A}_p$: $[\upsilon_1$- $\upsilon_7, mode_v^\eta, \alpha, \xi_p, \ldots]$.

In the paper we use ∅ to indicate a missing value, x^{faulty} for a faulty value of the variable in position *x*, and ¬ξ 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 $\mathbf{M}$:[…, m_k, …], k = 1..Number_of_nodes. *Arrow #2*, input to AFH block, illustrates the duplicated set of received sensor values $\mathbf{S}_{in}$: $[S_1^1, \ldots, S_n^1]$ and $[S_1^2, \ldots, S_n^2]$ 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 actuator voters, typically carrying normal mode control commands, $\mathbf{A}_p$: $[\upsilon_1$- $\upsilon_7, mode_v^\eta, \alpha, \neg\xi, \ldots]$. 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, $\mathbf{V}_{p\ out}$: $[\upsilon_p]$ 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

Only transient and permanent physical faults causing errors handled by the actuators are identified in this paper. Below, the considered fault classes and error descriptions for the DFCS components and functions are listed.

Input signals can be erroneous due to permanent or transient faults in sensor nodes (S_i^m, i=1..n, m=1,2) or buses. The errors appear in the actuator(s) as:

a) Wrong or missing value(s) in *all* actuators during *all* sequential TDMA rounds (e.g. permanent physical damage in sensor node or bus).
b) Wrong or missing value(s) in *all* actuators during *one* TDMA round (e.g. transient bit flip in sensor node or EMI disturbance on one, or both, buses).
c) Correct values in one subset of actuators and wrong values in the other actuators (different delays or disturbance on buses, referred to as Byzantine faults [5]).

Faults in the communication interface can manifest as:

d) Permanently corrupted or no message / values (e.g. permanent fault in the sequence handler).
e) Transiently corrupted message / value (e.g. bit flip in communication memory buffer, might cause inconsistency between affected and correct actuator nodes).

Faults in the processor / memory can manifest as (error in the information flow on arrow 4):

f) Permanent crash
g) Permanent (all TDMA rounds) omitted value or value error
h) Transient (one TDMA round) omitted value 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 section by stating the fault handling requirements on the DFCS as well as the assumptions used in the analysis, and continue with the main focus, description of the fault management mechanisms in the actuator nodes. The actuator analysis is more comprehensive since all sensor adaptation, control law computation, signal monitoring, etc are allocated to the actuators.

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: No combinations of two transient faults lead to streamlining.

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.

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. Moreover, the mode change should take place within a predefined number of cycles in each actuator, and the control decision taken in the very same TDMA cycle.

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 is required to take 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. For the distributed case we must additionally 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 must reconfigure synchronously.

In this section 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 subsection 4.3 and the assumptions in next subsection.

4.2. 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.

4.3. Actuator fault handling mechanisms

In this section we present detection and handling mechanisms for the faults described in §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 later in this section.

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) 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.

The table shows there are 5 rows in which the system will resort to streamlining (reconfiguration) of actuator surface. These fault-handling scenarios will be further considered in section 5.

Table 1 DFCS Actuator Fault handling

DETECTION MECHANISM	FAULT HANDLING MECHANISM	SYSTEM EFFECT
Sensor		
Wrong sensor value –Compare with replicated value, assertion checks and sensor model in AFH	Exclude faulty sensor value in CLC	If transient fault: None If permanent fault: Redundancy loss
Missing sensor value – detected by bus protocol	Use redundant sensor value	
Bus		
Destroyed messages detected by CRC	Use messages from redundant bus	If transient fault: None If permanent fault: Redundancy loss
Nothing or noise on one of the buses detected by bus protocol	Switch to duplicate bus	
Actuator node		
-Communication Interface		
Corrupted message(s) detected by CRC	Masked by voter	None
No messages – detected by bus protocol	Streamlining	Reconfiguration
-Processor		
Wrong result (faulty operating calculation units) detected by comparison in voter	Reuse previous states in CLC, masked by voter	None
No results (no messages) – detected by bus protocol	Streamlining	Reconfiguration
-CLC (memory)		
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)		
Wrong result (bit flip in program) detected by monitoring	Streamlining	Reconfiguration
-Monitoring (memory)		
Wrong result (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

This section puts forward the arguments that show inherent replica consistency is upheld in the DFCS in presence of discrete mode changes and certain combination of faults. The analysis covers: single transient faults (section 5.1), multiple transient faults (section 5.2), single permanent faults (section 5.3) and combination of single permanent and transient faults (section 5.4).

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 arbitrary 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 Transient Faults

We begin our analysis by considering single transient faults. The first column in Table 2 embraces all transient faults listed in §3.3. The second column shows which vector/vectors that are affected, and especially the voters input vectors, $\mathbf{A}_p$, p = 1..7.

From Table 2 follows that no single transient fault leads to inconsistency among the distributed control nodes, and from this first assessment we can recognize that transient faults affecting the voter or the monitoring functions will not violate the consistency requirement (Req.2). Moreover, a single transient fault will not lead to streamlining.

Table 2 Single transient faults

ERROR DESCRIPTION	ERROR MANIFESTATION	COMMENTS
Wrong or missing value from sensor S_i^1 See §3.3 b)	$\mathbf{S}_{in}$: […, S_{i-1}^1, ∅, S_{i+1}^1…,] ∧ […, S_{i-1}^2, S_i^2, S_{i+1}^2…,]	Inconsistency cannot happen since all actuator nodes receive the same sensor values.
Corrupted message, m_k, in one bus during *Comm. Phase 1* or *Comm. Phase 2* See §3.3 e)	**M**: […,m_{k-1}, m_k, m_{k+1}, …] followed by **M**: […,m_{k-1}, ∅, m_{k+1}, …]	Inconsistency cannot happen since all actuator nodes receive messages on the duplicated bus.
Transient fault during *Proc. Phase 1* in A_p See §3.3 h)	$\mathbf{A}_p$:[υ^{faulty}, …,$mode_v^{\eta}$, α, ¬ξ, …] q≠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 A_p See §3.3 h)	$\mathbf{A}_{p\ out}$: [υ_p^{faulty}] q≠p ⇒ $\mathbf{A}_{q\ out}$: [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 A_p, See §3.3 h)	As in the present FCS (Saab restricted information).	Have no impact on decisions in the voters.

In the next section we consider all possible combinations of single transient faults above except for those appearing in Voter and Monitoring.

5.2. Concurrent transient faults

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

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

Table 3 Combination of transient faults

CONCURRENT ERRORS	ERROR MANIFESTATION	COMMENTS
Double sensor faults: Wrong or missing sensor value in sensor, S_i^1, and in sensor, S_j^2	$\mathbf{S}_{in}$: […, 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 if affected signal was to enforce one.
Double bus faults: Corrupted message, m_k, on Bus 1 and, m_l, on Bus 2	$\mathbf{M^1}$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...] $\mathbf{M^2}$: […, 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: Faulty operating calculation units in A_p and in A_q	$\mathbf{A}_p$, $\mathbf{A_q}$: […, υ^{faulty}, …]	If p = q all voters vote on 6 correct vectors and one (p's) faulty. If p ≠ 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	$\mathbf{S}_{in}$: […, S_i^{faulty}, …] […, S_i^2, …] $\mathbf{M^1}$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...]	Handled by duplication in sensors and buses.
Concurrent sensor and actuator faults: S_i^1 and A_p send faulty values on both buses	$\mathbf{S}_{in}$: […, S_i^{faulty}, …] […, S_i^2, …] $\mathbf{A}_p$: […, υ^{faulty}, …]	Single sensor faults are handled by AFH. All voters vote on 6 correct vectors and one faulty.
Concurrent actuator and bus faults: A_p send faulty values and m_k is corrupted on bus one	$\mathbf{A}_p$: […, υ^{faulty}, …] $\mathbf{M^1}$: […, 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.

5.3. Single permanent faults

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 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. We now show that all seven voters shall take the decision on reconfiguration simultaneously and within a certain number of cycles. First we consider single permanent faults (Table 4).

Table 4 Single permanent faults: Streamlining and synchronous reconfiguration

In Cycle j a Permanent fault causes control surface p to streamline		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ¬ξ, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [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_{1-7} , 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.		

In next section we show that the above property (reconfiguration in the same cycle) will not be affected by a concurrent single transient fault.

5.4. 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 single permanent fault from Table 4 is combined with the transient faults from Table 2 (except for those appearing in Voter and Monitoring) and analyzed in Table 5. The combined cases are:

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

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

Case III: Actuator *p* issues streamlining and in the same TDMA cycle transient computation faults occur during *Processing Phase 1* in actuator *q*. This case includes two sub-cases, *III a)* and *III b)* in both of which the permanent fault occurs in actuator *p*. In *III a)* the transient fault that appears in actuator *q* affects continuous signals υ, and in *III b)* it affects the discrete signal ξ.

Table 5 Combined permanent & transient faults

Case I In Cycle j: Streamlining in A_p and Sensor fault in S_i^1		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{S}_{in}$: […, S_{i-1}^1, S_i^{faulty}, S_{i+1}^1, …] ∧ […, S_{i-1}^2, S_i^2, S_{i+1}^2, …] $\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ¬ξ, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\xi}$, α, ξ_p, …]
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 4.		
Case II In Cycle j: Streamlining in A_p and message, m_k, is corrupted on Bus 1		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{M}^1$: […, m_{k-1}, m_k^{faulty}, m_{k+1}, ...] $\mathbf{M}^2$: […,m_{k-1}, m_k, m_{k+1}, ...] $\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, ∅, ξ_p, …] ∀ $q \neq p$ $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\xi}$, α, ξ_p, …]
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 4.		
Case III a) In Cycle j: Streamlining in A_p and computation faults during *Processing Phase 1* in actuator *q* resulting in faulty command values.		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] $\mathbf{A}_q$: […, $\boldsymbol{\upsilon}^{faulty}$, $mode_v^{\eta}$, α, ¬ξ, …] ∀ $r \neq p, q$ $\mathbf{A}_r$: [υ_1- υ_7, $mode_v^{\eta}$, α, ¬ξ, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $i \neq p$ $\mathbf{A}_i$: […, $mode_v^{\eta}$, α, ξ_p, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $i \neq p$ $\mathbf{A}_i$: […, $mode_v^{\xi}$, α, ξ_p, …]
Cycle j to j+2: Faulty command words in A_q are masked by the voter and we will have the same result as in the single permanent case with synchronous reconfiguration in three TDMA rounds.		
Case III b) Streamlining in A_p, and transient faulty discrete signal in A_q e.g. *q* flags for streamlining, ξ_q.		
TDMA cycle j	TDMA cycle j+1	TDMA cycle j+2
$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] $\mathbf{A}_q$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_q, …] ∀ $r \neq p, q$ $\mathbf{A}_r$: [υ_1- υ_7, $mode_v^{\eta}$, α, ¬ξ, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\eta}$, ∅, ξ_p, …] ∀ $i \neq p$ $\mathbf{A}_i$: [υ_1- υ_7, $mode_v^{\eta}$, α, ξ_p, …]	$\mathbf{A}_p$: [υ_1- υ_7, $mode_v^{\xi}$, ∅, ξ_p, …] ∀ $i \neq p$ $\mathbf{A}_i$: [υ_1- υ_7, $mode_v^{\xi}$, α, ξ_p, …]
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, is present in A_q. Thus, also this case is similar to the single permanent one and reconfigure in 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, *III b)*. The above sections thus complete the proof sketch that replica consistency

eventually holds in the distributed nodes (in the j+3rd cycle) in presence of the interesting fault classes.

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 further studied the replica consistency property that was initiated in [1] and covered continuous signals. Here, the reasoning is extended to include discrete values. 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. What remains to complete the picture is the consideration of permanent faults in the communication system.

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 defining and detailing the design, especially the voter, at a formal level where the analysis can be checked by employment of formal verification tools. A recent study of a much simpler component, an avionic sensor voter, indicates that the derivation of environment models is the major step in verifying the component under development [9]. In that work several models of the environment were iteratively developed as a result of counter examples generated by a formal verification engine. In our case, the detailed analysis above is a major step towards characterizing the conditions (combinations of faults) under which the formal proofs would be meaningful and reliable.

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).

8. References

[1] K. Ahlström, J. Torin, K. Fersán, P. Nobrant, “Redundancy Management in Distributed Flight Control Systems; Experience & Simulations”, in proceedings of AIAA and IEEE 21th Digital Avionics Systems Conference, Irvine, CA, USA, 2002.

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

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

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

[5] H. Sivencrona, P. Johannessen, J. Torin, “Protocol Membership in Dependable Distributed Communication Systems – A Question of Brittleness”, SAE World Congress paper No. 2003-01-0108, Detroit, USA, 2003.

[6] D. Szentivanyi, S. Nadjm-Tehrani, Middleware Support for Fault Tolerance, Chapter in the book "Middleware for communications", Q. Mahmood (Ed.), John Wiley and sons, July 2004.

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

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

[9] S. Dajani-Brown, D. Cofer, A. Bouali, “Formal Verification of an Avionics Sensor Voter using SCADE”, In proceedings of International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, (FTRTFT´04), 2004.