Experimental Evaluation of the Fault Tolerance of an Atomic Multicast System

Jean Arlat
LAAS-CNRS, Toulouse
Martine Aguera
LAAS-CNRS, Toulouse
Yves Crouzet
LAAS-CNRS, Toulouse
Jean-Charles Fabre
INRIA/LAAS-CNRS, Toulouse
Eliane Martins
LAAS-CNRS, Toulouse
David Powell
LAAS-CNRS, Toulouse

Key Words — **Atomic multicast, Coverage, Experimental evaluation, Fault injection, Fault-tolerant computing, Distributed system, Protocol testing**

Reader Aids —
Purpose: Presentation of a fault injection-based experimental validation study
Special math needed for explanations: Probability theory
Special math needed to use results: None
Results useful to: Fault-tolerant system designers and reliability analysts

Abstract — **This paper presents a study that contributes to the validation of a dependable local area network providing multipoint communication services based on an atomic multicast protocol. This protocol is implemented in specialized communication servers that exhibit the *fail-silent* property, ie, a kind of halt-on-failure behavior enforced by self-checking hardware. The tests that have been carried out utilize physical fault injection and have two objectives: i) to estimate the coverage of the self-checking mechanisms of the communication servers and ii) to test the properties that characterize the service provided by the atomic multicast protocol in the presence of faults. An appreciable part of the paper is devoted to the description of the testbed that has been developed to carry out the fault injection experiments. The major results are presented and analyzed.**

INTRODUCTION

This paper deals with the application of fault injection for the experimental validation of an open dependable distributed system architecture developed in the framework of the Delta-4 project [1].

For a system to merit the epithet "dependable", it is necessary that its users may justifiably place their confidence in the system's ability to deliver the specified service [2]. Thus one of the major problems related to the development of dependable computer systems concerns validation of the fault tolerance mechanisms (FTM's) intended to procure system dependability: error detection, recovery, etc.. The goal of validation is then to obtain confidence in the ability of the FTM's to comply with their assigned role.

Besides other possible approaches such as proving or analytic modeling whose applicability and accuracy are appreciably restricted in the case of complex fault-tolerant systems, *fault-injection* is particularly attractive [3-8]. By speeding up the occurrence of errors and failures, fault injection is a method for *testing* the FTM's with respect to their own specific inputs: *the faults*.

Two important contributions of fault injection concern the verification of the FTM's and the characterization of their behavior, thus enabling any weakness in their design and/or implementation to be revealed. Also, a statistical analysis of the responses obtained during the fault injection experiments enables some relevant dependability parameters—coverage, fault dormancy, error latency, etc.—to be estimated. Although many experimental results based on fault injection have been recently published (eg, see [9-14]), the statistics presented are often specific and cannot be easily compared. Accordingly, the comparison of the results obtained in this study with other similar studies is beyond the scope of this paper. The interested reader can refer to the references.

Different forms of fault injection experiments (eg, fault simulation [15], fault emulation [16], error seeding [17], mutation testing [18], physical fault injection [19]) can be considered depending on i) the complexity of the system to be validated (the **target system**), ii) the types of faults injected and iii) the level of its application at various stages of the development process [20]. The fault injection method used here is **physical fault injection**: the faults are directly injected on the pins of the IC's that implement the prototype of the target system. Although this methodology can only be applied at the final stages of the development process, its main advantages are that the tested prototype is close to the final system and that it enables a global validation of a complex system integrating both hardware and software.

From a general point of view, the experimental validation is based on the concept of a fault injection **test sequence**. More precisely, a fault injection test sequence is characterized by an **input** domain and an **output** domain. The **input** domain corresponds to a set of injected **faults F** and a set **A** that specifies the data used for the **activation** of the target system and thus, of the injected faults. The **output** domain corresponds to a set of **readouts R** that are collected to characterize the target system behavior in presence of faults and a set of **measures M** that are derived from the analysis and processing of the **FAR** sets. Together, the **FARM** sets constitute the major attributes that can be used to fully characterize a fault injection test sequence. In practice, the fault injection test sequence is made up of a series

0018-9529/90/1000-0455$01.00©1990 IEEE

of **experiments**; each experiment specifies a point of the {**FxAxR**} space.

The main feature of the Delta-4 architecture, from the dependability viewpoint, is the tolerance of *accidental hardware (physical) faults*. The basic principle of fault-tolerance is the replication of software components on distinct host computers interconnected by a broadcast local area network (LAN). The Delta-4 multipoint communication system (MCS) provides generalized multipoint services that enable dependable communication between the replicated software components. The essential basis for these generalized multipoint services is an atomic multicast protocol (AMp) implemented over the medium access control (MAC) sub-layer for several alternative standard LAN's. To date, AMp has been implemented for ISO 8802.4 token bus and ISO 8802.5 token ring; development is under way for ISO 9314 (FDDI). The results presented here concern the ISO 8802.5 token ring implementation.

MCS is implemented in communication servers specific to the Delta-4 architecture, called **network attachment controllers** (NAC's). The NAC's exhibit the *fail-silent* property (a kind of halt-on-failure behavior) enforced through self-checking hardware. With the goal of validating the dependability properties of the Delta-4 architecture, an automated distributed testbed was built around the fault injection tool MESSALINE developed at LAAS and previously used for the experimental validation of a computerized interlocking system [21].

The fault injection experiments reported in this paper contribute to both aspects of validation [2]: **fault removal** and **fault forecasting**.

The first aspect addresses the *removal of design and/or implementation faults* in the FTM's. The implemented testbed enables the service delivered by the AMp to be tested; the required service is defined by means of a set of properties that characterize what is meant by an atomic multicast. Since the system under test is meant to be able to tolerate hardware faults that occur during its operational life, the testing of AMp should also be carried out in the presence of such hardware faults. From the viewpoint of fault removal, fault injection experiments can be seen as complementary to other techniques such as informal or formal verification.

Fault forecasting requires the prior estimation of the rates of occurrence of component failures (due to residual design and implementation faults or to operational faults) and of the effectiveness of any fault-tolerance mechanisms that have been built into the system. Such estimations provide input parameters to a global dependability model allowing prediction of the system dependability in terms of reliability, availability, safety, etc. The implemented testbed contributes to such dependability evaluation by providing the means to estimate the effectiveness of the system's FTM's.

The remainder of the paper has four sections. Section I gives a short presentation of the Delta-4 architecture, emphasizing the fault tolerance issues. The hardware and software characteristics of the testbed developed to carry out the fault injection experiments are described in section II. Section III details the parameters of the **FARM** attributes that fully define the fault injection test sequence. Finally, the results obtained are presented and analyzed in section IV.

I. FAULT-TOLERANCE IN DELTA-4

I.1. *Fault Hypotheses*

The Delta-4 architecture is primarily intended to cope with accidental physical faults [22]. However, the tolerance of i) accidental design faults of the software and ii) intentional faults (such as intrusions) are also investigated [1].

Taking into account physical faults only, two classes of computers can be distinguished according to their assumed failure modes:

- *fail-silent* host computers, for which it can be assumed that a faulty computer cannot send any message,
- *fail-uncontrolled* host computers for which the possible failure modes include: omission to transmit (some) messages, sending of messages with erroneous contents, sending of extra messages, message refusal, etc..

In order to handle both types of computers in the Delta-4 architecture, specific network attachment controllers (NAC's) have been designed to exhibit a *fail-silent* behavior; this is enforced through self-checking hardware. In the token ring implementation, upon detection of an error in the NAC, a relay is commanded to provoke its extraction from the ring.

The principle of the Delta-4 architecture is illustrated by figure 1 that shows the components of the architecture as well as the considered fault hypotheses.

Each NAC is thus intended to implement a barrier to error propagation so as to prevent host failures from altering the operation of other hosts. The evaluation of the actual coverage provided by the NAC extraction mechanisms is of prime importance since the design and operation of the AMp are based on this *fail-silent* behavior hypothesis.

I.2. *The Atomic Multicast Protocol (AMp)*

The Delta-4 multipoint communication system provides a set of multipoint protocols allowing both *visible* multicasting between groups of logically distinct application software components and *invisible* multicasting to the various replicates of software components that have been installed on distinct hosts for the purposes of fault tolerance. This multipoint communication system also provides functions for processing message transmission and refusal errors of replicated software components. The foundation of these multipoint communication services is the atomic multicast protocol (AMp) implemented as an extension of the MAC sub-layer, providing the user with standard MAC and *atomic multicast* service primitives.

Table 1 presents the plain language definition of the AMp service properties that have been tested in the fault injection test sequence; more details on AMp and on the definition of further properties of AMp can be found in [23] and [1], respectively.

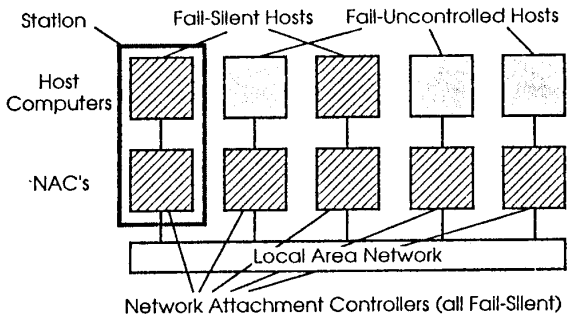

Figure 1. Principle of Delta-4 Architecture

TABLE 1
The AMp Properties

PA. *Unanimity*: Any message delivered to a correct recipient is delivered to all other correct recipients.

PB. *Non-triviality*: This property is divided into two sub-properties:
PB.1: Every message positively confirmed to a correct sender, is delivered to all correct recipients.
PB.2: Every message delivered to a correct recipient is positively confirmed to its sender, if the sender is correct.

PC. *Order*: If a pair of messages is delivered to a pair of correct recipients, then both messages are delivered in the same order to both recipients.

In these definitions, a *recipient* designates an AMp service user of a predefined group. The *universe of recipients* is a logical group of entities to which the message is addressed. Conversely, the *sender* is the service user which requests messages to be multicasted to the group. The sender belongs to the universe of recipients. A *correct recipient* (*sender*) is a recipient (sender) residing on a fault-free station.

II. THE FAULT INJECTION TESTBED

II.1. *Guidelines*

The validation of the dependability properties of the Delta-4 architecture has two objectives:

- to estimate the coverages provided by the FTM's: i) the *local coverage* achieved by the self-checking mechanisms commanding the extraction of the NAC's and ii) the *distributed coverage* corresponding to the fault tolerance provided by both AMp and the NAC self-checking mechanisms,
- to test, in the presence of faults, the *service* provided by AMp.

With respect to service testing, the following points were considered.

i. As it is not possible to test the entity implementing the AMp in isolation, a multi-layer testing [24] is performed: the tested entity incorporates the AMp (MAC + extensions) sub-layer and the physical layer.
ii. The only point of control and observation of the tested entity is at its upper boundary and only the AMp service primitives are tested.
iii. In order to test the multicast service offered by AMp, the testbed must support the interconnection of several replicates of the tested entity.
iv. The testbed must be able not only to control and observe the service provided by the tested entity, but also to synchronize the activities of the latter with the injection of faults into the hardware on which the protocol is executed.
v. The testbed should be flexible in order to support the testing of other layer(s) of the Delta-4 communication architecture, requiring as few changes as possible in its structure.

Furthermore, as the experimental validation takes place at the last stage of the development process, the testbed should be able to adapt easily to several variants of the NAC architecture—in particular with respect to improved self-checking mechanisms—and to successive versions of the AMp software.

The quality of the results depends on the application of as many faults (or combinations of faults) as possible. Thus, many experiments must be carried out. Due to the duration of the execution of a set of fault-injection experiments for a single IC, and in order to prevent inconvenience to other users of the prototype system, the tests were run overnight. Therefore, the testbed had to be able to operate without human intervention to tolerate failures of the stations induced by the injected faults.

II.2. *The Hardware Architecture*

Figure 2 shows the hardware architecture of the testbed. It is mainly composed of the **Target System** and of the fault injection tool **MESSALINE** that are interconnected by specific hardware lines and by means of the **Testbed Network** and the **Experiment Supervisor**.

The **Target System** contains four stations interconnected by the *Target System Network* (the token ring LAN). Each station is made up of a NAC containing the implementations of the AMp and physical entities that are under test together with a *Target System Host* (*TSH*) which activates and observes the service offered by AMp. The activation consists in the generation of traffic flow through the target system network and the observation consists in the collection of all data obtained from each station.

The purpose of the **Experiment Supervisor (ES)** is twofold: i) run-time control of the target system and ii) collection and analysis of the observed data. The **Testbed Network**, which is an Ethernet type LAN in the present implementation, ensures the communication between the ES and the TSH's. The ES is implemented on a Bull SPS9 machine.

Most hardware and software necessary for the fault injection test sequence are part of **MESSALINE**. For sake of conciseness, only the two principal components are briefly presented here; a more detailed description of MESSALINE hardware architecture is given in [20].

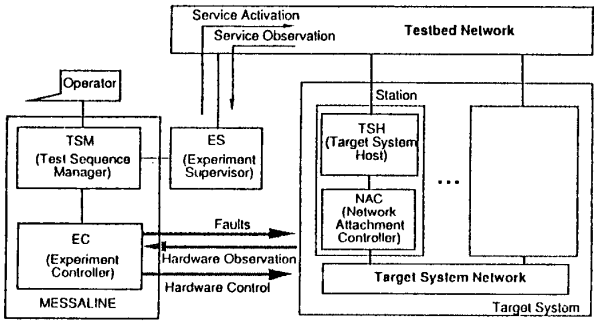

Figure 2. The Testbed Architecture

The *Experiment Controller* (*EC*) actually implements the injection of the elements of the **F** set and the collection of the hardware signals used to elaborate some elements of the **R** set. Faults are injected into the NAC component of a specific station (designated hereafter as the *injected station*) by means of a multi-pin probe, which enables faults to be directly applied to IC pins. A connection with the wiring concentrator of the target token ring network allows the states of the insertion relays of the stations connected on the ring to be read; these are used for the post-test analysis, as will be explained later. The only intervention of the operator is to position the probe on the circuit selected by the *Test Sequence Manager* (*TSM*).

The TSM is implemented on a Macintosh II computer, connected to the EC and to the ES through serial lines. There are also physical connections between MESSALINE and all the stations (TSH's and NAC's) in order to enable automatic hardware resets: the NAC's are reset after each experiment whereas a host reset is requested only in case of a station crash.

In the reported experiments, the Target System is made up of four Bull SPS7 machines, running Unix System V, as TSH's. The preliminary NAC architecture tested here contains only limited self-checking based on internal bus parity checks and a watch-dog timer. Improved self-checking NAC's featuring duplicated processors and memory have subsequently been designed and are now being validated on the same testbed; this NAC is currently implemented to interface a Ferranti Argus 2000 TSH. Both NAC architectures are made up of two boards:

- a **main board** that ensures the interfacing with the host computer,
- a **specific board** that connects the main board to the physical medium.

II.3. *The Software Architecture*

The TSM implemented in MESSALINE defines the **FARM** attributes of the test sequence. It is based on a highly parametrizable software module that has been described in [21]. The specification of the **FARM** parameters used for this test sequence will be presented in the next section. The remainder of this section focuses essentially on the software architecture specifically developed for the Experiment Supervisor and the Target System Hosts.

The software structure of the components concerned with the test of the atomic multicasting service is shown on figure 3.

The ES software is composed of four main modules.

i. *Test execution control*: This module performs the main supervisory function, ie, the run-time control of the target system and the analysis of the results obtained from AMp service monitoring.
ii. *ES-TSH communication*: This module allows the ES to request each station to open/close multicast connections, to start/drop traffic generation and to transfer the data collected by each station at the end of an experiment; this communication takes place through the Testbed Network.
iii. *Protocol analysis*: Based on data collected by the stations during an experiment, this module can determine whether the properties presented in table 1 are satisfied or not.
iv. *TSM-ES communication*: This module supports a command-reply style of communication between the TSM and the ES; TSM sends a command to the ES, which executes this command and sends back a reply, indicating the success or failure of the action executed and (when needed) the results obtained. In particular, there are commands to specify the activation mode of the target system, to activate/stop the activity of the target system, and to collect the results of the protocol analysis.

Besides the data files containing the results from protocol service observation, each TSH also sends information about locally observed events, which is added to the data generated by the ES itself and stored in a *history file* (*H*). This file contains, in text form, the results obtained for each experiment as well as a detailed trace of the execution of the test sequence and the codes of the incidental errors affecting either the ES or the target system. This information is used at present to assist the operator in the post-mortem analysis of failure situations.

The TSH software is composed of three modules:

i. *Observed traffic generator*: This generates the traffic flow used for protocol analysis;
ii. *Background traffic generator*: This generates a traffic flow used to create a more representative activity on the network under test;
iii. *Dispatcher*: This module provides access to the AMp service primitives, which include: insertion/removal of a station, open/close of a multicast communication, open/close of the connections for the observed and background traffic flows and data transfer.

The messages sent and received by the observed traffic flow are stored for later analysis. The observed traffic generator stores messages that were sent and confirmed positively (*P file*) or negatively (*N file*). The dispatcher stores the observed received messages in the *R file*. At the end of an experiment, these files are passed to the ES together with other information such as the total number of messages sent and received from both traffic flows and the mean confirmation delay (the time the transmitter spends sending a message, ie, the time before the transmitter can issue the next request).

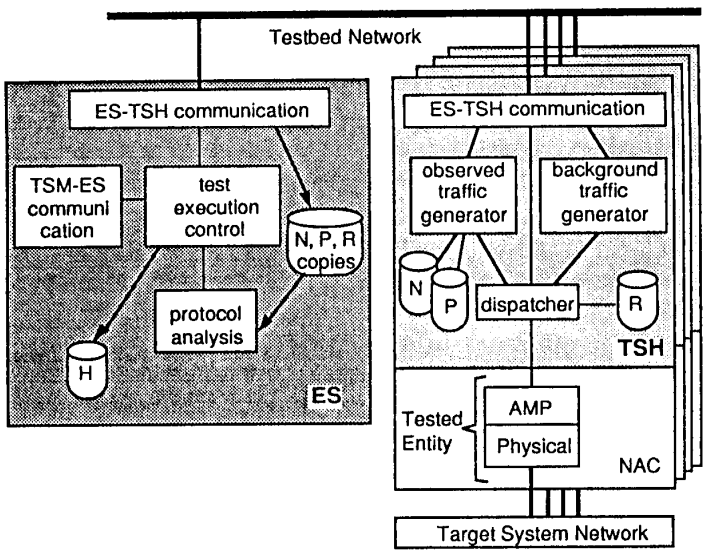

Figure 3. ES and TSH Software Structure (Only one station is detailed)

III. DEFINITION OF THE TEST SEQUENCE

This section presents successively the main parameters that specify the Fault, Activation, Readout and Measure (**FARM**) attributes of the test sequence.

III.1. *The F Set*

Faults are injected into a single NAC; the four stations were thus partitioned into two sets: the **injected station** (station S1) and the three **"correct"** (non-injected) **stations** (stations S2, S3, S4).

In order to tackle the most frequent fault types, temporary[1] faults were injected on multiple pins of a single IC at a time. The temporary nature of the injected faults enabled the **forcing** technique[2] to be used with a minimal risk of damaging the injected IC's.

In order to cope with the large number of parameters characterizing the **F** set, random sampling was used for the determination of the faults to be injected. For each IC, the injected pins, the nature and the timing characteristics of the injected faults were selected according to the order and form given below:

i. **Multiplicity (MX)**: Faults are injected on several (1, 2, or 3) pins of an IC, with a frequency of 50%, 30%, and 20% respectively.
ii. **Location**: Selection of MX pins among the injectable pins, with a uniform distribution.
iii. **Nature**: Stuck-at-0 and stuck-at-1 faults, each with equal probability of occurrence.
iv. **Timing characteristics**: The faults injected are essentially intermittent faults; their temporal parameters are composed of three values:

- **lead time** (from start of experiment to first pulse): uniformly distributed between 1s and 40s,
- **period**: logarithmically distributed between 10μs and 30ms,
- **width**: uniformly distributed between 2μs and 1ms, with a duty cycle $\leq$ 50%.

Because of a lack of sound available data concerning actually observed IC failure modes, most of the parameters were selected according to a uniform distribution among range values selected quite arbitrarily. However, the limitation to a multiplicity order of 3 is to some extent substantiated by the results in [8] for a microprocessor in the presence of single event upsets. These results show that more than 80% of the internal single-event upsets led to error patterns on up to 3 pins. The logarithmic distribution for the period is intended to obtain an appreciable number of short period intermittent faults while maintaining a wide selection range.

III.2. *The A Set*

The workload was varied to study its impact on the behavior of the target system. The activation is characterized by the application of two types of traffic flows: **observed traffic** flow with respect to which AMp properties are tested[3] and **background traffic** flow to provide more realistic activation of the target system.

In order to activate the various operational modes of the NAC's, the host of station S1 (the injected station) has to take successively the following activation modes:

[1]As an example, proportions ranging from 85% to 90% of temporary faults are cited in [15].

[2]In the case of the **forcing** technique, voltage levels are directly applied by means of multi-pin probes on IC pins and associated equipotential lines.

[3]The observed traffic consisted of 100 messages.

1: Transmits and receives both observed and background traffic messages.
2: Receives only messages of the observed traffic, while background traffic messages are both received and transmitted.
3: Transmits and receives only background traffic messages.
4: Is inserted into the token ring network, but is not concerned with traffic flow.
5: Is inactive, ie, is powered up but not inserted into the ring.

Stations S2, S3, S4 (*correct stations*) can have any configuration, but the following constraints are imposed in order to perform a meaningful test:

- all correct stations must receive both observed and background messages,
- at least two stations must transmit observed traffic messages,
- at least two stations must transmit background traffic messages.

There are a number of suitable configurations for a system with one injected station and three correct stations[4]; however, a single configuration per mode was chosen. Table 2 shows the Transmitter/Receiver allocations of the stations for each mode with respect to observed traffic and background traffic flows.

III.3. *The R Set*

Three types of readouts are collected for each experiment:

- **binary** readouts: activation of the injected faults, status of the ring insertion relays for each NAC, AMp properties derived from the analysis of the messages,
- **timing** readouts: time of activation of the injected fault, time of extraction of the stations,
- **message** readouts: number of messages exchanged for both traffic flows and number of messages positively or negatively confirmed for the observed traffic.

The information concerning the activation or not of the injected faults can be obtained from specific monitoring devices implemented in MESSALINE that sense current variations on the pin(s) where the fault is injected [20]. The status of these devices can be read by software. Such information was used to eliminate from the statistics those experiments where the fault was not activated and also to perform a direct measurement of fault dormancy.

The collection of occurrence and timing readouts enabled empirical distributions to be derived for the fault dormancy and for the coverage achieved by the hardware error detection mechanisms of the NAC's and by the AMp.

On the average, each fault injection experiment took about five minutes. This large value is mainly due to the experiment set-up and to the possible execution of the automatic recovery/restart procedures in case of failure of the testbed after a fault has been injected. More specifically, the watch-dog monitoring the useful duration of each experiment was set to 110 seconds.

TABLE 2
Transmitter/Receiver Allocation per Activation Mode

	Observed Traffic				Background Traffic			
Mode	S1	S2	S3	S4	S1	S2	S3	S4
1	TR	TR	R	R	TR	TR	TR	TR
2	R	TR	TR	R	TR	TR	TR	TR
3	—	TR	TR	R	TR	TR	TR	TR
4	—	TR	TR	R	—	TR	TR	TR
5	X	TR	TR	R	X	TR	TR	TR

T: Transmitter present, R: Receiver present, —: No Traffic, X: Not inserted; Station 1 (S1) is the injected station.

III.4. *The M Set*

The measures considered for the analysis presented here consist of two types of measures: **predicates** and **time distributions**.

Let E designate the **error occurrence** predicate; ie, E is true if the injected fault is activated on the faulted pin(s). Let Ii denote the status of the ring insertion relay of station Si, $i=1,\ldots,4$; Ii is true if Si is inserted into the ring)

The **local coverage** or **error detection predicate** D characterizes the efficiency of the NAC self-checking mechanisms:

$$D = E \cdot \bar{I}i$$

D is true if the NAC of the injected station is extracted and if the injected fault is activated (the anticipated behavior in presence of faults)[5]. The notation: $A \cdot B$ is used to designate the conjunction between predicates A and B.

Let P designate the **predicate** associated to the conjunction of **all the AMp properties** and let **predicate** C characterize the **confinement** of the fault/error (ie, all the "correct" stations remain inserted in the ring. The **distributed coverage** or **fault tolerance predicate** T, that characterizes the defensive properties of the protocol at the MAC layer, can be expressed as:

$$T = E \cdot P \cdot C = E \cdot (PA \cdot PB \cdot PC) \cdot (I2 \cdot I3 \cdot I4)$$

T is true whenever **all the AMp properties** are verified, the **confinement** of the fault/error is ensured and E is true. Although, it might *a priori* mask interesting characteristics, the grouping of the P and C predicates into one single predicate is substantiated by the two following remarks:

[4]Indeed, the number of possible configurations is 13 in mode 1, 8 in modes 2 and 3, 5 in modes 4 and 5.

[5]An opposite use of D has been made in the case of mode 5 to test if the station remained extracted when faults were injected; thus, in this case, a 100% coverage is assumed when the station remains extracted.

- the results obtained to date [25, 26] have never led to an observation of P being false when C was true,
- the status of predicate P is of little interest when predicate C is false.

Thus, in subsequent analyses, the status of predicate T can be strongly related to the status of predicate C.

Two types of time distributions complement the analysis. The **fault dormancy** measures the time interval between the injection of a **fault** and its activation as an **error** at the point(s) of injection; if T_d denotes this random variable, and T_E and T_F the error and fault times respectively, then:

$$T_d = T_E - T_F$$

The **extraction latency** corresponds to the time interval between the injection of a **fault** and the **extraction** of the NAC of the injected station; if T_l denotes this random variable, and T_D the extraction time, then:

$$T_l = T_D - T_F$$

IV. PRINCIPAL RESULTS AND ANALYSES

Due to the complexity of the target system resulting from its distributed architecture, it was not feasible to synchronize the activity of the target system and the instants of fault injection (which would have been necessary to carry out repeatable experiments) and thus the analyses carried out rely more on statistical evaluation than on deterministic verification. However, traces and memory dumps recorded for each major failure event occurring during an experiment provided useful data for the fault removal task.

The statistics reported hereafter gather the results obtained for fault injection experiments involving the two boards that implement the NAC components: i) a **main** board that ensures the interfacing with the host computer, ii) a **specific** board that connects the main board to the physical medium. The full details on the experiments carried out on these boards appear in [26] and [25], respectively.

IV.1. *Data Set Description*

For each IC subjected to fault injection, **30** experiments were carried out for each of the 5 activation modes; ie, the complete test of one IC consisted of a run of **150** experiments that was fully automated to enable the experiments to be carried out overnight. Table 3 summarizes the characteristics of the test sequence for the two boards.

TABLE 3
Characteristics of the Test Sequence

	Number of IC's	Number of IC's Tested	Useful Statistics	Number of Experiments	Pin Coverage
Specific Board	42	20	20	3000	80%
Main Board	59	20	17	1799	87%
Total	101	40	37	4799	84%

The first row indicates that complete statistics were obtained for the **20** tested IC's in the case of the **specific board**. For the **main board,** complete statistics could not be obtained for all the IC's initially selected for the test sequence. Indeed, some problems were encountered in conducting the experiments. In many experiments, the injected faults provoked severe crashes of the injected station that could not be easily recovered by the automatic reset procedures implemented in the testbed: a complete reload of the injected station system disk was necessary! This problem had two important impacts:

- it extended the duration of the test sequence,
- it restricted the amount of useful statistics available.

Even though only a subset of IC's were tested, the use of the forcing fault injection technique means that a high proportion of actual equipotentials on the boards are faulted. Translated into the proportion of pins connected to faulted equipotentials, the resulting **pin coverage** was about **80%** for the specific board, **87%** for the main board and more than **84%** for the complete NAC.

Due to the functional difference between the two boards, the analyses presented in the remainder of the paper emphasize the comparison between the results obtained for each board.

IV.2. *Predicate Analysis*

IV.2.1. Impact of the Activation Modes

Table 4 compares the ratings obtained for the E, D and T predicates and shows the impact of the activation modes. Due to the specificity of mode 5, for which it has been observed that in all cases the injected station remained non inserted and the predicate T was verified, only the influence of modes 1 to 4 is presented here.

The columns labeled "Exp." give the number of experiments used in the statistics to compute the E predicate. The statistics for the D and T predicates are based on the number of activated faults, ie, "Exp. $\times$ E." In each cell of the table, the **boldface** symbols give the nominal percentage obtained while the values given above and below, in *italics*, designate respectively the upper and lower bounds of the corresponding 95% statistical confidence interval [27].

The results concerning predicate E show that the activity on the main board is higher for all activation modes. For predicates D and T, the values obtained for mode 4 differ from those obtained for modes 1-3. The lack of activity of the injected station in mode 4 explains the important and opposing[6] deviations observed on the D and T predicates. Accordingly, in subsequent analyses of the D and T predicates, the influences of modes 1-3 will be merged.

[6]The opposing effects on D and T can be explained by the fact that since the activity is lower in mode 4, the injected station detects fewer errors (decrease in D), but also propagates fewer errors (leading to an increase in T).

TABLE 4
Influence of the Activation Modes on the Predicates

	Specific Board				Main Board				Global			
Mode	Exp.	*E*	*D*	*T*	Exp.	*E*	*D*	*T*	Exp.	*E*	*D*	*T*
		93.49%	*81.50%*	*94.04%*		*96.79%*	*90.41%*	*86.45%*		*94,45%*	*84,52%*	*90,33%*
1	**600**	**91.83%**	**78.77%**	**92.38%**	**409**	**95.35%**	**87.95%**	**83.59%**	**1009**	**93,26%**	**82,57%**	**88,74%**
		89.79%	*75.75%*	*90.30%*		*93.32%*	*84.96%*	*80.26%*		*91,84%*	*80,44%*	*86,92%*
		92.90%	*84.11%*	*89.54%*		*97.03%*	*91.95%*	*81.37%*		*94,07%*	*86,51%*	*85,66%*
2	**600**	**91.17%**	**81.54%**	**87.39%**	**360**	**95.56%**	**89.53%**	**77.91%**	**960**	**92,81%**	**84,62%**	**83,73%**
		89.06%	*78.64%*	*84.86%*		*93.40%*	*86.49%*	*74.01%*		*91,31%*	*82,52%*	*81,58%*
		94.97%	*80.99%*	*91.59%*		*96.81%*	*86.96%*	*81.04%*		*95,29%*	*82,51%*	*86,92%*
3	**600**	**93.50%**	**78.25%**	**89.66%**	**360**	**95.28%**	**83.97%**	**77.55%**	**960**	**94,17%**	**80,42%**	**85,07%**
		91.64%	*75.25%*	*87.35%*		*93.07%*	*80.43%*	*73.63%*		*92,79%*	*78,15%*	*83,01%*
		91.69%	*68.60%*	*96.95%*		*96.38%*	*39.91%*	*97.90%*		*92,97%*	*56,91%*	*97,01%*
4	**600**	**89.83%**	**65.31%**	**95.73%**	**340**	**94.71%**	**35.40%**	**96.58%**	**940**	**91,60%**	**54,12%**	**96,05%**
		87.61%	*61.85%*	*94.05%*		*92.32%*	*31.15%*	*94.48%*		*89,98%*	*51,31%*	*94,80%*

IV.2.2. Predicate Combinations Analysis

Table 5 characterizes the predicate combinations used for the analysis.

Figure 4 synthesizes the results for modes 1-3. The histograms reveal an important variation in the distribution of the combinations between the main board and the specific board, in particular with respect to the $\bar{D}\cdot T$ and $D\cdot\bar{T}$ combinations. This is a direct consequence of the slight increase of the D predicate and of the reduction of the T predicate identified on table 4.

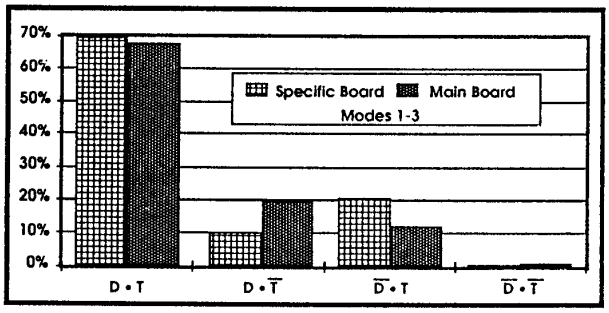

Figure 4. Predicate Combination Coverages

TABLE 5
Characterization of the Predicate Combinations

D·T:	This corresponds to the ideal case when the error results in: • the permanence of the connection of the correct stations and the verification of the properties of the protocol, • the extraction of the injected station.
D·T̄:	This corresponds to a failure of AMp that may result from: • either, an implementation deficiency in the protocol to handle the extraction of a station, • or, an excessive extraction latency of the injected station resulting in alteration of the AMp properties.
D̄·T:	In this case, the injected fault does not alter the observed behavior of the system (the protocol properties are verified and all the stations, including the injected one, remain connected); this case may correspond to: • an activated fault that creates an error at the point of injection but which is not propagated, • a propagated error which remains latent in the NAC, • an error propagated outside the NAC, but tolerated by the defensive mechanisms of the protocol entities.
D̄·T̄:	This case characterizes a major failure of both the extraction mechanisms and the AMp properties.

These results show that a large proportion (about 22% for the main board and 12% for the specific board) of the **local coverage** ($D = D\cdot T + D\cdot\bar{T}$) still results in the failure of the AMp properties (case $D\cdot\bar{T}$); further analysis of this combination will be presented later. On the other hand, a large proportion (15% for the main board to 25% for the specific board) of the apparent **distributed coverage** ($T = D\cdot T + \bar{D}\cdot T$) is due to the $\bar{D}\cdot T$ combination.

IV.3. *Timing Analysis*

IV.3.1. Dormancy

Figure 5 synthesizes the fault dormancy distributions obtained for modes 1-3.

Although the time-out was set to 110 seconds, all readouts fell below 100 seconds, accordingly the time scale was decomposed into five logarithmic time intervals. Both distributions show that about 95% of the faults are activated in less than 0.1 seconds and that if a fault is not activated within 1 second there is a high probability that it will never be activated. The main board histogram identifies a reduction in the dispersion of the dormancy that corresponds to a speed-up of the activation of the injected faults that supports the difference between the boards already noted with respect to predicate E (refer to table 4).

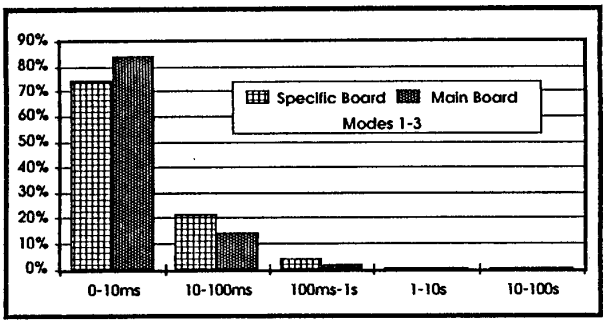

Figure 5. Fault Dormancy Distributions

IV.3.2. Latency

Figure 6 synthesizes the extraction latency distributions obtained for modes 1-3. The specific board histogram depicts a bi-modal distribution that identifies (at least) two distinct behaviors with respect to the injected faults; such an observation is of special interest with respect to modeling when selecting the number and types of coverage parameters. In the case of faults injected on the main board, there is an important reduction in the dispersion of the latency distribution (the bi-modal shape is no longer present) that leads to an overall increase in the mean extraction time. The difference noted between the main board and the specific board can be explained by the fact that most of the hardware detection mechanisms are currently integrated in the specific board.

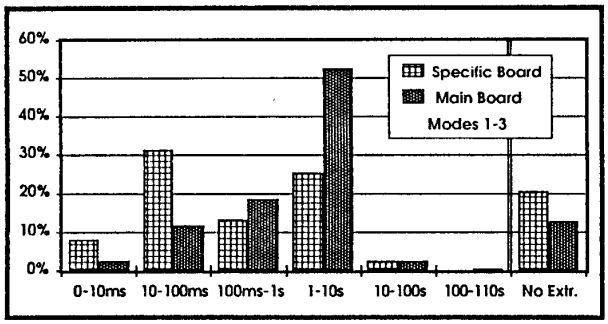

Figure 6. Extraction Latency Distributions

IV.4. *Complementary Tests*

The results presented in this subsection provide further insight into the behavior of the target system in the presence of faults. Two major topics are investigated, namely:

- the possible causes of the $D \cdot \overline{T}$ predicate combination (table 5),
- the impact of the timing characteristics of the injected faults.

IV.4.1. Influence of Station Extraction on the Distributed Coverage

This analysis is intended to justify and discriminate between the two possible causes—AMp Design/Implementation Problem vs Large Extraction Latency—for the occurrence of the $D \cdot \overline{T}$ predicate combination.

IV.4.1.1. Specific Extraction Experiments

Specific experiments were carried out to characterize AMp behavior upon extraction of one station. In these experiments, the predicate D is always true (each experiment is in reality a forced station extraction obtained by commanding the relay that controls the extraction of the NAC). Table 6 compares the results obtained with those of the (random) fault injection experiments.

TABLE 6
Comparison of Specific and Fault Injection Experiments

Experiments	Extrac. (D)	D·T̄ (Occ.)	D·T̄ (Freq.)
Specific	1600	21	1,31%
Random Fault Injection			
•Specific Board	1319	165	12.5%
•Main Board	939	210	22.4%
•Complete NAC	2258	375	16.6%

It appears that for the 1600 specific experiments carried out, one or more of the "correct" (non-injected) stations were extracted in 21 cases (1.31%). This figure clearly identifies a potential design/implementation problem. However, the 10% to 20% difference with respect to the fault injection experiments shows that this is not the only cause for the observation of the $D \cdot \overline{T}$ predicate combination. Globally, this design/implementation problem corresponds to 7.89% (1.31%/16.6%) of the $D \cdot \overline{T}$ combination.

IV.4.1.2. Correlation between the $D \cdot \overline{T}$ Combination and the Extraction Latency

Figure 7 gives, for modes 1-3, the relative distributions between the two combinations $D \cdot T$ and $D \cdot \overline{T}$, as a function of the extraction latency. The observed distributions clearly suggest that a positive correlation exists between the proportion of non confinement ($D \cdot \overline{T}$ combination) and the extraction latency, in particular for the specific board.

Such a potential correlation can be further investigated in a more formal way by considering simply the fact that if 2 events A and B are positively (resp. negatively) correlated then[7]:

$$\frac{\Pr\{A \cdot B\}}{\Pr\{A\} \times \Pr\{B\}} > 1 \left(\text{resp. } \frac{\Pr\{A \cdot B\}}{\Pr\{A\} \times \Pr\{B\}} < 1 \right)$$

[7]An analogous argument was recently used in a different experimental context [28] and was further refined in [29].

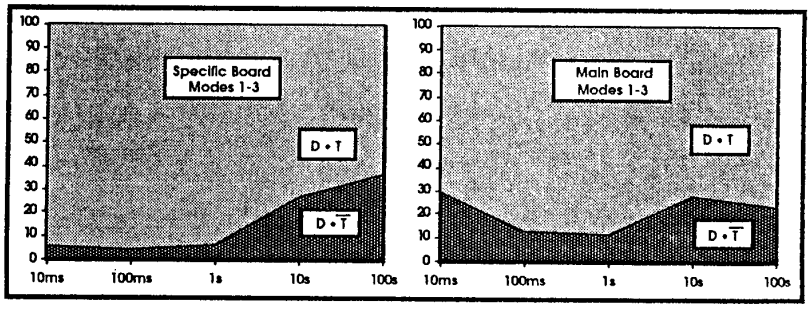

Figure 7. Relative Distribution between $D \cdot T$ and $D \cdot \bar{T}$ Predicate Combinations according to Extraction Latency

TABLE 7
Results of the Tests of the Correlation

		Specific Board	Main Board
Test #1:	$\frac{Pr\{\bar{T} \cdot D \cdot (T_1 > 1s)\}}{Pr\{\bar{T}\} \times Pr\{D \cdot T_1 > 1s)\}}$	$\frac{7.35\%}{10.19\% \times 27.25\%} = 2.6$	$\frac{15.32\%}{20.15\% \times 54.78\%} = 1.4$
Test #2:	$\frac{Pr\{\bar{T} \cdot D \cdot (T_1 \leq 1s)\}}{Pr\{\bar{T}\} \times Pr\{D \cdot (T_1 \leq 1s)\}}$	$\frac{2.59\%}{10.19\% \times 52.26\%} = 0.5$	$\frac{6.53\%}{20.15\% \times 32.40\%} = 0.6$

The equality to 1 characterizes statistical independence.

Consider $\bar{T}$ as event A and $\{D \cdot (T_1 > 1s)\}$ as event B (test #1). The investigation is complemented by a second test (test #2) considering the events $\bar{T}$ and $\{D \cdot (T_1 \leq 1s)\}$. The results obtained for both boards are shown on table 7. The results clearly support the hypothesis of a positive correlation, but also indicate that the correlation is lower in the case of the main board.

IV.4.2. Impact of the Timing Characteristics of the Injected Faults

The results presented up until here refer to the injection of intermittent faults characterized by the parameters described in sub-section III.1. In order to investigate the influence of the timing characteristics, further experiments were conducted based on the injection of **permanent** and **transient faults**.

Three IC's previously faulted were selected on the specific board to carry out these experiments[8]. In order to force the injection of permanent and transient faults, the range of the width and period parameters were modified. Permanent faults can be obtained by extending the width parameter to infinity, while transient faults correspond to the case where the period is made infinite. In order to provide consistent comparisons, all other parameters of the injected faults were kept to the same values. The detailed analysis is described in [26].

These experiments confirmed that temporary faults—especially transient faults—are much less prone to create errors. Though all the injected permanent faults were activated, only about 90% of the intermittent faults and less than 70% of the transient faults were activated. Figure 8 presents the impact observed on the predicate combinations.

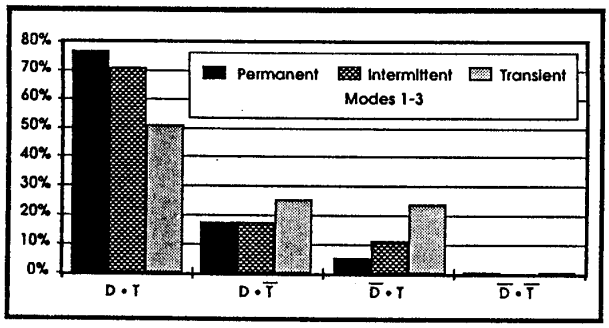

Figure 8. Impact on Predicate Combinations (Specific Board)

The results show that temporary faults—especially transient faults—are more difficult to handle than permanent faults; in particular, an important decrease (more than 25%) of the complete coverage (extraction and confinement) of the errors is observed for the $D \cdot T$ combination.

[8]The IC's were selected on the basis of their poor performance with respect to the global coverage that was observed in the previous experiments.

This is somewhat compensated for by the fact that errors induced by transient faults tend to provoke less damage to the other stations; a large increase (about 20%) in the proportion of the errors that do not provoke the extraction of the injected station do not affect the other stations (case $\bar{D}\cdot T$). It is interesting to note that this combination corresponds to the case where **nothing** is observed as a result of the activation of the injected fault. As already identified in table 5, three possibilities can lead to such a behavior. The fact that temporary faults increase the proportion of cases where nothing is observed provides an *a priori* support of the first or second alternatives (the error is not propagated outside of the NAC).

Finally, the impact on the lack of distributed coverage ($\bar{T} = D\cdot\bar{T} + \bar{D}\cdot\bar{T}$) appears to be of about 10%. The impact applies with respect to both permanent and intermittent faults, since the injected intermittent faults tend to behave more as permanent faults than transient faults. The observed increase suggests that among the faults which are detected locally (D true), transient faults disturb the other stations (T false) more than intermittent or permanent faults.[9]

SUMMARY & CONCLUSIONS

This paper has presented the current status of the experimental validation of some of the major dependability features of the Delta-4 architecture, namely the self-checking mechanisms of the Network Attachment Controllers (NAC's) intended to support the **fail-silent** property and the **fault tolerance** of the AMp extension of the 8802.5 network.

To perform these experiments, a flexible distributed testbed has been developed that enables the automatic control of the fault injection experiments. The main features of this testbed concern: i) the testing of the services provided by AMp by means of the **observation of the interactions between several communication entities** that implement this protocol, ii) the estimation of the effectiveness of the fault tolerance features of the target system in presence of **injected physical faults**, and iii) the automated execution of a test sequence without operator intervention, that was made possible by the **integration of fault tolerance features in the testbed** itself. As an example, at the end of each experiment, should the injected station be found to have crashed, it is rebooted automatically.

Although most of the results are specific to the system tested and cannot always be generalized to other systems, they nevertheless exemplify the types of analysis that can be carried out using **physical fault injection** and in particular when a tool such as **MESSALINE** is available.

As a summary of the results obtained for the fault injection test sequence carried out on the limited self-checking NAC, table 8 synthesizes the statistics for the main and specific boards.

The D and T symbols correspond to the predicates characterizing respectively the **local coverage** (error detection and extraction of the NAC) and the **distributed coverage** (the fault is tolerated; the AMp properties are verified and the error is confined, ie, no "correct" station is also extracted). $\bar{D}$ and $\bar{T}$ designate the complementary events. The third column discriminates the $D\cdot\bar{T}$ combination cases between design/implementation versus latency problems. Each number in the ratio column is obtained as the product of the percentages of the considered row.

The minimum fraction of the faults of the NAC covered by the tested system (NAC + AMp) is approximately 68%. The apparent fault coverage corresponds to the distributed coverage and is equal to 85.5% (68% + 17.5%). The actual coverage is function of the distribution case $\bar{D}\cdot T$ (17.5%) among the three possibilities defined on table 5. To obtain such a distribution, an in-depth analysis of the results of the test sequence is necessary; furthermore, supplementary observations of AMp might be needed.

The specific experiments consisting of direct action on the relay controlling the extraction of the NAC, showed that approximately 92% of the 17% of extractions observed in the fault injection test sequence for which the error was not confined,

[9]Based on these results, it would seem appropriate to modify the parameters of the intermittent faults in order to increase the period and reduce the duty cycle; however, it is worth noting that – as exhibited by transient faults – such a modification could also tend to significantly decrease the number of activated faults and thus the number of useful experiments.

TABLE 8
Synthesis of the Statistics for the Tested NAC (modes 1-3)

D vs. $\bar{D}$	T vs. $\bar{T}$	Des.-Impl. vs Latency	Ratio	Diagnosis
(D) 82%	(T) 83%	–	68.0%	Locally-detected and tolerated by AMP
	($\bar{T}$) 17%	8%	1.2%	Locally-detected but not tolerated due to AMP (design/implementation error)
		92%	12.8%	Locally-detected but not tolerated due to excessive latency (limited self-check of tested NAC)
($\bar{D}$) 18%	(T) 97%	–	17.5%	Not locally-detected but of no consequence to other stations (tolerated or masked locally)
	($\bar{T}$) 3%	–	0.5%	Not detected and not tolerated

can be attributed to excessive latency of the error detection of the NAC self-checking mechanisms. The other 8% are due to design or implementation errors of AMp: the "clean" extraction of the NAC is not tolerated by the other stations.

Although only a subset of AMp properties have been tested so far—the testing of the other properties given in [1] would have required a global clock and a global ordering of the events among the stations—, the fault injection test sequence has had an important impact on the project. The traces collected during the test sequence have been analyzed by AMp designers and they have succeeded in isolating several errors. Two successive new versions of the AMp software have been developed and tested on the same testbed with the same NAC hardware. Based on the test of 8 of the IC's previously tested, it was found that several error codes were eliminated and that the distributed coverage was greatly improved. Specifically, for the three successive versions, the $D \cdot \overline{T}$ predicate combination was reduced from 21% to 16% and finally to less than 11%.

Furthermore, the fault injection test sequence has evidenced the limited performance of the self-checking mechanisms implemented on the tested NAC and justified (especially for the main board) the need for the improved self-checking mechanisms implemented in an enhanced NAC architecture employing duplicated circuitry. The next step will thus be to test this new version of NAC hardware so as to evaluate the (hopefully) improved local and distributed coverages and make a motivated decision as to whether or not the increased cost of enhanced self-checking is justified. It is also worth noting that recent developments of the testbed now enable AMp properties related to the reconfiguration to be included in the analysis.

ACKNOWLEDGMENT

This work has been partially supported by the Commission of European Community (ESPRIT P2252 Delta-4) and by the *Conseil Régional de Midi-Pyrénées*. The work of the Delta-4 project is carried out by a consortium of 13 partners: Bull SA (F), Crédit Agricole (F), Ferranti Computer Systems Ltd. (UK), IEI-CNR (I), IITB-Fraunhofer (D), INESC (P), LAAS-CNRS (F), LGI-IMAG (F), MARI Applied Technologies Ltd. (UK), NCSR (UK), Renault (F), Sema Group (F), and the University of Newcastle upon Tyne (UK).

We are pleased to thank Jean-Claude Laprie for his encouragement and constructive comments that have greatly helped in the clarification of the ideas set forth in this paper. Suggestions provided by the anonymous referees were also very helpful in improving the quality of the paper.

The encouragement and assistance of our partners in the Delta-4 project are also gratefully acknowledged with special thanks to Marc Chérèque and René Ribot from Bull and Brian Gilmore from Ferranti.

REFERENCES

[1] *Delta-4: Overall system Specification*, The Delta-4 Project Consortium, (ed. D. Powell), ISBN 2-907801-00-7, *Delta-4 Document n°S88.040/12/P*, 1988 December.

[2] J.-C. Laprie, "Dependable computing and fault-tolerance: Concepts and terminology", *Proc. 15th IEEE Int'l Symp. Fault-Tolerant Computing*, Ann Arbor, Michigan, USA, 1985 Jun, pp 2–11.

[3] Y. Crouzet, B. Decouty, "Measurements of fault detection mechanisms efficiency: Results", *Proc. 12th Int'l Symp. Fault-Tolerant Computing*, Santa Monica, California, USA, 1982 Jun, pp 373–376.

[4] J. H. Lala, "Fault detection, isolation and reconfiguration in FTMP: Methods and experimental results", *Proc. AIAA/IEEE Digitals Avionics Systems Conf.*, 1983 Nov, pp 21.3.1-21.3.9.

[5] Segall, Vrsalovic, Siewiorek, Yaskin, Kownacki, Barton, Rancey, Robinson, Lin, "FIAT-Fault injection based automated testing environment", *Proc. 18th Int'l Symp. Fault-Tolerant Computing*, Tokyo, Japan, 1988 Jun, pp 102–107.

[6] A. Damm, "Experimental evaluation of error-detection and self-checking coverage of components of a distributed real-time system", Doctorate thesis, Tech. Univ. Vienna, 1988 Oct.

[7] R. Chillarege, N. S. Bowen, "Understanding large system failures—A fault injection experiment", *Proc. 19th IEEE Int'l Symp. Fault-Tolerant Computing*, Chicago, Illinois, 1989 Jun, pp 356–363.

[8] U. Gunneflo, J. Karlsson, J. Torin, "Evaluation of error detection schemes using fault injection by heavy-ion radiation", *Proc. 19th IEEE Int'l Symp. Fault-Tolerant Computing*, Chicago, Illinois, USA, 1989 Jun, pp 340–347.

[9] M. E. Schmid, R. L. Trapp, A. E. Davidoff, G. Masson, "Upset exposure by means of abstraction verification", *Proc. 12th Int'l Symp. Fault-Tolerant Computing*, Santa Monica, California, USA, 1982 Jun, pp 237–244.

[10] M. A. Schuette, J. P. Shen, D. P. Siewiorek, Y. X. Zhu, "Experimental evaluation of two concurrent error detection schemes", *Proc. 16th Int'l Symp. Fault-Tolerant Computing*, Vienna, Austria, 1986 Jul, pp 138–143.

[11] K. G. Shin, Y. H. Lee, "Measurement and application of fault latency", *IEEE Trans. Computers*, vol C-35, 1986 Apr, pp 370–375.

[12] R. Chillarege, R. K. Iyer, "Measurement-based analysis of error latency", *IEEE. Trans. Computers*, vol C-36, 1987 May, pp 529–537.

[13] G. B. Finelli, "Characterization of fault recovery through fault injection on FTMP", *IEEE Trans. Reliability*, vol R-36, 1987 Jun, pp 164–170.

[14] Segall, Barton, Vrsalovic, Siewiorek, Rancey, Robinson, "Fault injection based automatic testing: Practice and examples", *Proc. 8th Digital Avionics Systems Conf.*, San Jose, California, 1988 Oct.

[15] G. S. Choi, R. K. Iyer, V. Carreno, "A fault behavior model for an avionic microprocessor: A case study", *Proc. 1st Int'l Working Conf. Dependable Computing for Critical Applications*, Santa Barbara, California, USA, 1989 Aug, pp 71–77.

[16] J.-P. Gérardin, "Design aid to reliable and safe systems: The DEFI", *Electronique Industrielle*, n°116, 1986 Nov, pp 58–63, (in French).

[17] A. Mahmood, D. M. Andrews, E. J. McCluskey, "Executable assertions and flight software", *Proc. 6th AIAA/IEEE Digital Avionics Systems Conf.*, Baltimore, Maryland, USA, 1984 Dec, pp 346–351.

[18] R. A. DeMillo, R. J. Lipton, F. G. Sayward, "Hints on test data selection: Help for the practicing programmer", *Computer*, 1978 Apr, pp 34–41.

[19] M. L. Côrtes, S. D. Millman, H. A. Goosen, E. J. MacCluskey, "Techniques for injecting non stuck-at faults", *CRC Tech. Report n°87-21*, 1987 Mar.

[20] Arlat, Aguera, Amat, Crouzet, Fabre, Laprie, Martins, Powell, "Fault-injection for dependability validation—A methodology and some applications", *IEEE Trans. Software Engineering*, vol 16, 1990 Feb, pp 166–182.

[21] J. Arlat, Y. Crouzet, J.-C. Laprie, "Fault-injection for dependability validation of fault-tolerant computing systems", *Proc. 19th IEEE Int'l Symp. Fault-Tolerant Computing*, 1989 Jun, pp 348–355.

[22] D. Powell, P. Verissimo, G. Bonn, F. Waeselynck, D. Seaton, "The Delta-4 approach to dependability in open distributed computing systems", *Proc. 18th Int'l Symp. Fault-Tolerant Computing*, Tokyo, Japan, 1988 Jun, pp 246–251.

[23] P. Verissimo, L. Rodrigues, M. Batista, "AMp: A highly parallel atomic multicast protocol", *Proc. ACM SIGCOMM 89 Symp.*, Austin, Texas, USA, 1989 Sept, *Computer Communication Review*, vol 19, 1989 Sept, pp 83–93.

[24] ISO "Open systems interconnection – Conformance testing methodology and framework, Part 1: General concepts", DIS 9646-1, 1988 Jul.
[25] M. Aguera, J. Arlat, Y. Crouzet J.-C. Fabre, E. Martins, D. Powell, "Results of fault-injection into an MCS network attachment controller with limited self-checking", *LAAS Research Report 89-071, Delta-4 Document n°R89.022/12/P*, 1989 May, 68 pp.
[26] J. Arlat, M. Aguera, Y. Crouzet, J.-C. Fabre, E. Martins, D. Powell, "Dependability testing Report LA1", *LAAS Research Report 89-410, Delta-4 Document n°R89.139/11/P*, 1989 Dec, 35 pp.
[27] V. D. Agrawal, "Sampling techniques for determining fault coverage in LSI circuits", *J. Digital Systems*, vol 5, n°3, 1981, pp 189–202.
[28] P. G. Bishop, F. D. Pullen, "Error masking: A source of failure dependency in multi-version programs", *Proc. 1st Int'l Working Conf. Dependable Computing for Critical Applications*, Santa Barbara, California, USA, 1989 Aug, pp 25–32.
[29] R. K. Iyer, L. T. Young, P. V. Iyer, "Automatic recognition of intermittent failures: An experimental study of field data", *IEEE Trans. Computers*, vol 39, 1990 Apr, pp 525–537.

AUTHORS

Jean Arlat; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Jean Arlat (M'80) was born in Toulouse in 1953. He received the Certified Engineer degree from the National Institute of Applied Sciences of Toulouse (INSAT) in 1976 and the Doctor-Engineer degree from the National Polytechnic Institute of Toulouse (INPT) in 1979. He is *Chargé de Recherche* with the National Center of Scientific Research (CNRS) and member of the group "Dependable Computing and Fault Tolerance" of the Laboratory for Automatics and Systems Analysis (LAAS) of the CNRS that he first joined in 1976. During 1979-80 he spent a postdoctoral year in the Computer Science Department of the University of California at Los Angeles, working on the dependability and performance evaluation of software fault tolerance architectures. Since 1980 he is a research staff member at LAAS-CNRS. His research interests focus on the evaluation of hardware-and-software fault-tolerant systems including both analytical modeling and experimental fault injection. Dr. Arlat is a member of the IEEE Computer Society, of the Fault-Tolerant and Simulation Technical Committees, and the AFCET Working Group "Dependability of Computing Systems".

Martine Aguera; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Martine Aguera was born in Carcassonne, France, in 1957. She received the "Diplôme d'Enseignement Supérieur de Technologie" (DEST) in Automatic Control from the "Centre National des Arts et Metiers (CNAM)", Toulouse in 1987. She gained experience about microprocessor architectures in the company CEIS Espace (France) with the realization of the French ground station of the SARSAT System (Search And Rescue Aided by SATellite) in 1980-1983. She is now working in Information Processing and Instrumentation support service at LAAS-CNRS. In this context, she is responsible for the implementation of the Delta-4 Network Testbed at LAAS.

Yves Crouzet; LAAS-CNRS; 7, Aveune du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Yves Crouzet was born in Toulouse, France, in 1952. He received the Engineer degree from the Higher National School of Electronics, Electrical Engineering, Computer Science and Hydraulics, Toulouse, in 1975 and obtained his Engineering Doctorate from the National Polytechnic Institute, Toulouse, in 1978. He is "Chargé de Recherche" at the National Center for Scientific Research (CNRS). Since 1975 he has been a member of the Dependable Computing and Fault-Tolerance group at LAAS-CNRS. During 1975-1982 he worked on the design and realization of self-checking VLSI circuits. Since 1982 his research interests have concerned the experimental validation of dependable systems by fault-injection.

Jean-Charles Fabre; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Jean-Charles Fabre was born in Toulouse, France, in 1957 and received his Masters and Diploma for Further Studies in Computer Science in 1980 from the Univesity of Toulouse, and his Doctorate in 1982. From 1981 to 1983 he was at INRIA working in the field of Distributed Computing Systems Architectures, especially with the Chorus project. He became a permanent researcher (Chargé de Recherche) of INRIA in January 1984 and joined the Dependable Computing and Fault-Tolerance group at LAAS-CNRS. He is currently involved in the LAAS-INRIA Saturne project and the European Esprit Delta-4 project. His interest concerns distributed algorithms, fault and intrusion-tolerance in Distributed Systems, implementation validation by fault-injection. He was consultant with the Chorus Systèmes Company on the Colombus project of the European Space Agency and has served as an expert for a project of the European Esprit program.

Elaine Martins; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Elaine Martins was born in Rio de Janeiro, Brazil, in 1955. She received her BS in Computer Science from the Rio de Janeiro Federal University (UFRJ) in 1976, and her MS in the same area, from the same University, in 1982. She is now a PhD student at the "Ecole Nationale Supérieure de l'Aéronautique et de l'Espace (ENSAE)", in Toulouse. Her research interests include methodologies and tools for protocol and distributed systems testing and the experimental validation of the hardware and software fault-tolerant systems. The thesis is being developed on the group "Dependable Computing and Fault Tolerance" of the LAAS-CNRS, and concerns the experiment validation of an open distributed dependable architecture using the fault-injection approach. She worked in the design and implementation of software for mainframe, mini- and micro-computers, which included a real-time executive for a communication server based on loosely-coupled INTEL 8085 microprocessors for the Brazilian Research Space Institute.

David Powell; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

David Powell was born in Greenwich, England, in 1951 and received his Bachelor of Science degree in Electronic Engineering from the University of Southampton, England in 1972. He has been at LAAS-CNRS since 1972 and is a member of the Dependable Computing and Fault-Tolerance group. He obtained his Speciality and State Doctorates in 1975 and 1981 respectively. His current research work in the Dependable Computing and Fault-Tolerance group at LAAS concerns fault-tolerance and security in distributed computing systems. He has written over 30 papers for international and national journals and conferences and holds a patent for a fault and damage-tolerant network for data transmission. He has managed several national and European research constracts and carried out consultancy work with several aerospace, telecommunication and data processing companies in France. He is currently "Directeur de Recherche" at the National Center for Scientific Research (CNRS) and is Scientific Director of the European Esprit Delta-4 project.

Manuscript TR90-305 received 1990 January 16; revised 1990 May 21.

IEEE Log Number 37709 ◄TR►