# Online Diagnosis and Recovery: On the Choice and Impact of Tuning Parameters

Marco Serafini, *Student Member*, *IEEE*, Andrea Bondavalli, *Member*, *IEEE*, and Neeraj Suri, *Senior Member*, *IEEE*

**Abstract**—A sequenced process of *Fault Detection* followed by the erroneous node's *Isolation and system Reconfiguration* (node exclusion or recovery), that is, the FDIR process, characterizes the sustained operations of a fault-tolerant system. For distributed systems utilizing message passing, a number of diagnostic (and associated FDIR) approaches, including our prior algorithms, exist in literature and practice. Invariably, the focus is on proving the completeness and correctness (all and only the faulty nodes are isolated) for the chosen fault model, without explicitly segregating permanent from transient faulty nodes. To capture diagnostic issues related to the persistence of errors (transient, intermittent, and permanent), we advocate the integration of count-and-threshold mechanisms into the FDIR framework. Targeting pragmatic system issues, we develop an adaptive online FDIR framework that handles a continuum of fault models and diagnostic protocols and comprehensively characterizes the role of various probabilistic parameters that, due to the count-and-threshold approach, influence the correctness and completeness of diagnosis and system reliability such as the fault detection frequency. The FDIR framework has been implemented on two prototypes for automotive and aerospace applications. The tuning of the protocol parameters at design time allows a significant improvement with respect to prior design choices.

**Index Terms**—Error detection, transient faults, online diagnosis, system reliability, recovery.

## 1 Introduction

A fault-tolerant system is designed to provide sustained delivery of services despite encountered perturbations. The ability to accurately detect, diagnose, and recover from faults[1] in an online manner (that is, during system operation) constitutes an important aspect of fault tolerance. This Fault Detection followed by Isolation and system Reconfiguration (FDIR) process has two primary objectives: to *consistently identify* a faulty node so as to restrict its effect on system operations and to support the process of system *recovery* via isolation and reconfiguration of the system resources to sustain ongoing system operations. If FDIR is performed as an online procedure [32], [33], then this provides an effective capability of resource management, responding promptly to the appearance and disappearance of faults, with a small duration of system susceptibility to subsequent fault accumulation.

However, the capacity of consistently identifying faulty nodes does not necessarily imply the ability to select the best recovery action. For example, an overpessimistic FDIR can overreact and exclude all nodes encountering transient faults, thus reducing available resources and impacting reliability. A possible solution is the use of count-and-threshold approaches [6], which established a fundamental basis for online recording and handling of transients. It enables accumulating "Fault Detection" information over system operations before triggering the most appropriate "Isolation and Reconfiguration" actions. The health of a node is thus determined, based on the persistency and recurrence of its failures, by postponing its isolation, even if some errors are observed.

In this paper, we introduce a generic FDIR framework for integrating existing distributed diagnosis approaches with a count-and-threshold algorithm. As the relative occurrences and ratios of permanent, intermittent, and transient hardware faults are matters of ongoing debate, especially as technology changes continually affect these rates [9], we develop a modeling methodology to probabilistically study the effects of such rate variations and to guide the choice of design parameters accordingly.

Our focus is on distributed systems, but the analysis and derived metrics are general enough to be adapted for the tuning of any periodic error detection subsystem, similar to [6], [7]. Despite appearing intuitive, most of the obtained results have not, to our knowledge, been comprehensively developed, linking both the diagnostic protocol and the count-and-threshold aspects.

The process of local detection, global diagnosis, isolation, and recovery from a given fault instance is a multifaceted problem. The specific aspects addressed in this paper are listed as follows:

- The capability of the FDIR processes to accurately capture the "severity" of the error. For example, errors of core-system-level functions are more severe than those of optional application-level functions

1. In reality, one detects the manifestation of a fault, that is, an "error." Thus, error detection is the accurate term to use [19]. Nevertheless, we utilize the more conventionally accepted term of "fault detection" for the FDIR terminology.

- *M. Serafini and N. Suri are with the Department of Computer Science, TU Darmstadt, Hochschulstr. 35, 64289 Darmstadt, Germany. E-mail: {marco, suri}@informatik.tu-darmstadt.de.*
- *A. Bondavalli is with the Dipartimento di Sistemi ed Informatica, Università degli Studi di Firenze, Viale Morgagni, 65, 50134 Firenze, Italy. E-mail: bondavalli@unifi.it.*

*Manuscript received 22 May 2006; revised 2 Apr. 2007; accepted 5 June 2007; published online 19 June 2007.*
*For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0062-0506.*
*Digital Object Identifier no. 10.1109/TDSC.2007.70210.*

and, consequently, will result in faster isolation and reconfiguration. It should be emphasized that the process of fault detection itself may not necessarily provide information on the severity of an error, unless specific error detection mechanisms exist, which correspond to established severity types.

- The capability to capture the "duration" of an error (the time period when it is continuously observed) and its "recurrence" (the frequency of successive observations). The desired response of the FDIR operations to a transient, regardless of its severity level, can be very different from the response to a permanent fault. For example, should we isolate a node encountering only transients?
- The impact of different settings of the parameters of the count-and-threshold algorithm on the resilience and the reliability of the system. For example, if error detection is executed too frequently, then the same fault will be detected multiple times, increasing the likelihood of isolating nodes, regardless of the transient nature of the fault. This can unnecessarily degrade system redundancy.

These issues motivate our research on FDIR processes. We highlight the trade-offs in the tuning of the design parameters, discuss the trends, and propose methods to aid tuning of the diagnostic process as tailored to specific system characteristics and requirements. We show how the approach is applicable to prototype systems for automotive and aerospace applications. The probability of isolation due to transient faults could almost be ruled out in both scenarios by considering that all functions, including safety-critical ones, show a certain degree of tolerance to transient outages. However, nodes with dormant faults activating as seldom as every 10 hours on the average can be isolated by appropriately tuning our FDIR algorithm.

### 1.1 Related Work

A variety of approaches exist, which address the FDIR process (or parts of it), and a complete survey is beyond the scope of this paper. We limit ourselves to a brief overview of the main existing work in the field.

The theoretical problem of diagnosis was set up in the Preparata-Metze-Chien (PMC) model [26]. The focus of this work and of many related approaches was on characterizing system configurations, fault sets, and assignments, where $n$ active components (units) are able to diagnose, in the presence of up to $t$ faulty units, all the faulty units (one-step $t$-diagnosability) or at least one of them (sequential $t$-diagnosability). The problem of assignment has been further developed from many viewpoints, trying to define sufficient and necessary conditions when only some combinations of the elements are known. Many extensions exist to the PMC assumptions, considering the fact that a fault might not always manifest in a permanent manner [5], [23] or extending the analysis from multiprocessor systems to distributed systems [17], [30]. An excellent survey on the strong similarities between diagnosis and consensus problems in distributed systems can be found in [2].

An important element for the timeliness of online diagnosis, especially in real-time systems requiring timely reaction to faults, is the ability to execute diagnostic tests without interrupting system operation, that is, without explicit testing capabilities. A well-known solution is the comparison approach [4], [24], [28], where multiple nodes execute the same task, and the outcomes are compared by other nodes.

If nodes are assumed to be fail silent, then group membership protocols can be used for FDIR operations. They ensure that all nodes have a consistent view of the current set of correct nodes. The first definition of the group membership problem and a solution in asynchronous systems were developed within the ISIS project [3]. One of the first approaches to group membership for synchronous systems was proposed in [10]. The time-triggered protocol (TTP) [16] intertwines a membership protocol with clock synchronization in synchronous systems.

Our previous work [33] introduced a family of distributed diagnostic algorithms for synchronous systems based on the Customizable Fault/Error Model (CFEM) [32], where the fault assumptions can be adapted to meet the fault hypothesis of the core fault-tolerant protocols of the system (for example, clock synchronization). One advantage is that diagnosis is not considered as an offline and fault-free procedure but as an online core fault-tolerant mechanism fully integrated in the system fault-tolerant strategy. Instead of executing dedicated performance-impacting tests like in the PMC model or constraining the allocation of application-level tasks to nodes like in the comparison approach, it uses error detection information derived by the execution of fundamental system-level activities like message delivery and clock synchronization to diagnose the system. This approach is complementary to graph-based application-level approaches. The diagnostic protocol is seen as a special case of consensus under the CFEM. The need for recording the duration and recurrence of errors and to assign them different severity levels has been pointed out.

Most previous diagnostic services provide snapshot-level information about a single manifestation of a fault. After a fault is detected, nodes are declared as either always permanent or always transient faulty. An evaluation of the effect on system reliability of these two different policies was conducted in [20]. The intuitive result was that optimal reliability is not attained by either.

In practice, nodes oscillate between faulty and correct behavior. To handle this, a range of mechanisms collectively called "count-and-threshold" schemes were established in our previous work [6], [7]. The idea is that "components should be kept in the system until the benefit of keeping the faulty component online is offset by the greater probability of multiple (hence, catastrophic) faults." Apart from the class of permanent faults, when the component always fails every time it is activated, a basic discrimination is done in the context of temporary errors spanning intermittents and transients: the first are due to faults internal to the component and show a high occurrence rate, which eventually might turn them to permanent faults, whereas the second are due to reasons that are external to the component, generally have an uncorrelated occurrence rate, and should not determine the exclusion of the component. Therefore, after detecting a transient, it is advocated to wait

and see if the error reappears before isolating the component. Error counters for each component are incremented when the node fails and decremented when it delivers a correct service. When a chosen threshold value is exceeded, the corresponding component is isolated.

The fundamental advantages, disadvantages, and trade-offs involved in using online count-and-threshold mechanisms and in defining the related thresholds have been characterized in [6], where a generic class of low-overhead count-and-threshold mechanisms, called $\alpha$-count, is defined. This model was elegantly extended in [7] to include double-threshold mechanisms, where a component is temporarily excluded after the first threshold is exceeded but still given an opportunity to be reintegrated and where more complex error distributions are considered. The applicability of the analysis is restricted to the case where error duration does not exceed the diagnostic period. Also, the presence of an "error detection subsystem" is assumed, which, for distributed diagnosis, means the existence of an underlying error detection, aggregation, and agreement service to support the threshold counting. The powerful $\alpha$-count model has also been implemented in the Generic Upgradable Architecture for Real-time Dependable Systems (GUARDS) architecture [25] for distributed diagnosis. A binary accusation on the node health is shared using consensus, voted upon, and given as input to the $\alpha$-function.

Many proposed diagnostic approaches use similar custom parametric schemes, paired with sophisticated statistical techniques, to discriminate between transient and intermittent faults [15], [21]. However, due to their complexity, these are not usable in an online mode.

### 1.2 Our Contributions

Most cited works on distributed diagnosis have focused on establishing the correctness and completeness of the diagnostic approaches for varied fault models. An often used assumption is that once a node fails, it must be isolated as fast as possible. This implicitly rules out the pragmatic issue that healthy nodes can suffer from transient outages that can be detected and treated but still do not require isolation. We introduce and examine a generic online FDIR framework, which is able to generalize the previously proposed distributed diagnosis protocols and to enhance them with count-and-threshold techniques in order to effectively handle transient faults.

In particular, our aim is to 1) determine the effect of the duration and recurrence of faults on the effectiveness of the online diagnosis protocols and 2) ascertain the sensitivity and the trade-offs of choices of some selected FDIR design parameters[2] in determining the correctness and completeness of the FDIR protocols and in improving system reliability.

We consider system parameters that can influence the effectiveness of the FDIR process. For example, in a synchronous distributed system, every node exchanges data at an epoch, also known as the *communication round*. As error detection takes place over each round, we can also consider it the minimal achievable *diagnostic round*. Over each communication round, the system health is "sampled" by the different nodes and exchanged by using the diagnostic protocol. In this context, the assumption that errors manifest only over a single round, as characterized in previous analyses, is not adequate. The length of the diagnostic round is a parameter that, together with other count-and-threshold parameters, will influence the likelihood with which a node is excluded from system operation. In fact, if the round is too short, then a transient fault may be perceived as permanent and, consequently, lead to pessimistic resource isolation. This can be particularly problematic for long-duration missions. On the other hand, if the round length is too large, then one would expect large diagnostic latencies in the system. This increases the probability of coincident errors within the same round and might be undesirable for critical applications with short mission times and requirements of rapid response to perturbations.

A discrimination between transient and intermittent or permanent faults solves two key problems: the depletion of system resources (and, consequently, of system resilience to faults) caused by the isolation of transient faulty nodes and the reduced coverage of the system fault hypothesis (that is, the assumption on the number of faults tolerated by the core system protocols within a given time window) if intermittent faulty nodes are left operative. Our contribution is to study the choice of diagnostic round length and other system parameters within an architectural context to highlight the correctness, completeness, and reliability trade-offs. We consider the following design parameters:

- *Diagnostic round rate*. The rate at which nodes exchange diagnostic data, aggregate it and, consequently, update penalties and rewards.
- *Penalty counter threshold values*. The number of temporally correlated diagnostic rounds, after which an erroneous node gets isolated.
- *Reward counter threshold values*. The number of diagnostic rounds, after which a node (previously suspected as erroneous) displaying correct behavior gets readmitted into the system as a "good" node.
- *Penalty increments*. The penalties assigned after errors with varied severities are detected.

We provide a generic FDIR framework that can be instantiated in multiple different implementations. Alongside, we provide stochastic techniques to examine the main trends related to the identified parameters. Furthermore, we report on the use of the FDIR approach and the related tuning techniques in prototypes for the automotive and aerospace domains. We discuss how different severity levels can be established and describe how, by means of a finer tuning, better settings could be found at design time, that is, without carrying out measurements on an implemented system.

The paper is organized as follows: Section 2 details the generic online fault diagnosis process supporting the FDIR algorithm, which is discussed in Section 3. Section 4 introduces the diagnostic measures and models used to evaluate the goodness of the design choices related to the FDIR process. Section 5 details the main trends involved in

2. We concentrate our analysis on parameters representing phenomena that are considered here for the first time, whereas we skip others already studied in [7], since their role is well established.

tuning the parameters. Section 6 reports on the practical application of the FDIR approach to prototypes from the automotive and aerospace domains. Section 7 summarizes our results and insights.

## 2 A Generic FDIR Framework

A generic FDIR framework consists of four key steps:

1. collection of local syndromes from internal and internode local *error detection*,
2. *dissemination* of the local syndromes,
3. *analysis* to consistently diagnose the faulty nodes, and
4. (possibly) *isolation* of the faulty nodes and *reconfiguration*.

The first three steps are generally carried out by a distributed diagnostic protocol. We propose to separate the diagnosis of faults (1-3) from the decision on the isolation of the node 4) and to isolate a node only if multiple instances of the diagnostic protocol indicate a sustained faulty behavior. In the following, we describe each of these steps in detail.

In order to establish a basis for our analysis, we present a system model and an associated count-and-threshold approach to support online diagnosis and FDIR. We consider a distributed system framework by using a round-based (synchronous) message dispersal protocol. Essentially, such a communication model implies that messages are broadcast and received by the system nodes periodically at specific times following an a priori deterministic schedule. A nonfaulty receiver node can identify the sender of an incoming message and can detect the absence or time deviance (early or late) for an expected message. It is important to mention that we have chosen a synchronous system model for simplicity of presentation. Our analysis developed in this paper can directly be extended to partially synchronous models (for example, timed asynchronous [11] or asynchronous augmented with failure detectors [8]), as long as there are mechanisms for 1) periodic error detection to form local syndromes and 2) authenticated channels to ensure that the sender of a message can be correctly identified.

As a comprehensive example of how the four steps apply to a distributed diagnostic protocol, we utilize the diagnostic protocol defined in [29] as a basic reference. We consider a CFEM [32], where faults can be either *benign*, that is, each node can locally detect the related errors, or *malicious*. The malicious faulty nodes can either send erroneous information symmetrically or asymmetrically. The latter case is the classical Byzantine case. The ability of a node to send correct messages in the designated time windows is used as a periodic diagnostic test. No assumption is made on the persistence of faults, as the correct delivery of each message is diagnosed independently. The protocol is able to diagnose bursts of multiple concurrent benign faults and to tolerate malicious faults.

### 2.1 Error Detection, Dissemination, and Analysis

During *error detection*, each node collects the evidence on system health that are locally observed at runtime. Besides self-checks executed by each node to detect internal errors and ensure error containment, online internode error detection is achieved through the constant monitoring of the message exchange. The result of this online monitoring occurring during system operation is condensed by each node into a *local syndrome* representing its local view of the correctness of the other nodes.

The granularity of the information stored in the local syndrome can vary. Different error classes, for example, missing message, late message, early message, wrong syntax, and corrupted cyclical redundancy checking (CRC), can be defined and associated with different severity levels. Also, errors impacting different system services with different criticality levels can be reported separately, allowing different fault-handling actions.

Due to malicious and symmetric faults, this local detection information is not sufficient to identify and locate faults. Therefore, local syndromes are *disseminated* to achieve a global view. This is periodically done at discrete points in time at the boundaries of what we call *diagnostic rounds*. Generally, the communication rounds and the diagnostic rounds coincide as local error detection and dissemination take place during each message exchange round.

After dissemination, each node *analyzes* the received information based on the fault model of the diagnostic protocol and derives a global and consistent *snapshot view* of the system state. The snapshot view assesses whether (and how) a node is faulty and ensures the following properties under the given fault assumption:

- *Correctness*. A correct node is never diagnosed faulty.
- *Completeness*. All benign nodes are diagnosed faulty.
- *Consistency*. All nodes agree on the same set of faulty nodes at each diagnostic round.

All of the local syndromes are collected to build a syndrome matrix, where each row represents a local syndrome, and each column contains all the local views on the health of a certain node. Similar to the second round of the protocol $OMH(1)$ [22], the analysis consists of performing a Hybrid Majority voting along the columns. The protocol ensures correctness, completeness, and consistency. In particular, it is able to detect all benign faults and those asymmetric faults that are detected by at least a majority of nodes. The presence of malicious faults cannot always be detected but does not disrupt consistency. A link fault is equated to a node fault, but the more sophisticated node-link discrimination approaches such as [13], [27] can be used as well. However, transient external faults on the communication network can be filtered out by using our FDIR algorithm.

### 2.2 Isolation of Unhealthy Nodes

The diagnostic protocol ensures that *each node* obtains, *at each* diagnostic round, consistent information on the manifested faults. This is called a snapshot view, as it refers to the detection of fault manifestations within a single diagnostic round.

Current approaches define *isolation* policies solely based on this snapshot view. Our purpose is to observe the behavior of the system over a given time interval before

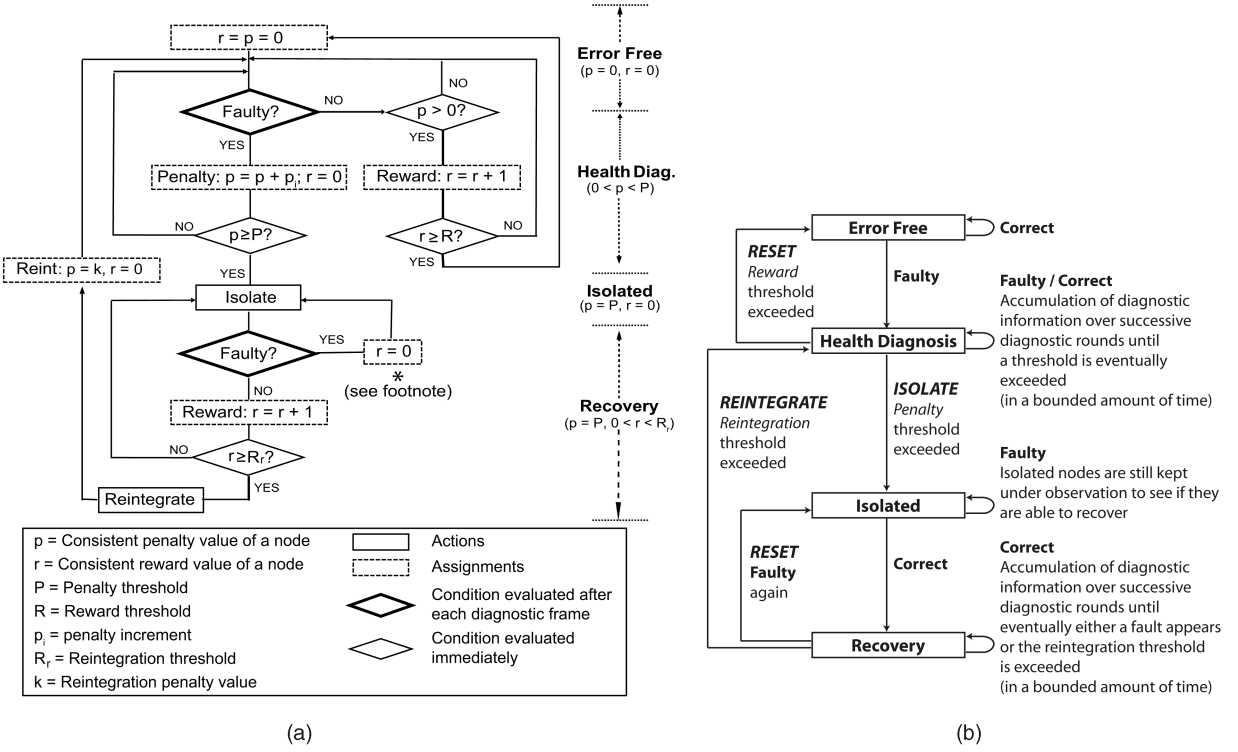


Fig. 1. The local online FDIR algorithm for each node. (a) Block diagram. (b) States (phases) that a node can visit. *A "permanency counter" can be put up to exclude a node as a permanent fault if it continues to remain faulty for a specified number of unsuccessful recovery attempts.

taking a decision on whether to isolate a node or not. Relying on correct, complete, and consistent snapshot views provided to each node by the distributed diagnosis protocol, we develop an expanded $\alpha$-function, extending on [7], that accumulates this data over successive diagnostic rounds to discriminate between unhealthy and healthy (although, at times, faulty) nodes.

Fault models typically used for diagnostic protocols do not consider the fact that faults can disappear and reappear, that is, the duration and recurrence of faults. We *extend* the fault models used by the diagnostic protocol and assume that at a given time, nodes can be

- *unhealthy*, if they have internal faults and fail in a permanent or intermittent manner, or
- *healthy*, if they fail only on external transients.

Healthy nodes can become unhealthy during system operation. We introduce these two terms to distinguish a node being faulty in a single diagnostic round from a node showing correlated subsequent failures. The goal of our FDIR protocol is to isolate only unhealthy nodes, whereas healthy nodes should be kept operative. In order to make this discrimination possible, we make two assumptions:

- (A1) Nodes can fail and recover an infinite number of times.
- (A2) Healthy nodes fail with lower frequency than unhealthy nodes.

These assumptions do not only arise from intuition but also reflect experimental results, as in [31].

## 3 The Parametric FDIR Algorithm

We propose to use a count-and-threshold algorithm on top of the diagnostic protocol to reduce the likelihood of isolation and increase the availability of healthy nodes in case of external transient faults. Each node executes the algorithm represented by the flow diagram in Fig. 1a and accumulates the observations of the health of all nodes obtained through snapshot views by using two values: a *penalty counter* and a *reward counter*. We describe the operations of the algorithm on a single node. As every update of the penalty and reward counters is based on the consistent snapshot view, it is ensured that all updates are executed consistently. Therefore, each node has the same penalty and reward counters for all nodes in the system. A node can be in one of the four possible states, each corresponding to the four phases of the FDIR algorithm, as depicted in Fig. 1b, namely, *Error Free*, *Health Diagnosis*, *Isolated*, or *Recovery*.

### 3.1 Error-Free and Health Diagnosis Phases

In the initial system state, each node is Error Free, and the values of the penalty and reward counters ($p$ and $r$ in Fig. 1a) are set to 0. The conditional block labeled *"Faulty?"* represents the content of the consistent snapshot view of the current diagnostic round.

As long as no errors from a node are detected, the algorithm loops in the Error-Free phase. After the node is diagnosed faulty for the first time, the system keeps the target node under observation for a finite time span to produce an assessment of its health and to isolate it only if the duration or recurrence of errors exceeds a tolerable rate.

This phase is called Health Diagnosis. Each time a node is diagnosed faulty, the related penalty counter is increased by a *penalty increment* reflecting the severity level. Conversely, if a node in the Health Diagnosis phase produces correct messages, then the reward counter is increased by 1 but is set to 0 as soon as another error appears. In Fig. 1a, the dashed boxes represent updates of the penalty and reward counters. The Health Diagnosis phase can have two outcomes (see Fig. 1b): 1) If the penalty counter exceeds a predefined penalty threshold $P$, then the node is isolated. 2) If the reward counter exceeds a predefined reward threshold $R$, then the diagnostic process is reset by setting *both* penalty and reward counts to 0. This process of updating and checking $p$ and $r$ is performed at each diagnostic round.

If the diagnostic protocol is capable of discriminating different severity classes of errors $\langle s_1, \ldots, s_n \rangle$, then these can be ordered in growing degree of criticality. Intuitively, a node showing more severe errors should be assigned higher penalty increments than other nodes with less severe errors in order to reach the penalty threshold faster. Therefore, different penalty increments $\bar{p} = \langle p_1, \ldots, p_n \rangle$ can be associated to different severity levels, where $p_1 < p_2 < \ldots < p_n$. In Section 6, we elaborate on how the choice of varied penalty increments can be tuned to satisfy desired system requirements.

We use two counters and two related thresholds to represent two different kinds of information. *Rewards* are related to the correlation between subsequent faults. In the algorithm in Fig. 1a, snapshot views are evaluated after each diagnostic round. The reward counter stores the number of consecutive fault-free diagnostic rounds that a node under Health Diagnosis displays. If the length of such rounds is $T$ time units, the reward threshold is thus reached after $R \cdot T$ time units without faults. In this case, subsequent faults are considered uncorrelated with the previous. On the other hand, *penalties* and the penalty threshold $P$ are related to the maximum length of tolerated faulty bursts before a node is isolated, which is $P \cdot T$ time units.

### 3.2 Isolated and Recovery Phases

Even though the introduction of the Health Diagnosis phase increases the availability of healthy nodes, the likelihood that long and bursty transients lead to incorrect node isolations cannot generally be ruled out, especially in cases of adverse external conditions or high-severity errors [29]. Therefore, we introduce a Recovery phase after node isolation, which provides an observation period to handle any residual faults and also to allow reintegrating a node that is incorrectly isolated.

If the penalty counter exceeds its threshold, then the Health Diagnosis phase ends, and the node goes into the Isolated state; that is, it is declared erroneous, and its participation on the ongoing computations in subsequent rounds is restricted. However, even in the Isolated state, the incriminated node continues, as long as it is viable, to participate by sending its messages at the prescribed instants, allowing a selective isolation based on the type of service that the node provides.

Potentially, the Recovery phase could allow unhealthy nodes to be reintegrated. Therefore, reintegrated nodes are assigned a penalty of $k > 0$ so that successive fault manifestations will lead to a faster reisolation. The reward threshold for recovery $(R_r)$ does not necessarily need to be equal to the reward threshold for diagnosis $(R)$ and can be adjusted, together with $k$, to handle this trade-off. Similar to the Health Diagnosis phase, the Recovery phase also reaches an outcome, either bringing the node back to the Isolated state or reintegrating it, after a bounded time of at most $R_r$ rounds.

In some systems, especially long-life systems, diagnosis is also mandated to signal when a node needs to be replaced. Representing this case requires that the Recovery phase in Fig. 1a is executed a finite number of times (for example, by setting a "permanency counter"), and if the node is not able to recover, it must be completely excluded. Such behavior is also recommended to limit the additional overhead involved in checking the behavior of an unhealthy node. A variation of the algorithm in this sense can be the use of a double-threshold approach [7], where a faulty node can continue accruing penalties after isolation, and if a second penalty threshold is reached, then the node is signaled for replacement.

## 4 Modeling the FDIR Effectiveness

In the FDIR process, the nodes of a distributed system are reconfigured using penalty and reward counters that are periodically updated at each diagnostic round until a threshold is reached. Thus, the design issue for online diagnosis and FDIR considered in this paper is given as follows:

*Given a system with specific transient and intermittent fault duration and reappearance times, what are the "best" parameter settings that minimize wrong isolations and maximize correct isolations? Note that the notion of "best" can change, depending on the design goals of the system, that is, whether the objective of the FDIR approach is to maximize the isolation of unhealthy nodes, minimize the isolation of healthy nodes, or increase the overall system reliability.*

In this section, we define the basis of the model used to evaluate the effectiveness of the FDIR approach and define stochastic measures for the FDIR effectiveness. These measures are functions of the specific aspects of the studied system (for example, fault duration and recurrence) and of the design parameters (for example, the diagnostic round rate and the penalty and reward thresholds).

### 4.1 Measures for FDIR Effectiveness

Of the four phases of the FDIR algorithm, we call the Health Diagnosis and Recovery phases *transitory phases*. The reason is that a node can remain in these phases only for a limited amount of time, and a diagnostic outcome is ensured within bounded time. The role of transitory phases is to discriminate between healthy nodes hit only by transient faults and unhealthy nodes showing an intermittently or permanently faulty behavior. The measures for FDIR effectiveness must reflect the capability of correct discrimination.

For the Health Diagnosis phase, we define two notions of completeness and correctness for healthy and unhealthy nodes, which we assign specific names to distinguish them from the similarly named properties of the underlying diagnostic protocols:

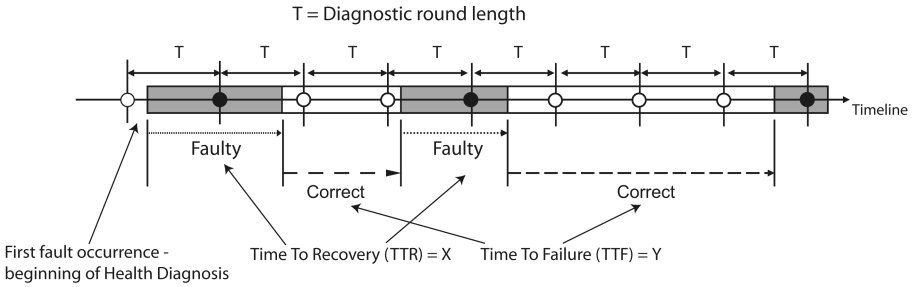


Fig. 2. Appearance and disappearance of faults.

- **Accumulated Correctness** (or *accuracy*) is the probability that a healthy node in the Error-Free state entering the Health Diagnosis phase is not isolated.
- **Accumulated Completeness** (or *coverage*) is the probability that an unhealthy node in the Error-Free state entering the Health Diagnosis phase is isolated.

As Accumulated Completeness trivially equals 1 for unhealthy nodes displaying permanent faults, we restrict our analysis to intermittent faults.

Besides Accumulated Correctness and Completeness, another measure of interest for the Health Diagnosis phase is the time needed to isolate unhealthy nodes:

- **Diagnostic Latency** is the interval between a node becoming unhealthy and its isolation.

Even if Health Diagnosis is a transitory phase and is terminated in a bounded time, a node can switch between the Error-Free and Health Diagnosis phases multiple times before being isolated. In [6], [7], the Health Diagnosis phase can last for an unbounded period of time. Therefore, two different measures were defined to capture this aspect: the overall diagnostic latency from the first fault appearance in an unhealthy node to its isolation $D$ and the fraction of unused lifetime of a healthy node $NU$, that is, the time between wrong isolation of a healthy node and its eventual transition to the unhealthy state divided by the time needed to become unhealthy from the beginning of its operational life. Our measures can be used to obtain $D$ and $NU$ by considering each execution of the Health Diagnosis phase as a Bernoulli trial, where success is node isolation.

Although unhealthy nodes should be kept isolated, healthy nodes should be reintegrated. Similar measures can be thus defined to describe the behavior of the algorithm after a node is isolated:

- **Stable Correctness** is the probability that an isolated healthy node entering the Recovery phase is reintegrated.
- **Stable Completeness** is the probability that an isolated unhealthy node entering the Recovery phase is not reintegrated.

The FDIR algorithm is a parametric algorithm. In the rest of this section, we relate the measures introduced in this section to the settings of the parameters.

### 4.2 Characterization of the System

We consider that nodes can alternate between periods of correct and faulty behavior, as assumed in Section 2.2. After a fault is activated, errors are observable for a time, which we term *Time to Recovery* (TTR), before they *disappear*. Eventually, errors will reappear either because of new transient faults or correlated intermittent faults. The time to error *reappearance* is called *Time to Failure* (TTF). This is depicted in Fig. 2. We can characterize the behavior of a given specific system by measuring or estimating the probabilities of error disappearance and reappearance in each diagnostic round.

The Health Diagnosis phase begins when a previously Error-Free node is diagnosed faulty. Nodes can pass from the correct to the failed states and back infinitely often (assumption A1). The TTR represents the permanence time in the faulty state before errors disappear and can be modeled by a continuous stochastic variable $X$, whose probability distribution function (pdf) is $f_X(t)$, and whose Cumulative Distribution Function (CDF) is $F_X(t)$. Once recovered, a node will eventually fail again. The TTF represents the permanence time in the correct state and can be represented by a similar continuous stochastic variable $Y$.

As the count-and-threshold algorithm receives data at discrete points in time corresponding to the diagnostic rounds, we study the behavior of the protocol as a discrete time problem, where the time unit is represented by the diagnostic round length $T$. The pdf of the discrete stochastic variable $\widehat{X}$ resulting from $X$ is

$$f_{\widehat{X}}(i) = \begin{cases} \int_{(i-1)T}^{iT} f_X(t)dt & \text{if } i > 0 \\ 0 & \text{if } i = 0. \end{cases} \quad (1)$$

The pdf of the discrete stochastic variable $\widehat{Y}$ can be obtained analogously.

In each diagnostic round following the manifestation of a fault, there is a probability, called *disappearance hazard* $d(i)$, that the fault disappears. It is the discrete hazard function of $f_{\widehat{X}}(i)$, that is, the probability of fault disappearance at diagnostic round $i$, conditioned by the fact that the error did not disappear in the previous round. It can be calculated as

$$d(i) = \frac{f_{\widehat{X}}(i)}{1 - F_{\widehat{X}}(i-1)}.$$

Analogously, for correct nodes, we can associate a *reappearance hazard* $m(i)$ to $f_{\widehat{Y}}(i)$, that is, a probability of fault reappearance in each diagnostic round.

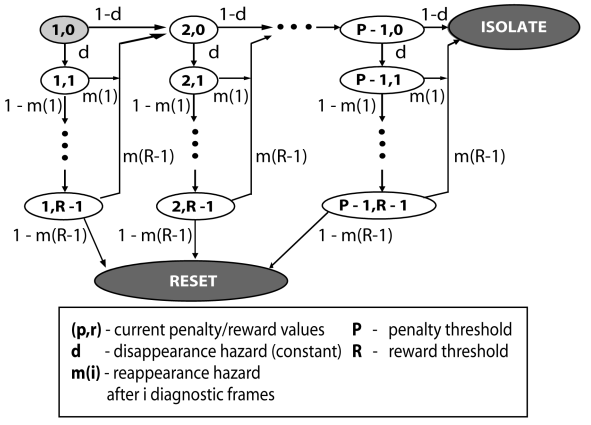


Fig. 3. DTMC for constant $d(i) = d$.

We define the *stochastic characterization* of the specific system under study as a quadruple $\langle d_h(i), m_h(i), d_u(i), m_u(i)\rangle$ composed by the disappearance and reappearance hazards at round $i$ of healthy nodes ($d_h(i)$ and $m_h(i)$) and unhealthy nodes ($d_u(i)$ and $m_u(i)$), respectively.[3] Assumption A2 in Section 2.2 can now be formalized by assuming the expected value of the distribution $m_u(i)$ to be much smaller than of $m_h(i)$. The multiple factors that influence $d(i)$ and $m(i)$ are discussed in more detail in Section 6.

### 4.3 Stochastic Models for the FDIR Algorithm

We use the disappearance and reappearance hazards $d(i)$ and $m(i)$ to model subsequent failures of a node over time instead of the logical predicates normally used by the existing diagnostic protocols. The properties of the protocol therefore become probabilistic and can be obtained by means of the stochastic models that we present below. As correct nodes consistently update penalties and rewards, we can use a single model to study the execution of the transitory phases of the FDIR algorithm in each correct node.

The measures of Accumulated Correctness and Completeness are defined based on the probability of isolation of healthy and unhealthy nodes, respectively, during an execution of the Health Diagnosis phase. To calculate them, we build a model of how the penalties and rewards associated with a node are consistently updated. We model the case of unary penalty increments upon errors, but the analysis can be easily extended to the case of different increments associated to varied severity levels. In fact, the probability of isolation when the penalty increment is 1, and the penalty threshold is $P$, is the same as if the increment is $p_j$, and the threshold $\lceil P/p_j \rceil$.

Values for $d(i)$ and $m(i)$ can be either expressed using an analytical distribution or defined using experimental results to assign a probability for each value of $i$. Regardless, it is possible to model the behavior of the protocol by using a Discrete Time Markov Chain (DTMC).

If the disappearance hazard is constant,[4] that is, $d(i) = d$, then the probability of isolation of a node after a failure and a subsequent single execution of the Health Diagnosis phase can be obtained from the simpler DTMC in Fig. 3. Each state is depicted as $\langle p, r\rangle$, representing that the node under consideration has accrued consistent penalty counter $p$ and reward counter $r$. In the initial state $\langle 1, 0\rangle$, the node has just displayed an error. Each transition models subsequent diagnostic rounds, where errors may be present or not. The probabilities of error disappearance and reappearance are $d$ and $m(i)$, respectively. States marked as $\langle p, 0\rangle$ follow the detection of an error and, consequently, $d$ is only used for their outgoing transitions. The other states follow a correct round and have outgoing transitions defined in terms of $m(i)$.

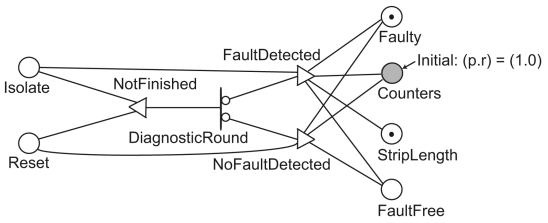


Fig. 4. DTMC for the general case.

In this particular case, the probability of isolation can be calculated (see the Appendix) as[5]

$$P_{isol} = \left(1 - d \cdot \prod_{i=1}^{R-1}(1 - m(i))\right)^{(P-1)}. \quad (2)$$

However, if the disappearance hazard is not constant but follows a generic distribution $d(i)$, then the complexity of the model grows, as $i$ must be represented in each state of the DTMC.[6] We can thus model it by using a higher level formalism such as the Stochastic Activity Network in Fig. 4 and solve it by using a tool like Möbius [12].

The places *Faulty* and *FaultFree* in Fig. 4 hold a token when the node is in the corresponding state. Therefore, in the initial marking, a token is put in the place *Faulty*, whereas the place *FaultFree* is empty. *Counters* is an extended place that stores the tuple $\langle p, r\rangle$ rather than simply tokens. The activity *DiagnosticRound* represents the execution of one diagnostic round. It has two cases associated with the probability of detecting a fault. If the node is faulty, then the probabilities associated with the two cases are $1 - d(i)$ and $d(i)$, respectively, whereas these are $m(i)$ and $1 - m(i)$ if the node is fault free. The output gates *FaultDetected* and *NoFaultDetected* update the penalty and the reward counters and check them against the threshold, possibly putting one token into the places *Isolated* or *Reset*. If this happens, the activity *DiagnosticRound* is disabled by the input gate *NotFinished*, and the model reaches an absorbing state. The output gate *FaultDetected* also adds a token in the place *StripLength*, which records the current number of diagnostic rounds $i$ from fault occurrence (respectively, disappearance) necessary to determine $d(i)$ (and $m(i)$). The

3. As notation, we add the subscripts $h$ and $u$ for the measures referring specifically to unhealthy (intermittent) and healthy (transient) faults. When we refer jointly to both cases, no subscript is added.

4. The fault models of [6], [7], which assume that faults disappears after a diagnostic round, represent a special case, where $d(i)$ is constant and equal to $d = 1$.

5. It can be observed that $\prod_{i=1}^{R-1}(1 - m(i))$ can also be calculated as the probability that a fault reappears before the reward threshold $R$ is reached; that is, $1 - F_Y((R-1)\cdot T)$ or, equivalently, $1 - F_{\widehat{Y}}(R-1)$ (see the Appendix for details).

6. For the reappearance hazard $m(i)$, the parameter $i$ is already implicitly defined by the current reward counter $r$.

model has two absorbing states characterized by the presence of a token in *Isolate* or *Reset*, respectively. The probability that the model reaches the first absorbing state is $P_{isol}$. The number of steps before an absorbing state is reached gives the *Diagnostic Latency* in terms of diagnostic rounds.

Accumulated Correctness and Completeness can be calculated from $P_{isol}$ by using the disappearance and reappearance hazards of, respectively, healthy and unhealthy nodes. Accumulated Correctness is the probability of *not* isolating a healthy node, whereas Accumulated Completeness is the probability of isolating an unhealthy node.

The Recovery phase can be modeled using a similar DTMC and is simpler, as only reward accumulation needs to be considered. In this case, the probability of reintegration upon error disappearance is the probability that further errors do not appear before the Reintegration threshold $R_r$ is reached:

$$P_{reint} = \prod_{i=1}^{R_r-1} (1 - m(i)).$$

As in the previous case, Stable Correctness and Completeness can be calculated from this expression by using the reappearance hazard $m(i)$ of healthy and unhealthy nodes. Stable Correctness is the probability of reintegrating healthy nodes, whereas Stable Completeness is the probability of *not* reintegrating unhealthy nodes.

This model is appropriate in those cases where replacement is not considered. There are also cases where a node, after some attempts to recover, is considered permanently faulty and is extracted from the system, as no benefit but only damage can be envisaged from keeping it operative. In such cases, a slight modification of the model is sufficient in order to count how many times the node enters the Recovery phase before being signaled for replacement.

## 5 Impact of the Design Parameters on Health Diagnosis

The modeling framework that we have defined previously allows system designers to tune the design parameters according to the specific system under study. This section provides an insight into the main issues and trends involved with the parameter tuning by evaluating the resulting values of Accumulated Correctness and Completeness and Diagnostic Latency (for brevity, these measures are also respectively termed *accuracy*, *coverage*, and *latency* in the rest of the paper). An example of harder tuning in a different scenario is also described, where the expected TTF for healthy and unhealthy nodes is similar. Finally, the impact of the FDIR parameters on reliability is highlighted.

For our trend analysis, we consider a generic automotive system. The average TTR $E[X]$ is 5 ms, and we consider four discrete distributions of $\widehat{X}$: binomial, geometric (where the hazard $d(i)$ is constant), Poisson, and uniform. For finite-support distributions, we assume a maximum TTR $X_{max} = 10\,\mathrm{ms}$, as in [14]. The small differences in the accuracy and coverage obtained for different distributions of $\widehat{X}$ confirms that in this case, the simplified model in Fig. 3 and the related closed-form analytical expression (2) provide a good approximation. The TTF (transient) for healthy nodes $Y_h$ is assumed to be exponentially distributed, with an expected value $E[Y_h] = 1,000$ hours, and the TTF (intermittent) for $Y_u$ for unhealthy nodes follows a Weibull distribution [31], with increasing failure rate $(\alpha = 1.4)$ and an expected value $E[Y_u] = 1$ hour. Therefore, according to assumption A2 in Section 2.2, we assume that $E[Y_h] > E[Y_u]$. While conducting sensitivity analysis on each design parameters, we fix the others to the nominal values $P = 5$, $R = 10^6$, and $T = 5\,\mathrm{ms}$. Table 1 summarizes the considered design and system parameters with their nominal values. Initially, we consider unity penalty increments.

### 5.1 Tuning of the Design Parameters

The analysis confirms that the diagnostic round length has a strong impact on the measures of interest. The longer the diagnostic rounds, the higher the probability of observing an event during a round, either recovery or failure. The resulting accuracy and coverage are given in Fig. 5a. In this figure, as in some other following ones, multiple curves overlap with each other. However, all figures display conformal trends. We can observe the presence of points of minimal coverage $(T = 1 - 5\,\mathrm{ms})$ and maximal accuracy $(T = 10\,\mathrm{ms} - 1\,\mathrm{sec})$. In fact, if the diagnostic rounds are too short, then nodes do not have enough time to recover before the penalty threshold is reached and are always isolated. In this case, the accuracy is 0, and the coverage is 1. The same result is obtained when the diagnostic round is excessively long. In this case, the period of correct operation before the counters are reset becomes too long, and even subsequent transient faults are considered as correlated. Overall, we cannot consider a setting of the penalty and reward thresholds as good *per se* without considering the diagnostic round length.

In Fig. 5a, as well as in the subsequent figures, the results obtained using different distributions of the TTR $\widehat{X}$ are similar, especially if both measures are close to 1. Therefore, when a highly refined tuning is not necessary, the geometric distribution, where $d(i) = d$, can be adopted for the analysis. This enables using the closed-form analytical expression of (2) rather than simulations or numerical analysis.

The average latency of isolation of unhealthy nodes at varying values of $T$ is plotted in Fig. 5b. As expected, increasing the length of the diagnostic round also increases the time necessary to isolate an unhealthy node. However, for values of $T$ greater than 100 ms, the latency tends to grow much less. The reason is that when the coverage is close to 1, isolation is usually achieved, in a bounded time, after the first Health Diagnosis following the fault. Therefore, the latency depends on how many error bursts are necessary to reach the penalty threshold. For finite-support distributions, the latency is higher than the one observed for other distribution, especially when diagnostic rounds are large enough to make it impossible for a single error burst to determine node isolation. In general, the longer the tail of the distribution of the TTR, the shorter the diagnostic latency.

The impact of the reward threshold $R$ on the average accuracy and coverage is depicted in Fig. 6a. Healthy and

TABLE 1
Design and System Parameters and Their Nominal Values

| Parameter | Description | Nominal values |
|---|---|---|
| $P$ | Penalty threshold | 5 |
| $R$ | Reward threshold | $10^6$ |
| $T$ | Diagnostic round length | $5ms$ |
| $f_{\hat{X}}(i)$ | Discrete distribution of Time To Recovery | Several distributions |
| $E[X]$ | Expected Time To Recovery | $5ms$ |
| $X_{max}$ | Maximum Time To Recovery (binomial and uniform distributions) | $10ms$ |
| $f_{Y_u}$ | Continuous distribution of Time To Failure for unhealthy nodes | Weibull ($\alpha = 1.4$) |
| $E[Y_u]$ | Expected Time To Failure for unhealthy nodes | $1h$ |
| $f_{Y_h}$ | Continuous distribution of Time To Failure for healthy nodes | Exponential |
| $E[Y_h]$ | Expected Time To Failure for healthy nodes | $1000h$ |

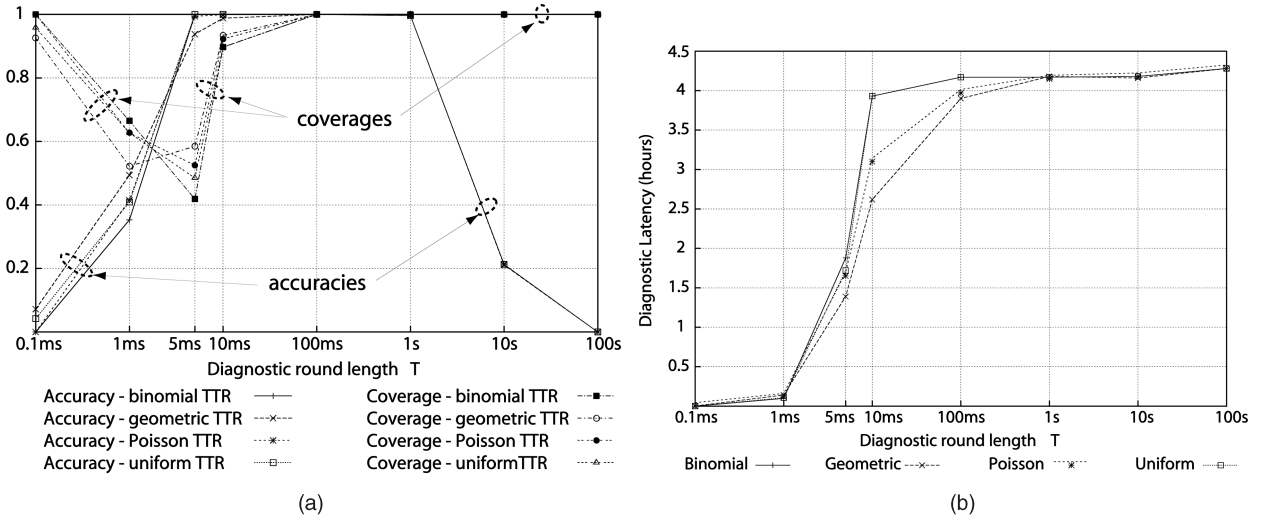


Fig. 5. Sensitivity analysis for the diagnostic round length $T$. (a) Accumulated correctness (accuracy) and completeness (coverage). (b) Diagnostic latency (time to isolation of unhealthy nodes).

unhealthy nodes are discriminated based on their TTF. The FDIR algorithm is designed such that a node that always fails before reaching $R$ is always isolated, independent of the TTR. Thus, proper tuning of $R$ is essential to obtain a good discrimination. The trade-off faced in this case is that before resetting the counters, the algorithm must wait long enough to correlate successive intermittent faults (for coverage) but not so much that independent successive transient faults also get correlated (for accuracy). The best trade-off in our example is found for settings around $R = 10^7$. Penalty and rewards are reset to 0 after $R \cdot T \simeq 14$ hours, which is enough to correlate intermittent faults (activated every hour on the average) but not to correlate transient faults (appearing every 1,000 hours on the average).

The average diagnostic latency for varied values of $R$ is reported in Fig. 6b. Similar to the previous sensitivity on $T$, the latency converges to a constant value when the coverage is close to 1. The reason is that once the protocol is set to wait enough to catch the reappearances of intermittent errors with a high probability, it will not likely wait for a longer time if the reward threshold is further increased. The asymptotic constant value depends on the number of faulty bursts necessary to reach the penalty threshold and is dependent on the specific distribution of the TTR that we consider. In our example, a setting of $R = 10^7$ allows capturing most of the intermittent errors.

Finally, we consider variations of the penalty threshold $P$, as illustrated in Fig. 7a. $P$ is the maximum number of faulty diagnostic rounds that a node is allowed to exhibit before assessing it as unhealthy. Tuning $P$ can reduce the probability of incorrect node isolations due to transient error bursts, but this alone is not sufficient to obtain high levels (that is, $> 0.9$) of accuracy *and* coverage, unless a proper distinction between healthy and unhealthy nodes is made by tuning $R$. As expected, the accuracy for finite-support distributions (binomial and uniform) is 1 as soon as the time to isolation for a single faulty burst ($(P-1) \cdot T$) is larger than their support ($X_{max} = 10$ ms).

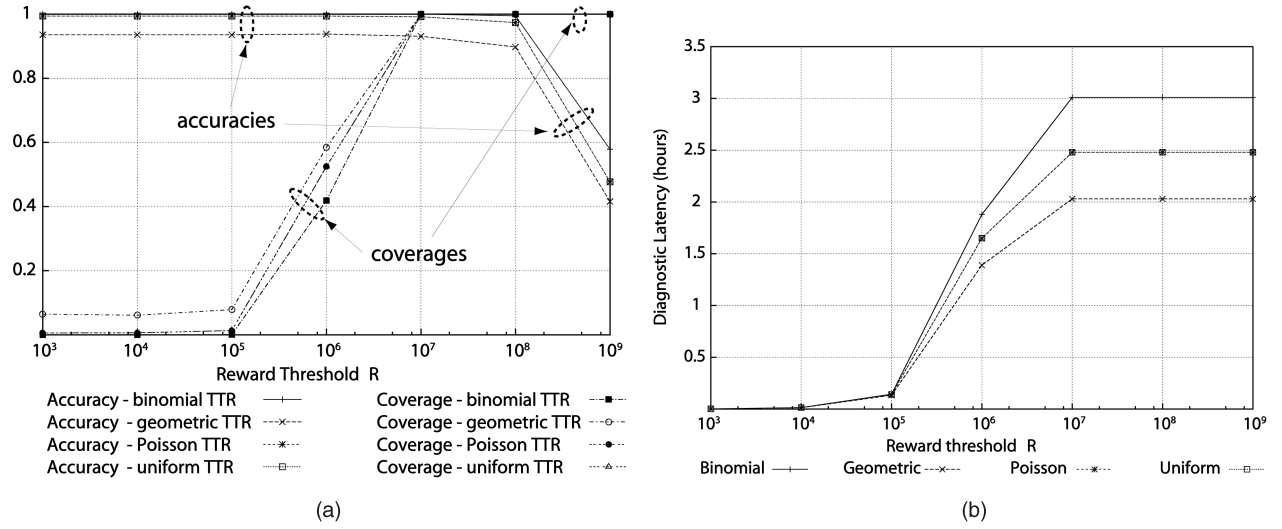


Fig. 6. Sensitivity analysis for the diagnostic round length $R$. (a) Accumulated correctness (accuracy) and completeness (coverage). (b) Diagnostic latency (time to isolation of unhealthy nodes).

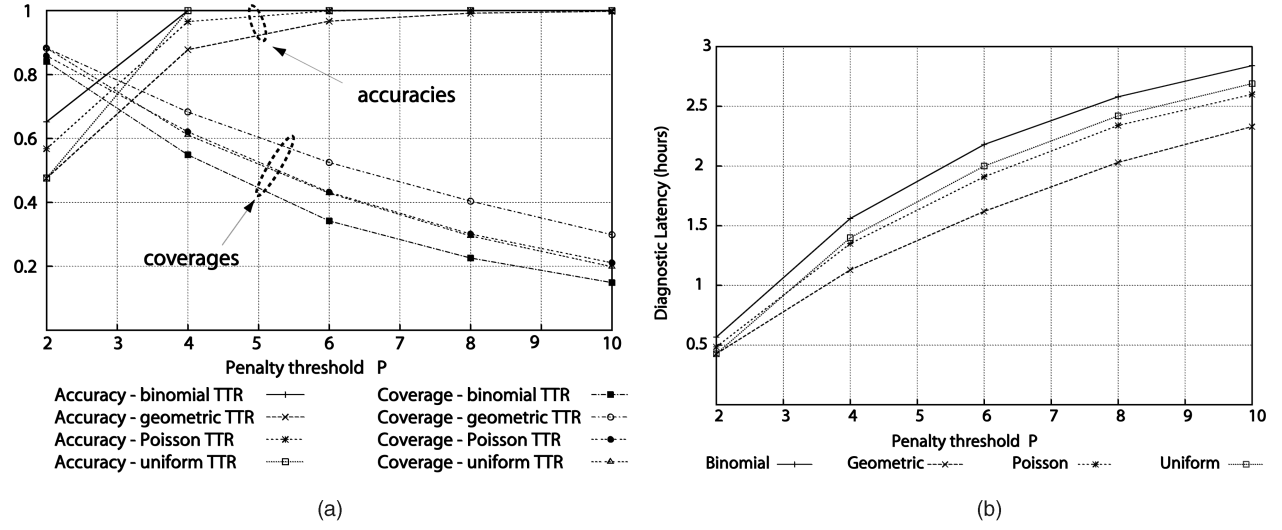


Fig. 7. Sensitivity analysis for the diagnostic round length $P$. (a) Accumulated correctness (accuracy) and completeness (coverage). (b) Diagnostic latency (time to isolation of unhealthy nodes).

By increasing $P$, more alternating periods of faulty and correct behavior are needed to achieve isolation of unhealthy nodes. Therefore, a single error burst will less likely result in isolation, and the trends of coverage of the diagnostic latency are opposite (see Fig. 7b).

The current analysis considers unary penalty increments. For high-severity faults, it is possible to increase the penalty increment to favor coverage and reduce diagnostic latency, even if this comes at the cost of reduced accuracy. From an analysis standpoint, the case of nodes displaying faults with a related penalty increment $p_i > 1$ when the penalty threshold is $P$ is equivalent to the case of unary increments when the penalty threshold equals $\lceil P/p_i \rceil$. Also, if a node is reintegrated and assigned a reintegration penalty $k$, then the probability of reisolation in cases of subsequent faults before the counters are reset can be evaluated as if the penalty threshold was $P-(k+1)$, and its initial penalty counter was 1.

### 5.2 An Example of a Harder Tuning

The previous analysis has shown that by tuning the design parameters, the protocol can distinguish between the higher frequency of failure of unhealthy nodes and the lower frequency characterizing healthy ones. It is intuitive that the higher the difference in frequency between healthy and unhealthy nodes, the easier it is to find a correct tuning of the parameters. To confirm this, we evaluated the case when the average TTF is one order of magnitude lower (100 hours instead of 1,000 hours) for healthy nodes and 1 order of magnitude higher (10 hours instead of 1 hour) for unhealthy nodes. In this case, finding a good trade-off between accuracy and coverage becomes harder, as shown in Fig. 8. Different from the previous case, it can be observed that tunings with high values (greater than 0.9) for both accuracy and completeness do not exist. Thus, trade-offs accounting for the relative importance of the two properties must be pursued. Also, in this case, a more

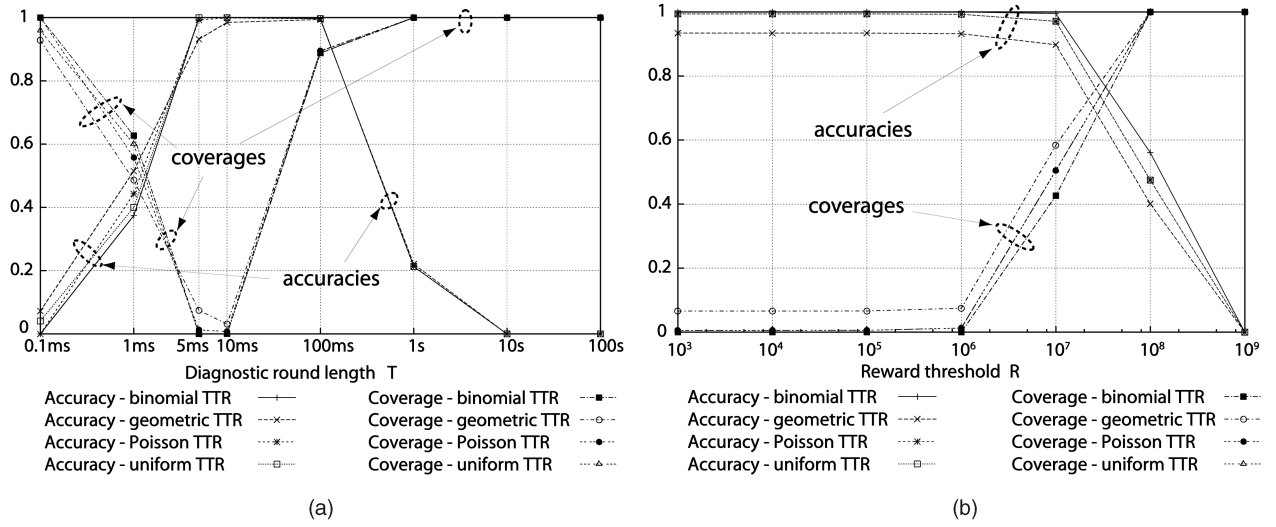


Fig. 8. Hard tuning: sensitivity analysis for $T$ and $R$ (Accumulated Correctness and Completeness). (a) Hard tuning: sensitivity for $T$. (b) Hard tuning: sensitivity for $R$.

refined analysis using simulation or numerical analysis might be required.

### 5.3 Reliability Issues

To give an example of the impact of the FDIR design parameters on reliability, we consider a four-node system tolerating at most one fault at a time, where node availability is thus highly critical. Although our evaluations are limited and do not explore the full set of the design space, some conclusions can be drawn, without the intent of generalizing much beyond the considered fraction of the design space.

We consider the parameters as in the previous analysis, but with small modifications. To enable Markov analysis, the TTF and TTR of healthy and unhealthy nodes follow an exponential distribution. Nodes are expected to fail every 1,000 hours on the average and to become unhealthy upon failure, with a probability of 0.1. For simplicity, the Recovery phase is not included, and isolated nodes are assumed to be substituted with an exponentially distributed delay, with an expected value of 5 hours. It is conservatively assumed that when a node fails, the probability that the system fails due to nearly coincident faults is given by the probability that any other node fails during $P$ rounds, which is the maximum number of allowed faulty diagnostic rounds within a single Health Diagnosis phase. In order not to underestimate the risk of coincident faults, we consider a conservative scenario where unhealthy nodes fail, on the average, approximately every 4 seconds. Even under these conservative assumptions on nearly coincident faults, we observe that reliability can benefit from delaying the isolation of faulty nodes.

The Mean TTF (MTTF) of the system resulting from different tunings of $P \in [1, 100]$ and $R \in [10^5, 10^8]$ is depicted in Fig. 9. At first, one can perceive how the different design settings of the FDIR parameters can result in very different trends. The gain can be as large as one order of magnitude. The plots confirm that the best tunings of $P$ and $R$ are not at the extremes. If $R$ is too low, then unhealthy nodes are kept in the system, and this increases the probability of nearly coincident faults. On the other hand, an excessive value of $R$ captures also independent transients faults as correlated and leads to resource depletion.

Despite conservative assumptions on the likelihood of coincident faults and due to the low resilience of the assumed system, an even higher sensitivity is observed with respect to $P$. The drift is evident if we compare the setting $P = 1$ with $P = 10$, which allows a relatively small number of faulty rounds before isolation. This indicates that the relative importance of optimizing accuracy or coverage changes, depending on the specific system architecture. In this example, accuracy turns out to be more critical than coverage. The case of $P = 1$ represents the classic FDIR approach, where all the faulty nodes are considered unhealthy and, therefore, isolated. The other alternative, where nodes are always considered to be healthy, can be approximated by $P \to \infty$, and it is also not optimal due to nearly coincident faults. This confirms the results already available in the literature related to the discrimination of transient and intermittent or permanent faults (for example,

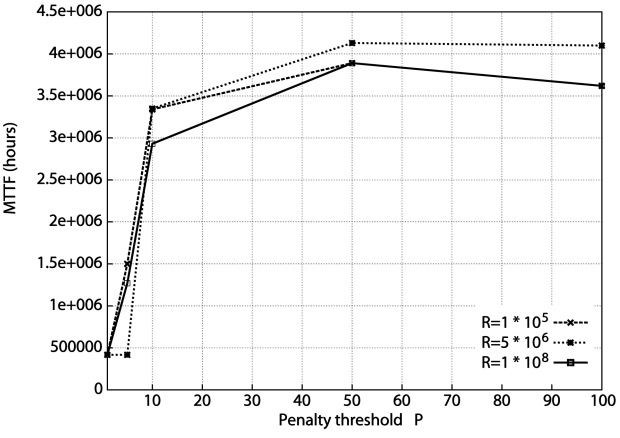


Fig. 9. Reliability of a system for varied $P$ and $R$.

TABLE 2
Application-Specific Requirements on the Diagnostic Protocol

| Domain | Criticality class | Example | Tolerated outage | $T$ | $p_i$ | $P$ |
|---|---|---|---|---|---|---|
| Automotive | *Safety Critical (SC)* | X-by-wire | $20-50ms$ | $2.5ms$ | 49 | 196 |
| | *Safety Relevant (SR)* | Stability control | $100-200ms$ | | 6 | |
| | *Non Safety Relevant (NSR)* | Door control | $500-1000ms$ | | 1 | |
| Aerospace | *Safety Critical (SC)* | High Lift, Landing Gear | $50ms$ | $2.5ms$ | 49 | 196 |

[20]) and indicates the soundness and correctness of our analysis.

These results prove that the FDIR parameters can have a considerable impact on the reliability of the system. By fixing dependability goals (and, hopefully, more detailed goals for the attributes like safety, reliability, and availability), it is then possible to look for the required levels of accuracy, coverage, and latency for a given system. As we have shown previously, the possibility of finding values optimizing the contrasting attributes depends very much on "external" system parameters, which are not under designer control (for example, failure rates), and on other design parameters such as the completeness of the diagnostic protocol and of its fault assumption, which we did not specifically address in this work because they are not directly related to the count-and-threshold algorithm. Even if some parameters are not known, we describe next how our framework provides the system designers with techniques to study the effect of different design choices under a range of scenarios.

## 6 Practical Application of the FDIR Framework

Our evaluation approach has been applied to tune two prototypes, an automotive system and an aerospace system, running on a system implementing our FDIR framework. All requirements and design parameters used during the tuning below arose from actual automotive and aerospace applications [29].

During initial implementation, a reasonable though approximate setting of the FDIR parameters was established, which ensured high coverage of intermittent faults ($> 0.999$) only, as long as the related expected TTF was in the order of minutes. Successive use of the tuning process documented in this section significantly enhanced the initial setting. First, the application-level requirements constraining the diagnostic parameters were considered. Next, a range of realistic scenarios were defined in order to tune the unconstrained parameters. Without compromising accuracy, the new setting extended the range of unhealthy nodes isolated with high likelihood to those failing on the average as seldom as every 10 hours. This could be done at design time, as no measurement on the system was needed.

### 6.1 Application-Specific Design Constraints

In the time-triggered platform used for the implementation all nodes share a common (replicated) broadcast-based bus using a TDMA access scheme. Nodes consist of a host computer (Infineon Tricore 1796) and a communication controller (Xilinx Vertex 4 field-programmable gate array (FPGA)) providing interface to a generic time-triggered network (layered TTP). Each node is statically assigned a time window, called *sending slot*, to broadcast messages to all other nodes. Nodes are diagnosed based on their capability of sending messages during the designated sending slot. Therefore, the diagnostic round length $T$ equals a time-division multiple access (TDMA) round. Both considered that safety-critical domains are characterized by strict application requirements, which define the range of the feasible parametric settings. The TDMA round (and, consequently, the diagnostic round) must be short enough to allow satisfying all the application-level hard real-time deadlines.

The diagnostic protocol is also constrained by requirements related to the criticality of different applications. In automotive systems, multiple criticality classes can be identified. *Safety-Critical* (SC) functionalities are necessary for the physical control of the vehicle with strict reactivity constraints, for example, X-by-wire. Recovery actions must be timely and always preserve the availability of the (possibly degraded) service. *Safety-Relevant* (SR) functionalities support the driver, for example, the Electronic Stability Control and the Driver Assistant Systems. They are not necessary for the control of the car, but the driver must know if these are unavailable. Finally, we considered *Non-SR* (NSR) functionalities such as comfort and entertainment subsystems. In the aerospace prototype, only SC functionalities are running on the system, for example, the High Lift system related to the control of flaps and the Landing Gear system. A summary of the requirements is shown in Table 2.

Applications with different criticality classes have different requirements on the maximum *tolerated transient outage* time between the beginning of a faulty burst and the isolation of the node (and the consequent activation of recovery actions). As discussed in the previous section, penalty thresholds greater than 1 increase accuracy and node availability in the presence of transient faults. However, during the Health Diagnosis phase of the FDIR algorithm, an application might be prevented from correctly exchanging messages if some of its jobs are hosted on a faulty node that is still kept operative. Therefore, the maximum tolerated outage represents an upper bound of the sum of three delays: the *detection delay* to detect a fault for the first time, the *accumulation delay* when faults are continuously recorded by the diagnostic protocol, but the

node is still not isolated, and the *recovery delay* involved in triggering recovery actions.

The diagnostic protocol operates by detecting if time-triggered messages are delivered correctly and timely. As in the prototype system, all messages generated by the different jobs of a node are encapsulated into a single message, and the protocol does not discriminate between faults at different services running on the same node. A single criticality class, as well as a related penalty increment $p_i$, is thus assigned to each node according to the tolerated outage of its highest criticality job.

In the specific setup under consideration, the worst-case detection delay is four TDMA rounds (10 ms), and local recovery actions are immediately triggered as soon as a node is consistently isolated. The penalty threshold is first defined by considering the class of applications with the least criticality (NSR). We conservatively considered the shortest tolerated outage associated with the three criticality classes identified. In this case, a faulty node must be isolated after bursts of 500 ms. Considering the detection delay, the time from detection to isolation must be at most 490 ms, which corresponds to 196 diagnostic rounds. We consequently set the penalty threshold to $P = 196$ and the penalty increment to $p_{NSR} = 1$. After establishing the penalty threshold, the penalty increments for application classes with higher criticality can be derived. The maximum number of tolerated diagnostic rounds for the other two criticality classes ($tol_{\{SC,SR\}}$) can be similarly calculated by subtracting the detection delay from the tolerated outage. The penalty increments must ensure that the penalty threshold $P$ is reached within $tol_{\{SC,SR\}}$ diagnostic rounds, that is, $p_{\{SC,SR\}} = \lceil P \cdot T / tol_{\{SC,SR\}} \rceil$. It is remarkable that the resulting values reported in Table 2 are just slightly more conservative than the values experimentally identified in [29], reflecting our conservative assumptions.

### 6.2 Characterization of the System

Our tuning process allows evaluation of accuracy and coverage levels resulting from different parametric settings of the FDIR process. In our case study, $P$, $T$, and the penalty increments are constrained by domain-specific requirements. We are thus interested in setting $R$ in order to correlate the largest range of intermittent faults while avoiding an excessive reduction of accuracy due to correlation of successive external transient faults.

The tuning process requires three input parameters related to the system: the TTR of all nodes, the (transient) TTF of healthy nodes, and the (intermittent) TTF of unhealthy nodes. These values can be known from standards, expertise, or literature. When precise values are not available, a range of reasonable scenarios must be examined for sensitivity analysis.

The transient TTF for different classes of faults and operational conditions has been extensively studied and sometimes included in standards (see [20] for a survey on published rates for different types of faults that are typical of embedded systems). Reported values are all well below a rate of $10^{-3}$ faults/$h$. In our case, we considered two conservative rates of $10^{-2}$ and $10^{-3}$ faults/$h$ to account for the foreseen trend toward higher rates [9]. The TTF for intermittent faults is system specific and it depends on multiple factors such as the specific component being damaged or the activation patterns of the software. As a result, this value is unknown in most practical systems. Therefore, we consider an intermittent TTF following a Weibull distribution, with $\alpha = 1.4$ [31], and expected values ranging $[1 \text{ minute} - 100 \text{ hours}]$. Regarding the expected TTR, safety-critical systems are often validated by injecting temporary faults and observing the capability of the system to tolerate them. Such tests are supposed to represent real-world operational conditions. According to the International Organization for Standardization (ISO) 7637 testing standard for the automotive domain [14], we considered an expected TTR of 5 ms, which is also a reasonable value for the aerospace domain.

### 6.3 Tuning the System to Improve Coverage

The defined scenarios were used to study the coverage and accuracy levels for the three criticality classes with respect to different tunings of $R$ and to determine 1) how large $R$ can be set before accuracy is compromised and 2) what the largest expected TTF for intermittent faults resulting in high levels of coverage is.

The value $R = 10^6$ was chosen as the first setting in the context of the experimental validation of the protocol [29], as it appeared as a good practical trade-off. In fact, it allows correlating all intermittent faults appearing within a time window $R \cdot T \cong 42$ minutes, whereas two distinct transient faults are incorrectly correlated with a probability lower than 1 percent in the scenarios that we considered. In the following, we show that a better tuning can be found by means of a more extensive probabilistic evaluation.

The plots of accuracy and coverage for the three severity classes are depicted in Fig. 10 and were obtained using the simple closed-form expression of (2). Besides the fact that the main trends are consistent with those identified in the previous section, some interesting aspects specific to the application domains under consideration emerge. As expected, nodes with higher criticality display better coverage and worse accuracy.

The value selected during the experimental evaluation of the protocol $R = 10^6$ ensures a very high level of accuracy (above 0.98) for all criticality classes and transient fault rates. Unhealthy nodes are isolated with high coverage if their expected TTF is in the order of minutes. For larger expected TTF (greater than 1 hour) isolation is unlikely even if SC nodes are impacted.

In general, we observe that by tuning of $R \leq 10^8$, a level of accuracy above 0.97 can be reached for all criticalities, the sole exception being the case of SC nodes with the conservative transient TTF of 100 hours. In this case, accuracy drops to around 0.78. On the other hand, settings of $R \geq 10^8$ result in high coverage (above 0.9996) for all criticalities classes and for expected intermittent TTF $\leq 10$ hours.

Such tuning highlights that a better setting of $R = 10^8$ exists, which significantly improves coverage without impacting accuracy, especially if the transient TTF can be safely assumed to be more than 1,000 hours. Due to more restrictive requirements on the maximum tolerated outage, SC nodes are more likely subject to wrong isolations (especially under adverse external conditions when external

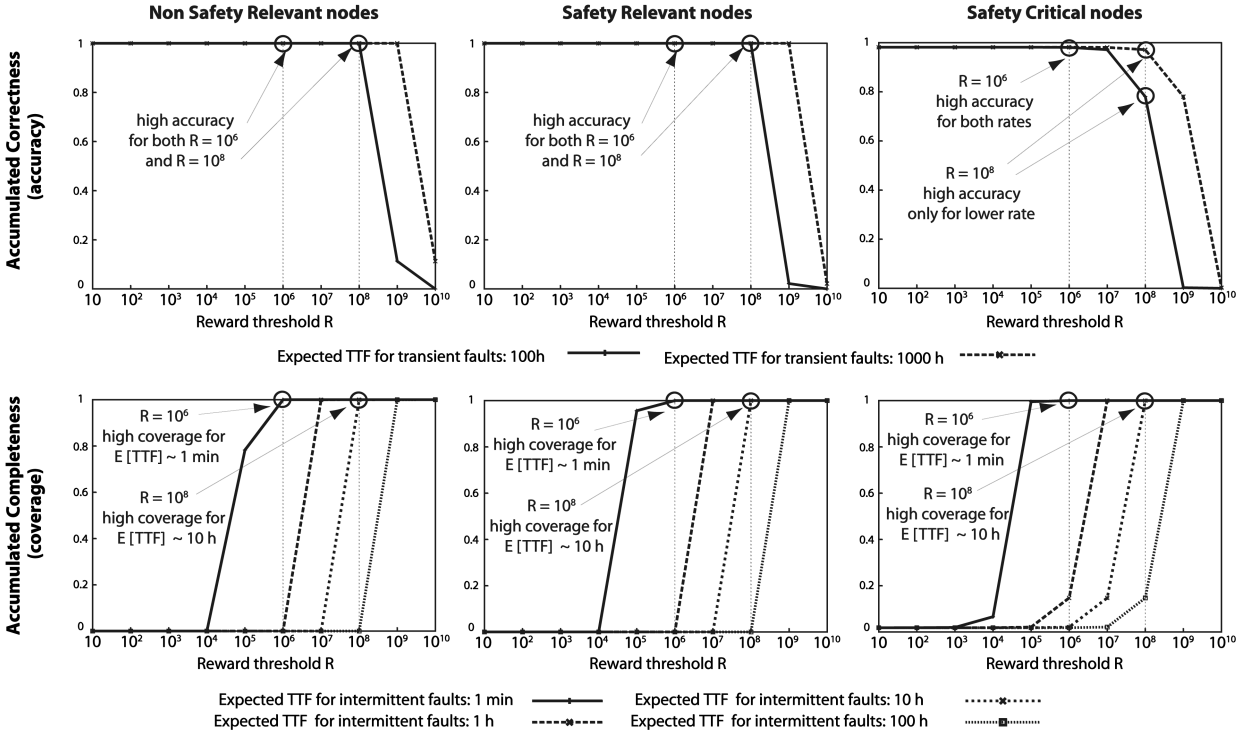


Fig. 10. Tuning of $R$ for the automotive and aerospace prototypes.

faults are more frequent than normal [29]). However, the Recovery phase of the FDIR algorithm can guarantee eventual reintegration of healthy nodes.

### 6.4 Determination of the System Parameters

The parametric tuning that we have shown requires, as a prerequisite, an estimation of two stochastic distributions that characterize the system: the probability of error disappearance $d(i)$ and the probability of successive error manifestation $m(i)$.

In our model, we have explicitly considered aspects of error duration. Conceptually, we have utilized the notion of *decay time*, that is, the length of time that an error would be present if a fault was activated for an instant. Thus, the error is the effect of an instantaneous fault activation at time $t_0$, which lasts for a time $t_0 + dec$.

Errors that have shorter *decay times* will have less time to further impact system operations. For example, a lost bit on a communication link due to a transient fault should be considered as an error with a short *decay time*. If a noise pulse affects the link, then some time will need to pass before the energy is dissipated from the medium. During this time, the messages being sent may be corrupted, depending on the level of noise. Another example would be a memory module with scrubbing. When an error occurs, there will be a time period where the error could propagate and induce further errors. Once the scrubbing mechanism detects the error and removes it, the immediate danger of error propagation will have lapsed (even though the erroneous source may still be present). In general, we can say that all the local recovery actions that are taken to handle the effect of errors can influence the distribution of $d(i)$. Decay rates can be determined if regular and predictable times exist, where errors can be detected or removed. Otherwise, it is prudent to assume a worst-case scenario.

### 6.5 Severity and Fault/Effect Binding

Beyond decay time, we have discussed how, if a function in the system core is impacted by a transient, it may be necessary to deal with it immediately instead of following a penalty counter-based FDIR procedure. This issue can be addressed in a number of ways. The first approach would be to assess a penalty so severe that it causes exclusion immediately so that further reliance on error detection is not needed. A second method would be to try identifying the worst case detection time by a higher level mechanism. This method may allow for some error propagation until it begins affecting a critical higher level function. A third alternative is to schedule more extensive FDIR tasks to collect more information while imposing greater overhead.

Understanding how faults can manifest as errors remains a central though extremely complex issue. There are many techniques such as failure mode and effects analysis (FMEA) that can be used to carry out this kind of analysis. In general, binding faults and the visible and detectable errors that they manifest with can be useful for two reasons: to assess the *severity* of the state that the system is in and to suggest the *best recovery or maintenance action* that might be taken.

## 7 Conclusions

We have introduced a comprehensive FDIR framework that combines a diagnostic protocol used to obtain "snapshot" diagnostic information on faulty nodes at each round with a

count-and-threshold algorithm, which accumulates this information, to produce a health assessment of the nodes taking transient faults of varied duration explicitly into account. The FDIR framework also details the recovery phase of a system as an organic part of the diagnostic process, considers the definition of varied severity classes, and makes proposals for their management.

This work has made contributions in 1) determining and establishing the effect of the duration and recurrence of errors on the effectiveness of online diagnosis protocols, 2) ascertaining the sensitivity and the trade-offs involving some FDIR design parameters in determining the correctness and completeness of the FDIR protocols and in improving system reliability, and 3) describing an application of the approach on two practical systems.

By developing a generic and comprehensive analytic framework, we have been able to provide methods to guide and ease the tuning of the parameters. We have shown that design parameters such as the diagnostic round length, which influences the performance of the system, can also considerably impact the system reliability and task-oriented availability. Thus, depending on the failure modes expected in a particular environment, the system designer can optimize the FDIR algorithm to minimize wrong isolations with the increased task-oriented availability of the system. We identified the main trends by means of a sensitivity study, varying the different FDIR parameters within reasonable bounds. Finally, we have shown the practicality of the approach by implementing and tuning it onto two prototypes for automotive and aerospace applications, addressing open issues such as the determination of proper severity levels for different classes of errors. Without violating any application-level constraints, the achieved probability of node isolation due to transient faults is almost negligible, whereas nodes with internal dormant faults are isolated, even if errors appear as seldom as every 10 hours.

## APPENDIX

In this section, we solve the DTCM of Fig. 3 and obtain the result of (2).

**Theorem.** *Consider an FDIR process with penalty threshold $P$, reward threshold $R$, diagnostic round length $T$, and unary penalty increments. If a node with a constant disappearance hazard $d(i) = d$ and reappearance hazard $m(i)$ enters the Health Diagnosis phase, then it is isolated with a probability:*

$$P_{isol} = \left(1 - d \cdot \prod_{i=1}^{R-1}(1 - m(i))\right)^{(P-1)}.$$

**Proof.** We solve the chain by adding to it two dummy transitions having probability 1 from the absorbing states to the initial state $\langle 1, 0\rangle$, thus modeling an infinite number of execution of the Health Diagnosis after an error appears, and solving the new irreducible model at the steady state. If the time-averaged steady state probability of the states "Isolated" and "Reset" are, respectively, $\pi_I$ and $\pi_R$, then we can derive $P_{isol}$ as

$$P_{isol} = \frac{\pi_I}{\pi_I + \pi_R}. \tag{3}$$

Assuming that $\pi_{p,r}$ represents the time-averaged steady state probability of state $\langle p, r\rangle$, the flow equations for the Markov chain are given by

$$\pi_{1,0} = \pi_I + \pi_R, \tag{4}$$

$$\pi_{p,0} = (1 - d)\pi_{p-1,0} + \sum_{r=1}^{R-1} m(r)\pi_{p-1,r}, \tag{5}$$

$$\pi_I = (1 - d)\pi_{P-1,0} + \sum_{r=1}^{R-1} m(r)\pi_{P-1,r}, \tag{6}$$

$$\pi_{p,1} = d\pi_{p,0}, \tag{7}$$

$$\pi_{p,r} = (1 - m(r-1))\pi_{p,r-1}, \tag{8}$$

$$\pi_R = \sum_{p=1}^{P-1}(1 - m(R-1))\pi_{p,R-1}. \tag{9}$$

Unfolding (8), we have

$$\pi_{p,r} = \prod_{i=1}^{r-1}(1 - m(i))\pi_{p,1}.$$

From (7) and by substituting $\pi_{p,r}$ in (5), we have

$$\pi_{p,0} = \left(1 - d + d\sum_{r=1}^{R-1} m(r)\prod_{i=1}^{r-1}(1 - m(i))\right)\pi_{p-1,0}.$$

From the definition of $m(k)$ as the discrete hazard function of $f_{\widehat{Y}}(k)$, it follows that

$$1 - m(i) = \frac{1 - F_{\widehat{Y}}(i)}{1 - F_{\widehat{Y}}(i-1)},$$

$$\prod_{i=1}^{r-1} 1 - m(i) = 1 - F_{\widehat{Y}}(r-1),$$

$$m(r)\prod_{i=1}^{r-1} 1 - m(i) = f_{\widehat{Y}}(r).$$

Therefore,

$$\begin{aligned}\pi_{p,0} &= (1 - d + d\sum_{r=1}^{R-1} f_{\widehat{Y}}(r))\pi_{p-1,0},\\ &= (1 - d(1 - F_{\widehat{Y}}(R-1)))\pi_{p-1,0},\\ &= \left(1 - d\prod_{i=1}^{R-1}(1 - m(i))\right)\pi_{p-1,0}.\end{aligned}$$

Unfolding and from (6), we have

$$\pi_{p,0} = \left(1 - d\prod_{i=1}^{R-1}(1 - m(i))\right)^{(p-1)}\pi_{1,0},$$

$$\pi_I = \left(1 - d\prod_{i=1}^{R-1}(1 - m(i))\right)^{(P-1)}\pi_{1,0}.$$

Using $\pi_{1.0}$ from (4), we can derive $P_{isol}$ from (3):

$$\pi_I = \left(1 - d\prod_{i=1}^{R-1}(1 - m(i))\right)^{(P-1)}(\pi_I + \pi_R),$$

$$P_{isol} = \left(1 - d\prod_{i=1}^{R-1}(1 - m(i))\right)^{(P-1)}.$$

□

## ACKNOWLEDGMENTS

The research of Marco Serafini and Neeraj Suri has been supported in part by the European Commission Dependable Embedded Components and Systems (EC DECOS), Resilience for Survivability in IST (ReSIST), and Deutsche Forschungsgemeinschaft Technische Universität Darmstadt Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Systems (DFG TUD GKMM). The authors are grateful to the entire Dependable, Embedded Systems and Software (DEEDS) Group, TUD, for their inputs and help.

## REFERENCES

[1] P. Agrawal, "Fault Tolerance in Multiprocessor Systems without Dedicated Redundancy," *IEEE Trans. Computers,* vol. 37, no. 3, pp. 358-362, Mar. 1988.

[2] M. Barborak, M. Malek, and A. Dahbura, "The Consensus Problem in Fault-Tolerant Computing," *ACM Surveys,* vol. 25, no. 2, pp. 171-220, June 1993.

[3] K. Birman and T. Joseph, "Exploiting Virtual Synchrony in Distributed Systems," *Proc. 11th Symp. Operating Systems Principles (SOSP '87),* pp. 123-138, 1987.

[4] D.M. Blough and H.W. Brown, "The Broadcast Comparison Model for On-Line Fault Diagnosis in Multicomputer Systems: Theory and Implementation," *IEEE Trans. Computers,* vol. 48, no. 5, pp. 470-493, May 1999.

[5] M. Blount, "Probabilistic Treatment of Diagnosis in Digital Systems," *Proc. Seventh Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '77),* pp. 72-77, 1977.

[6] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Discriminating Fault Rate and Persistency to Improve Fault Treatment," *Proc. 27th Ann. Int'l Symp. Fault-Tolerant Computing Symp. (FTCS '97),* pp. 354-362, 1997.

[7] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Threshold-Based Mechanisms to Discriminate Transient from Intermittent Faults," *IEEE Trans. Computers,* vol. 49, no. 3, pp. 230-245, Mar. 2000.

[8] T. Chandra and S. Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems," *J. ACM,* vol. 43, no. 2, pp. 225-267, Mar. 1996.

[9] C. Constantinescu, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits," *Proc. IEEE Int'l Conf. Dependable Systems and Networks (DSN '02),* pp. 205-209, 2002.

[10] F. Cristian, "Reaching Agreement on Processor-Group Membership in Synchronous Distributed Systems," *Distributed Computing,* vol. 4, no. 4, pp. 175-187, Dec. 1991.

[11] F. Cristian and C. Fetzer, "The Timed Asynchronous Distributed System Model," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, no. 6, pp. 642-657, June 1999.

[12] D.D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J.M. Doyle, and W.H. Sanders, "The Möbius Framework and Its Implementation," *IEEE Trans. Software Eng.,* vol. 20, no. 10, pp. 956-969, Oct. 2002.

[13] L. Gong, P. Lincoln, and J. Rushby, "Byzantine Agreement with Authentication: Observations and Applications in Tolerating Hybrid and Link Faults," *Proc. Fifth Conf. Dependable Computing for Critical Applications (DCCA '95),* pp. 139-157, 1995.

[14] "Road Vehicles—Electrical Disturbances from Conduction and Coupling" *ISO 7637,* Int'l Organization for Standardization, 1997.

[15] R. Iyer, L.T. Young, and P.V.K. Iyer, "Automatic Recognition of Intermittent Failures: An Experimental Study of Field Data," *IEEE Trans. Computers,* vol. 39, no. 3, pp. 525-537, Apr. 1990.

[16] H. Kopetz and G. Grunsteidl, "TTP—A Protocol for Fault-Tolerant Real-Time Systems," *Computer,* vol. 27, no. 1, pp. 14-23, Jan. 1994.

[17] J. Kuhl and S. Reddy, "Fault Diagnosis in Fully Distributed Systems," *Proc. 11th Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '81),* pp. 100-105, 1981.

[18] J. Lala and L. Alger, "Hardware and Software Fault Tolerance: A Unified Architectural Approach," *Proc. 18th Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '88),* pp. 240-245, 1988.

[19] J.-C. Laprie, "Dependable Computing and Fault Tolerance: Concepts and Terminology," *Proc. 25th Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '95),* pp. 2-11, 1995.

[20] E. Latronico and P. Koopman, "Design Time Reliability Analysis of Distributed Fault Tolerance Algorithms," *Proc. IEEE Int'l Conf. Dependable Systems and Networks (DSN),* pp. 486-495, 2005.

[21] T. Lin and D. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," *IEEE Trans. Computers,* vol. 39, no. 4, pp. 419-432, Oct. 1990.

[22] P. Lincoln and J. Rushby, "A Formally Verified Algorithm for Interactive Consistency under a Hybrid Fault Model," *Proc. 23rd Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '93),* pp. 402-411, 1993.

[23] S. Mallela and G. Masson, "Diagnosis without Repair for Hybrid Fault Situations," *IEEE Trans. Computers,* vol. 29, no. 6, pp. 461-470, June 1980.

[24] M. Malek, "A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems," *Proc. Seventh Ann. Symp. Computer Architecture,* pp. 31-36, 1980.

[25] D. Powell, J. Arlat, L. Beus-Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabéjac, and A. Wellings, "GUARDS: A Generic Upgradable Architecture for Real-Time Dependable Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, no. 6, pp. 580-599, June 1999.

[26] F.P. Preparata, G. Metze, and R.T. Chien, "On the Connection Assignment Problem of Diagnosable Systems," *IEEE Trans. Electronic Computers,* vol. 16, no. 12, pp. 848-854, Dec. 1967.

[27] U. Schmid, "How to Model Link Failures: A Perception-Based Fault Model," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '95),* pp. 57-66, 1995.

[28] A. Sengupta and A. Dahbura, "On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach," *IEEE Trans. Computers,* vol. 41, no. 11, pp. 1386-1396, Nov. 1992.

[29] M. Serafini, N. Suri, J. Vinter, A. Ademaj, W. Brandstätter, F. Tagliabò, and J. Koch, "A Tunable Add-On Diagnostic Protocol for Time-Triggered Systems," *Proc. Int'l Conf. Dependable Systems and Networks (DSN '07),* pp. 164-174, 2007.

[30] K. Shin and P. Ramanathan, "Diagnosis of Processors with Byzantine Faults in a Distributed Computing System," *Proc. 17th Ann. Int'l Symp. Fault-Tolerant Computing (FTCS '87),* pp. 55-60, 1987.

[31] D.P. Siewiorek and R.R. Swarz, *Reliable Computer Systems: Design and Evaluation.* AK Peters, 1998.

[32] C. Walter, M.M. Hugue, and N. Suri, "Continual On-Line Diagnosis of Hybrid Faults," *Proc. Fourth Conf. Dependable Computing for Critical Applications (DCCA '94),* pp. 150-166, 1994.

[33] C. Walter, P. Lincoln, and N. Suri, "Formally Verified On-Line Diagnosis," *IEEE Trans. Software Eng.,* vol. 23, no. 11, pp. 684-721, Nov. 1997.

**Marco Serafini** received the Laurea degree (summa cum laude) from the University of Florence, Italy. He is currently working toward the PhD degree at the Technische Universität Darmstadt, Darmstadt, Germany. His research interests include dependable distributed systems, in particular online monitoring, fault location, and failure model characterization. He coordinates the diagnostic activities of the European Union Dependable Embedded Components and Systems (EU DECOS) Project. He is also currently collaborating with Hitachi on the design, verification, and experimental validation of safety-critical middleware for generic time-triggered systems. He is a student member of the IEEE and the IEEE Computer Society.

**Andrea Bondavalli** (M'96) is a professor in the Department of Systems and Informatics, University of Florence, Italy. Previously, he was a researcher at the Italian National Research Council, working at the CNUCE Institute, Pisa, where he was responsible for the Dependable Computing Group. He has been the principal investigator (PI) in many projects funded by the European Community, currently IST-2004-26979 HIDENETS and 2005-31413 SAFEDMI, acted in several occasions as an expert for the European community, and served as the program chair of the most important conferences in the area, including the 2005 Dependable Computing and Communications Symposium—The International Conference on Dependable Systems and Networks (DCC-DSN), the 19th IEEE Symposium on Reliable Distributed Systems (SRDS '00), the 2002 European Dependable Computing Conference (EDCC-4), the Third Latin-American Symposium on Dependable Computing (LADC '07). He is a member of the International Federation for Information Processing (IFIP) Working Group 10.4 on "Dependable Computing and Fault-Tolerance," European Network of Clubs for Reliability and Safety of Software intensive systems (ENCRESS) Club, Italy, and the AICA Working Group on Dependability in Computer Systems. His research activities and interests are the design and validation of critical systems and infrastructures, in particular the design of fault-tolerant architectures, mechanisms, and protocols, as well as their evaluation in terms of dependability attributes such as reliability, availability, and performability. He is the author of more than 110 refereed publications in international journals and conference proceedings. He is a member of the IEEE and the IEEE Computer Society.

**Neeraj Suri** received the PhD degree from the University of Massachusetts, Amherst. He is currently the chair professor of the Dependable Embedded Systems and Software Group, Technische Universität Darmstadt, Darmstadt, Germany. His earlier academic appointments include the Saab endowed professorship and faculty member at Boston University. He is a member of the International Federation for Information Processing (IFIP) Working Group 10.4 on Dependability and Microsoft's Trustworthy Computing Academic Advisory Board. He has served as the program committee (PC) chair for the IEEE Symposium on Reliable Distributed Systems (SRDS), the IEEE International Symposium on High Assurance Systems Engineering (HASE), the International Service Availability Symposium (ISAS), and Microsoft-TUD RAF and will serve as the PC Chair for the upcoming 2008 Dependable Computing and Communications Symposium—The International Conference on Dependable Systems and Networks (DCC-DSN). His research interests include the design, analysis, and assessment of dependable and secure systems, software, and services. His group's research activities have garnered support from the European Commission, the US Defense Advanced Research Projects Agency (DARPA), Deutsche Forschungsgemeinschaft (DFG), the US National Science Foundation (NSF), NASA, the Naval Air Warfare Center (NAWC), the US Office of Naval Research (ONR), Boeing, Microsoft, Intel, Saab, and Hitachi among others. He is an associate editor of the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Software Engineering*, the *ACM Computing Surveys*, and the *International Journal of Security and Networks* and has been an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He is also a recipient of an US NSF Faculty Early Career Development (CAREER) award. He is a senior member of the IEEE and the IEEE Computer Society.