A quantitative approach to run-time monitoring of Markovian concurrent systems

Laura Carnevali, Fulvio Patara, Alessandro Pinzuti, Enrico Vicario
Dipartimento di Sistemi e Informatica - Università di Firenze
{laura.carnevali, fulvio.patara, alessandro.pinzuti, enrico.vicario}@unifi.it

Abstract—Techniques for modeling and evaluation of timed concurrent systems can be effectively employed to support run-time monitoring of safety-critical systems. We address the issue by developing a predictive analysis engine that can guide a distributed sensor network to adaptively monitor a system that depends on a set of components, so as to guarantee reliable observation of critical system conditions while restraining the number of observation samples.

The monitored system is abstracted by a Fault Tree (FT) that represents the combinations of components conditions that lead to a critical system event. In turn, components are assumed to have recurrent behavior with exponential sojourn time in each operation mode and they are specified by Continuous Time Markov Chains (CTMCs). On the one hand, a predictive analysis engine based on the fault-tolerance architecture receives observations of components from the monitoring utility and maintains a dynamic measure of the transient probability that the overall system is in a critical condition. On the other hand, the monitoring utility receives from the analysis engine a measure of the contribution of each component to each predicted critical condition of the system. This enables the definition of sampling strategies that adapt the sample rate of nodes to the actual behavior of the supervised system. Effectiveness of adaptive sampling with respect to fixed-rate sampling is experimented in a setting where the analysis engine is integrated with a monitoring utility through an Enterprise Service Bus (ESB).

Index Terms—Concurrent timed systems, Continuous Time Markov Chain, Fault Tree Analysis, distributed sensor network, adaptive sampling.

I. INTRODUCTION

In the recent years, monitoring infrastructures have significantly spread into several areas including industrial, military, civil, domotic, and ambient applications. On the one hand, this is supported by emerging technologies that integrate information acquisition and transfer capabilities, such as Wireless Sensor Networks (WSNs), Radio Frequency IDentification (RFIDs) tags, and Smart Meters. On the other hand, this is motivated by the increasing demand to improve safety and security standards. To guarantee efficiency and effectiveness of monitoring, these infrastructures should integrate energy saving strategies that are able to prolong their lifetime and intelligent engines that steer system observation so as to accomplish reliability requirements in the application layer.

In the specific context of WSNs, various strategies have been proposed to reduce energy consumption through duty-cycling mechanisms. In particular, these perform topology control by acting on the selection of nodes which acquire and transfer sensor measurements and enforce power management by regulating the power-on timing of the selected nodes, even at the cost of a reduction of data quality and an increase of latency of data transfer [16], [30]. Energy efficiency strategies also include cross-layer adaptivity mechanisms which combine functionalities at different layers of the protocol stack. In particular, this often involves the application layer conditioning the actual operation of the network on the basis of achieved measurements and on forecasts on the monitored system. In [34], the node transfers to a manager node the model of the dynamics of the measured variable and does not transmit any data on the evolution of the measurement until the model adequately approximates the measurement. In [35], the sensor sampling period is adapted to the dynamics of the monitored system in order to create a feedback loop that keeps an error function within a predefined stability interval. An explicit reference to the reliability requirements of the application level is proposed in [33] to select routing paths through the network. Few experiences have been reported on schemes based on a discrete event model of the monitored system. In [7], a sensor network is integrated in a wearable device to monitor the vital parameters of a human; to limit the energy loss, an adaptive sensing strategy which traces the parameters in real-time against a Markov Decision Process is applied. In [6], TinyDB [35] queries to a network of nodes are approximated by taking into account the state of a model of the monitored system, represented as a Markovian process which evolves with time progress and measurements acquisition.

In the literature of reliability analysis, various models combine expressiveness and analytic capabilities that may support the development of policies able to adapt monitoring parameters to requirements in the application layer. In [36], leaf events of a Fault Tree with Repeated Events (FTRE) [25] can be associated with an exponential Cumulative Distribution Function (CDF), and the Top Event (TE) probability over time is derived by composition of exponential rates through approximate analysis. In [22], the time of occurrence of each leaf event is associated with a CDF and the FTRE is translated into a Generalized Stochastic Petri Net (GSPN) model [24]. Thus, under the assumption that all leaves have a negative exponential CDF, the evaluation of the FT is reduced to the analysis of a Continuous Time Markov Chain (CTMC). This also opens the way to extend modeling power with state space concepts that may account for dependencies and complex repair mechanisms. In [5], a parametric FT is translated into a Stochastic Well-Formed Colored Net in order to generate a lumped Markov Chain. In [9], a formal semantics of DFTs in
components are observed to be correct. However, the approach fails rejuve
nated through a conditioning whenever the a sensor network. To this end, CDFs of components time-to-
ness, since requires the enumeration of the portion of the state-space that is currently explored by the system. In [12], the model of Dynamic Reliability Block Diagrams (DRBDs) is used to support dependability evaluation of the multiprocessor computing system described in [26], [28], comparing the technique with the DFT approach. In [32], an analysis method is proposed to compare the effects of recurrent preventive maintenance operations on a critical system behaving as a CTMC.

Analysis techniques are implemented in various tools. Symbolic Hierarchical Automated Reliability and Performability Evaluator (SHARPE) [38] is a SW tool for specification and analysis of performance, reliability, and performability models. In particular, it provides a specification language and analysis algorithms for various stochastic models including FTs and FTREs, reliability graphs, series-parallel acyclic directed graphs, product-form queuing networks, Markov and semi-Markov chains, GSPNs, and Markov Regenerative Processes (MRGPs). Moreover, it provides a specification language for building single or hierarchical combinations of models where output measures of a model can be used as parameters of other models, facilitating the hierarchical combination of different model types. Galileo [14], [13] is an experimental SW tool for Dynamic Fault Tree Analysis (DFTA) which exploits a divide-and-conquer methodology. In particular, a DFT is modularized into statistically independent subtrees, which are then solved using either a dynamic or a static solver. The Object-based multi-formalism MOdeling of SYSTEMs (OsMoSys) framework [39] comprises a SW environment for the analysis of complex and heterogeneous systems specified by multi-formalism architectures based on various models including FTs, Petri Nets, and Process Algebras. Different algorithms can be applied to solve system submodels, supporting composition of results and reuse of models.

Efficiency and effectiveness of monitoring can be significantly improved when the infrastructure embeds an on-line engine that recurrently updates the expected system behavior according to samples of the actual operation mode of components [27], [11], [29], [17], [18]. In fact, an adaptive approach permits to attain a more accurate estimate on possible future behaviors and supports application to models with a huge state-space, since requires the enumeration of the portion of the state-space that is currently explored by the system. In [23], an on-line engine is employed to maintain a dynamic measure of the failure probability of a system monitored by a sensor network. To this end, CDFs of components time-to-failures are rejuvenated through a conditioning whenever the components are observed to be correct. However, the approach effectively fits only those scenarios where components time-to-failures are exponentially distributed. In fact, general CDFs are expected to exceed a given threshold within a finite time interval even if they are recurrently updated according to components observations, since each rejuvenation would increase their slope.

In this paper, we develop a predictive analysis engine that supports on-line monitoring of safety critical systems made by a set of components using a distributed sensor network. The monitored system is modeled through a hierarchical fault-tolerance architecture which specifies the behavior of single components through CTMCs and employs a FT to express the combinations of components operation modes that may lead to a critical system condition. The analysis engine recurrently updates the model according to samples delivered by the monitoring utility, deriving the transient probability function that the system is in a critical state and the contribution of each component to each predicted critical system condition. Due to the strict limits over energy resources in sensor nodes, adaptive sampling strategies are needed to prolong the lifetime of the monitoring utility. We thus show how the on-line predictive analysis engine supports definition and evaluation of a sampling strategy that adapts the sample rate of nodes to the actual behavior of the supervised system so as to restrain power consumption while maintaining a sufficient level of coverage over critical system conditions. The engine does not perform exhaustive state-space analysis but only enumerates the portion of the state-space that is currently being explored by the system, thus permitting to encompass also models with a huge state space.

The rest of the paper is organized as follows. Sect.II describes the hierarchical fault tolerance architecture combining CTMCs and FTs; Sect.III illustrates the behavior of the on-line predictive analysis engine in the reference scenario and proposes an adaptive sampling strategy; Sect.IV describes the tested implementation and shows preliminary experimental results comparing adaptive and fixed-rate sampling. Conclusions are finally drawn in Sect.V.

II. FAULT TOLERANCE ARCHITECTURE OF SAFETY-CRITICAL SYSTEMS

We address a safety-critical system made by a set of components which cooperate to accomplish some system goal. A component is associated with a set of operation modes characterized by a different level of criticality; some combinations of these operation modes result in a condition that is critical for the overall system or produce a system-level failure. We abstract the system into a hierarchical fault tolerance architecture that separately specifies the behavior of single components and their interactions using a two-level structured model made by:

- the model of each component, specifying how its working functionalities evolve over time;
- the model of the system, specifying how the operating conditions of components affect system behavior.
A. Component model

We consider a component operating across multiple modes classified as safe or critical. We assume that sensors are able to identify the present operation mode and that the sojourn time in each mode is exponentially distributed. According to this, a sample capturing the mode is sufficient to characterize the current state of the component.

Let \( X_k = \{ X_k(t), \ t \in \mathbb{R}^+ \} \) be the CTMC with finite state space \( S_k = \{ s_i^k, \ i = 1, \ldots, A_k \} \) that describes the behavior of component \( C_k \). Each state is associated with a different operation mode and accordingly regarded as a safe or critical state. In particular, we denote with \( S_k^{safe} = S_k \cap S_k^{crit} \) and \( S_k^{crit} = S_k \setminus S_k^{safe} \). Let \( Q_k = [q_{ik}^k] \) be the infinitesimal generator and \( \Pi_k(t) = [\pi_{ij}^k(t)] \) the conditional transient probability matrix:

\[
\pi_{ij}^k(t) = \text{Prob}\{X_k(t) = s_j^k | X_k(0) = s_i^k\}. \tag{1}
\]

According to Chapman-Kolmogorov’s equations:

\[
\Pi_k(t) = \Pi_k(t)Q_k, \tag{2}
\]

which yields:

\[
\Pi_k(t) = e^{Q_kt}. \tag{3}
\]

Marginalization of transient probabilities over the set of critical states provides the conditional transient probability \( \omega_i^k(t) \) that the component is in a critical state at time \( t \), given that the component was in state \( s_i^k \) at time \( 0 \):

\[
\omega_i^k(t) = \sum_{j \cdot s_j^k \in S_k^{crit}} \pi_{ij}^k(t). \tag{4}
\]

As an example, Fig.1 illustrates the concept assuming the case of a component \( C_1 \) behaving as the triple modular redundant system of [8], [19]. The system combines three processors and a voting system and is operational iff at least two processors and the voting system are correct. A failure of the voting system forces failure of all components. Failures occur in parallel with rate \( \lambda \); repairs are performed in series with rate \( \mu \); the voting system fails with rate \( \nu \) and is repaired with rate \( \delta \). Fig.2 reports transient probabilities \( \omega_i^1(t) \) referred to each state \( s_i^1 \) of the CTMC of Fig.1, assuming \( \lambda = 1, \mu = 2.5, \nu = 0.01 \), and \( \delta = 2.5 \).

![Fig. 1. The CTMC modeling the triple modular redundant system of [8], [19].](image-url)

B. System model

FTs are widely employed in the industrial practice [4], [1], [2], [3] to represent the hierarchical relationships among the operating conditions of components that can yield an undesired outcome usually called Top Event (TE). The combinations of events leading to the occurrence of the TE are expressed by combining them through boolean logic gates. Following a top-down approach, the step is iterated until the so-called leaf events are identified. In qualitative evaluation, the enumeration of the Minimal Cut Sets (MCSs) provides the minimal combinations of leaf events that lead to the occurrence of the TE. In quantitative evaluation, the combination of the occurrence probability of leaf events according to the architecture of the tree provides the occurrence probability of the TE, supporting reliability and safety analysis [31], [15]. In particular, this can be accomplished following either an indirect approach, which derives the TE probability by combining the probabilities of all the MCSs, or a direct approach, which repeatedly combines nodes probabilities at each gate of the tree [36].

In our formulation, each leaf event is associated with an occurrence probability over time, which represents the conditional transient probability that the corresponding component is in a critical state given that it was observed in a certain state at a previous time instant. Transient probability functions of leaf events are combined at each gate according to the architecture of the tree, following a direct bottom-up method [36], [22]. This prevents the representation of repeated events and dependencies in components failures, but reduces the computational effort and, as a by-product, provides the transient probability functions at intermediate gates of the tree.

We consider \( N \) leaf components \( C_1, C_2, \ldots, C_N \) and denote their transient probability function to be in a critical state with \( \omega_i^{1}(t), \omega_i^{2}(t), \ldots, \omega_i^{N}(t) \), respectively. In particular, \( \omega_i^{1}(t) = \omega_{i_1}^{1}(t) \forall \ i \in [1, N] \), where \( h_i \) is the index of the state \( s_{h_i}^k \) in which component \( C_i \) was observed by the last sample. As usual, the transient probability function of the \( AND, OR \), and \( KofN \) combination of events \{\( C_i \)\}_{i=1}^{N} \) are derived by combining the transient probability functions of leaf events through sums and products. Repeated application of the step
at each gate according to the architecture of the tree permits to derive the conditional transient probability \( \omega_{TE}(t) \) that the overall system is in a critical condition.

**III. AN APPLICATION SCENARIO**

We describe an application scenario where a predictive analysis engine based on the hierarchical fault-tolerance architecture of Sect.II can be integrated with a monitoring infrastructure. The engine receives observations of system components from the sensor nodes and maintains a dynamic measure of the conditional transient probability that the overall system is in a critical state (see Fig.3). More specifically:

- Sensor nodes periodically sample the operation point of system components and send time-stamped observations to a base station or sink node. This, in turn, delivers them to the predictive analysis engine through an Enterprise Service Bus (ESB). Each time-stamped observation is a triple \( (C_j, s_k^j, t_j) \) stating that component \( C_j \) was observed in state \( s_k^j \) at time \( t_j \).
- At the start-up, each component \( C_j \in \{ C_1, C_2, ..., C_N \} \) is in its initial state \( s_1^j \). Thus, the predictive analysis engine derives \( \omega_{TE}(t) \) by combining conditional transient probability functions \( \omega_1^j(t), \omega_2^j(t), \ldots, \omega_N^j(t) \).
- Given a set of observations \( O = \{ (C_j, s_k^j, t_j), j \in [1, N] \} \) arrived at time \( t_u \), the predictive analysis engine derives the new transient probability function of each component and then recomputes \( \omega_{TE}(t) \). In particular, as illustrated in Fig.4: i) the transient probability function of each component \( C_j \) that appears in an observation \( (C_j, s_k^j, t_j) \) is set equal to \( \omega_k^j(t) \) and shifted by the time elapsed between the component observation and the arrival of the measure to the analysis engine; ii) the transient probability function of each component \( C_j \) that does not appear in an observation is shifted by the time elapsed since the arrival of the last component measure to the analysis engine:

\[
\omega^j(t) = \begin{cases} 
\omega_k^j(t + t_u - t_j) & \text{if } (C_j, s_k^j, t_j) \in O, \\
\omega_k^j(t + t_u) & \text{otherwise.}
\end{cases}
\]  

(5)

- During the time between subsequent arrivals of observations, the analysis engine estimates the set of critical time intervals during which \( \omega_{TE}(t) \) exceeds a given threshold \( \delta \) within a given time-horizon \( [0, T] \) (see Fig.5):

\[
I = \{ I_n = [t_n, t_n + \Delta_n] \subseteq [0, T] \mid \omega_{TE}(t) \geq \delta \forall t \in [t_n, t_{n+1}], n \in \mathbb{N}, \delta \in [0, 1] \subset \mathbb{R}^+ \}.
\]  

(6)

The set \( I \) can be numerically derived by estimating the points where \( \omega_{TE}(t) \) intercepts the line \( y(t) = \delta \). The analysis engine then feeds back the monitoring utility with a quantitative measure of the contribution of each component.
component to the overall system criticality during each predicted time interval. This enables the adoption of sampling strategies that adapt the operation of the monitoring infrastructure to the actual behavior of the supervised system so as to save energy and ensure the observation of critical system conditions.

The approach does not perform exhaustive state-space enumeration but follows system evolution by enumerating only the portion of the state-space that is currently being explored by the system. This notably reduces the complexity of the problem and opens the way to application to models with a huge state space.

A. Rejuvenation policies

The predictive analysis engine can follow various policies to determine the times at which transient probability functions of components are updated and $\omega_{TE}(t)$ is consequently recomputed. In particular, in our scenario, the update is performed whenever a period of $P$ time units has elapsed since the last update and at least 1 observation has arrived.

B. Measures of Importance of system components

A Measure of Importance $G_k(t)$ of a component $C_k$ at time $t$ encodes the contribution of $C_k$ to the level of criticality of the overall system at time $t$. In particular, $G_k(t)$ should take into account: i) the conditional transient probability $\omega^k(t)$ that $C_k$ is in a critical state at time $t$; ii) the weight that $\omega^k(t)$ has on the global system probability to be in a critical state $\omega_{TE}(t)$ due to the static structure of the tree and the conditional transient probability $\omega^h(t)$ of any component $C_h$ that is topologically related to $C_k$. A large part of the literature about Reliability Engineering and Fault Tree Analysis focuses on a few indexes to capture the importance of single components on the operating condition of the overall system. Two significant measures are the Birnbaum measure and the Fussell-Vesely measure.

- The Birnbaum measure of a component $C_k$ at time $t$ evaluates the difference between the probability that the system is in a critical condition at time $t$ given that $C_k$ is in a critical state at time $t$, i.e., $\omega^k(t) = 1$, and the probability that the system is in a critical condition at time $t$ given that $C_k$ is correct at time $t$, i.e., $\omega^k(t) = 0$:

$$G_k^B(t) = \omega_{TE}(t)|_{\omega^k(t)=1} - \omega_{TE}(t)|_{\omega^k(t)=0} \quad (7)$$

The measure of Birnbaum evaluates the impact of the operation mode of a single component on the behavior of the overall system, but it does not adequately take into account the topology of the fault tolerance architecture.

- The Fussell-Vesely measure of a component $C_k$ at time $t$ is the sum of probabilities of MCSs that include $C_k$:

$$G_k^{FV}(t) = \sum_{CS_j \in MCS(C_k)} \text{Prob}\{CS_j \text{ occurs at time } t\}, \quad (8)$$

where $MCS(C_k)$ is the set of MCSs that contain component $C_k$, which can be derived through the MOCUS algorithm (Method of Obtaining Cut Sets). The measure of Fussell-Vesely takes into account both the probability that single components are in a critical state and the topology of the tree. However, any two components that are part of the same MCSs turn out to have the same Fussell-Vesely measure, regardless of their own probability to be in a critical state.

To overcome the problem, we adopt a variant of the Fussell-Vesely measure that takes into account also the probability that the component is in a critical state at time $t$:

$$G_k(t) = \omega^k(t) \cdot \sum_{CS_j \in MCS(C_k)} \text{Prob}\{CS_j \text{ occurs at time } t\} \quad (9)$$

This measure combines two factors: on the one hand, $\omega^k(t)$ accounts for the evolution of the working functionalities of the component after $t$ time units; on the other hand, $\sum_{CS_j \in MCS(C_k)} \text{Prob}\{CS_j \text{ occurs at time } t\}$ provides a quantitative measure of how the operation mode of the component at time $t$ weights on the state of the overall system according to the topology of the fault tolerance architecture.

C. An adaptive sampling strategy

The monitoring utility receives from the analysis engine the following quantitative information about the operating conditions of the supervised system: i) the set of predicted critical time intervals; ii) the probability that each component is in a critical state evaluated at the beginning of each predicted critical time interval; iii) a measure of importance of each component evaluated at the beginning of each predicted critical time interval. Based on this information, the monitoring utility adapts the sampling rate of nodes to the actual operating conditions of the supervised system in order to guarantee early observation of critical system conditions. Various strategies can be defined. In particular, we consider the case where:

- at the start-up, all nodes have the same sample rate $R$;
- whenever the importance measure $G_k(t)$ of a component $C_k$ exceeds a given threshold $\epsilon \in [0, 1] \subseteq \mathbb{R}^+_0$ at the beginning of a predicted critical time interval $I_n$, the sample rate of the corresponding node is set equal to $H \cdot R$;
- when the interval $I_n$ has elapsed, the sample rate of the node is reset to $R$.\n
![Fig. 5. Critical time intervals within a time-horizon $[0, T]$.](image)
IV. EXPERIMENTATION

We performed a preliminary experimentation to prove feasibility and effectiveness of the approach. The adaptive sampling strategy was compared with a fixed-rate sampling strategy in a setting where system components fail with controlled failure times generated according to exponential distributions with known rate. This permits to derive a ground truth function that encodes the time intervals during which the system is safe or critical. Comparison of the ground truth with $\omega_{TE}(t)$ yields the number of correctly detected TEs, providing a measure of the effectiveness of monitoring. The rate of messages sent by sensor nodes is assumed as an estimate of the effort spent in the monitoring.

The experiment is performed in an emulated environment where the predictive analysis engine is integrated with a scaffold monitoring utility through an ESB based on the Apache Camel integration framework [21].

A. Derivation of the ground truth function of the TE

The SW module that emulates the sensor network is realized as a message broker that sends system observations to the analysis engine through the ESB. This module is made of a sub-module for each component of the supervised system, plus a sub-module for the sink node. A sub-module that emulates the behavior of a component stores a feasible behavior of that component, i.e., a behavior consistent with the CTMC that describes the transitions between its operation modes. The sub-module that emulates the sink node collects the observations sent by the other submodules and delivers them to the analysis engine through the ESB. Since behaviors stored by sub-modules that emulate a system components are known, the times at which component perform operation mode transitions are known. This permits to derive a ground truth function $g_{TE}(t)$ for each component $C_k$, which specifies the time intervals during which $C_k$ is actually in a safe or in a critical state:

$$g_{k}(t) = \begin{cases} 0 & \text{if } C_k \text{ is in a safe state at time } t, \\ 1 & \text{otherwise.} \end{cases} \quad (10)$$

Combination of ground truth functions of system components according to the structure of the fault-tolerance architecture enables the derivation of the ground truth function of the TE, which specifies the time intervals during which the system is actually in a safe or a critical condition, i.e., the time interval during which a TE occurs:

$$g_{TE}(t) = \begin{cases} 0 & \text{if the system is in a safe state at time } t, \\ 1 & \text{otherwise.} \end{cases} \quad (11)$$

The identification of critical time intervals after each rejuvenation permits to derive a function $z_{TE}(t)$ which specifies the time intervals during which a TE is predicted:

$$z_{TE}(t) = \begin{cases} 0 & \text{if } \omega_{TE}(t) < \delta, \\ 1 & \text{otherwise.} \end{cases} \quad (12)$$

As illustrated in Fig. 6, direct comparison of $z_{TE}(t)$ and $g_{TE}(t)$ permits to derive the number of TEs that have correctly been detected by the monitoring infrastructure, the latency that affects the detection of each observed TE, and the number of false positive/negative detected TEs.

B. Evaluation of the adaptive sampling strategy

We assess cost and effectiveness of the adaptive sampling strategy using different metrics:

- the cost is evaluated through the rate of messages sent by the nodes, which is equal to the rate of sampling operations and can be roughly assumed to be proportional to the energy spent by the nodes;
- the effectiveness is estimated through the number of TEs correctly predicted by the analysis engine, which is the main objective of the monitoring system.

We consider a safety-critical system made by 8 components behaving as the three-state CTMC of Fig.7, with $\lambda$ equal to 1 and $\mu$ equal to 2.5: a component includes two processors and is operational iff at least one of them is correct; failures occur in parallel with rate $\lambda$, while repairs are performed in series with rate $\mu$. Components are organized in a fault tolerance architecture according to the binary FT of depth 3 shown in
Fig. 8. All experiments were performed on an Intel Core 2 Quad Q6600 desktop processor for a time interval of 1000 seconds, i.e., nearly 15 minutes. The ground truth function $g_{TE}(t)$ of the system includes 198 TEs.

![CTMC diagram](image)

Fig. 7. A three-state CTMC modeling a component with two processors that is operational iff at least one of them is correct.

We compare the effectiveness of the fixed-rate policy and the adaptive-rate policy assuming $\delta$ equal to 0.01, $\epsilon$ equal to 0.002, and rejuvenation policy $A.4$ with period $P$ equal to 250 ms. Table I illustrates the rate of messages sent by sensor nodes and the percentage of detected TEs. Results show that, for the same value of the rate $R$, the proposed adaptive-rate policy always detects a higher number of TEs than the fixed-rate policy at the cost of a higher rate of exchanged messages. For instance, when the rate $R$ is increased from $1/8$ to 1, the rate of exchanged messages increases from 66.7 to 533.3 msg/min for the adaptive-rate policy and from 89.3 to 318.5 msg/min for the adaptive-rate policy with $H$ equal to 2, raising the percentage of detected TEs from 10.10% to 98.48% for the fixed-rate policy and from 19.70% to 100.00% for the adaptive-rate policy. In a similar manner, for the same value of the rate $R$ of the adaptive-rate policy, the higher is the parameter $H$, the higher is the rate of exchanged messages and the percentage of detected TEs. Also note that the rate of messages that is necessary to attain full TEs coverage is equal to 1066.7, 637.1, and 843 for the fixed-rate policy, the adaptive-rate policy with $H$ equal to 4, and the adaptive-rate policy with $H$ equal to 2, respectively.

<table>
<thead>
<tr>
<th>policy</th>
<th>$R$ (1/sec)</th>
<th>$H$</th>
<th>msg rate (# msg/min)</th>
<th>% detected TEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed rate</td>
<td>$1/8$</td>
<td>-</td>
<td>66.7</td>
<td>10.10</td>
</tr>
<tr>
<td></td>
<td>$1/4$</td>
<td>-</td>
<td>133.3</td>
<td>20.71</td>
</tr>
<tr>
<td></td>
<td>$1/2$</td>
<td>-</td>
<td>206.7</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>533.3</td>
<td>98.48</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td>1066.7</td>
<td>100.00</td>
</tr>
<tr>
<td>adaptive rate</td>
<td>$1/8$</td>
<td>2</td>
<td>89.3</td>
<td>19.70</td>
</tr>
<tr>
<td></td>
<td>$1/4$</td>
<td>2</td>
<td>194.4</td>
<td>41.41</td>
</tr>
<tr>
<td></td>
<td>$1/2$</td>
<td>3</td>
<td>318.5</td>
<td>80.81</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>318.5</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>637.1</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>109.3</td>
<td>33.84</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>224.3</td>
<td>68.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>414.6</td>
<td>81.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>843</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

TABLE I

Rate of exchanged messages and percentage of detected TEs for fixed-rate and adaptive sampling strategies, assuming $\delta = 0.01$, $\epsilon = 0.002$, and rejuvenation period $P = 250$ ms.

V. Conclusions

This paper proposes a quantitative approach to run-time monitoring of systems that are made by a set of distributed components, supporting the evaluation of cross-layer adaptive sampling strategies based on reliability requirements in the application layer. In the reference scenario, a safety-critical system is monitored by a sensor network in order to guarantee the observation of critical system conditions. The system is specified by a hierarchical architecture made by CTMCs modeling the behavior of single components and a FT representing the relationships among the operation modes of components that may lead to a critical system condition. The operation point of system components is recurrently sampled by sensor nodes and the observations are delivered to a predictive analysis engine, which rejuvenates the model of the system and maintains a dynamic measure of the transient probability that the system reaches a critical condition. This enables the definition of cross-layer energy efficiency strategies that adapt the sample rate of nodes to the current system operation according to reliability requirements in the application layer. The capability to operate on-line permits to maintain a more accurate estimate on system behavior, since the expectation is recurrently updated according to the actual evolution followed by the system. Moreover, the approach applies well to models with a huge state space, since exhaustive state-space enumeration is not necessary and the approach resorts to the enumeration of the portion of the state-space that is currently being explored by the system.

Cost and effectiveness of an adaptive sampling strategy are evaluated through different metrics that take into account the rate of messages sent by nodes and the number of correctly detected TEs, respectively. An experimental testbed has been implemented where the sensor network is emulated by a scaffold component integrated with the predictive analysis engine through an ESB. Preliminary results show that, for the same value of the sample rate $R$, the proposed adaptive-rate policy always detects a higher percentage of TEs than the fixed-rate policy at the cost of a higher rate of exchanged messages. Moreover, the adaptive-rate policy attains full cov-
erage of TEs with a significantly lower rate of messages than the fixed-rate policy. Future work includes the definition of more accurate adaptive sampling strategies that take into account also the latency in the detection of TEs, the number of false positive/negative predicted TEs, and the sojourn time of system components in individual states, so as to increase the sample rate when a component is in a state with small sojourn time and high probability to reach a critical state. Furthermore, the approach could adapted to fit the specific needs of different application contexts that require monitoring capabilities with recurrent rejuvenations, such as a cloud computing system where multiple resources are arranged in a virtualized infrastructure to supply distributed processing capabilities.

The proposed approach also applies to those scenarios where sojourn times of system components in operation modes are generally distributed, which can be effectively managed through transient analysis based on stochastic state classes [20]. However, in such a setting, the practical implementation of the monitoring system would face a major hurdle: a sample capturing the mode would not be sufficient to characterize the state of the monitored component, which also depends on the time elapsed since the most recent mode transition. Various approaches can be devised to overcome the limit, including the usage of sensors that can reconstruct the time elapsed when the mode was entered, or the usage of a frequent sampling that guarantees the observation of mode transitions within a negligible latency.

REFERENCES