Journal of Network and Systems Management (2024) 32:95
https://doi.org/10.1007/s10922-024-09868-y

Availability and Performance Assessment of IoMT Systems: A Stochastic Modeling Approach

Thiago Valentim[1] · **Gustavo Callou**[2] · **Cleunio França**[1] · **Eduardo Tavares**[1]

Received: 10 May 2024 / Revised: 3 August 2024 / Accepted: 30 August 2024

Abstract
Internet of Things (IoT) allows distinct elements of an environment to be remotely monitored using existing network infrastructures, creating a prominent integration of disparate computing systems. Such an integration commonly results in efficient data collection and processing. Indeed, the adoption of IoT can improve communication in gathering and transmitting data, especially in locations that deal with connectivity challenges. For instance, hospitals have adopted IoT to collect and transmit patient data to health professionals, as critical patients must be monitored uninterruptedly. As a consequence, healthcare systems typically require high availability, in which connectivity is essential for critical medical decisions. Some works have conceived techniques to assess the availability of Internet of Medical Things (IoMT) systems, but the joint assessment of performance and availability is generally neglected. This paper presents a modeling approach based on stochastic Petri nets (SPN) and reliability block diagrams (RBD) to evaluate IoMT systems. The proposed technique evaluates availability and response time of the communication between devices in an IoMT architecture. Experimental results show the practical feasibility of the proposed approach, in which a sensitivity analysis is adopted to indicate the components with the most significant impact on the system operation. Our approach contributes to the state of the art as an additional technique to evaluate different system designs before modifying or implementing the real system or a prototype.

Keywords Analytical models · Availability · Performance evaluation · IoMT

1 Introduction

Internet of Things (IoT) is a technological framework that has fostered progress in several aspects of human society. This paradigm has been applied to weather prediction systems, drones, waste management, intelligent agriculture, and smart hospitals [1, 2].

Extended author information available on the last page of the article

Published online: 13 September 2024

The growing interest in IoT devices has directly impacted investments in this field. Current projections indicate the IoT market will reach US$ 771 billion dollars in 2026, and, by 2030, there will be 500 billion objects connected to the Internet [3]. The adoption of IoT can improve communication in collecting and transmitting data, especially in locations that face connectivity challenges. For instance, hospitals have adopted IoT to collect and transmit patient data to health professionals, as critical patients must be monitored uninterruptedly [4].

Particularly, global investment in IoT-based medical systems reached US$217.34 billion in 2022, and it is expected to be around US$960.2 billion by 2030. The huge investment also concerns acquiring sensors and computing devices for remote patient monitoring (RPM) [5]. Consequently, the term Internet of Medical Things (IoMT) has been coined to highlight the importance of IoT in medical applications. IoMT systems may be critical because system failures affect patient lives. Therefore, over the years, research has been carried out to conceive techniques to improve availability in IoMT applications [6]. Besides, the need for real-time data sharing, security and confidentiality of patient data highlights the importance of private clouds.

Availability evaluation of IoMT systems is essential due to the critical operation of healthcare services, demanding the readiness of such systems. Indeed, unavailability can result in delays in patient monitoring and inaccurate diagnoses, which could result in risks to patient life [7]. The integrated availability and performance assessment, namely, performability, is fundamental for the analysis of IoMT systems. Adopting stochastic models, the unified evaluation allows non-functional requirements to be analyzed using distinct system configurations before implementing the real system [8]. Prominent works [9–13] have evaluate availability and performance of IoMT systems, but the joint evaluation is usually neglected.

This paper presents a modeling approach based on reliability block diagrams (RBD) and stochastic Petri nets (SPN) to evaluate the availability and performance of IoMT systems. Our proposed SPN model assumes an IoMT system in a private cloud, considering software and hardware components that compose the architecture (e.g., virtual machine, microcontroller, and sensor). Our proposal analyzes the interaction between IoMT devices to jointly estimate response time and system availability. Experimental results show the practical feasibility of the proposed approach, in which a sensitivity analysis is adopted to indicate the components with the most significant impact on system operation.

The remainder of this paper is organized as follows. Section 2 presents related work, and Sect. 3 introduces prominent concepts for a better understanding of this work. Section 4 describes the adopted methodology. Section 5 presents the adopted IoMT architecture, and Sect. 6 describes performability and availability models for assessing IoMT systems. Section 7 presents experimental results. Finally, Sect. 8 concludes this work.

2 Related Work

Over the last years, some researches have been carried out to evaluate availability and performance of IoMT systems. Availability is often overlooked, but it is a prominent attribute to ensure system operation.

For the sake of readability, this section is divided into two subjects: (i) availability and (ii) performance.

2.1 Availability

The distinct demands for IoT systems have created challenges to availability evaluation, as those systems may require continuous data gathering and transmission.

Shen et al. [14] propose an availability assessment mechanism for IoT systems utilizing edge computing. The approach adopts a modeling technique based on Markov matrix to represent malware infection. Rahmani et al. [15] present a failure detection method for IoMT systems using neural network. Whenever a failure occurs, the conceived mechanism searches for operational servers and moves them to the cluster to restore the system service. Despite the importance of such a work, a sensitivity analysis is not utilized, which could indicate prominent system parameters. Santos et al. [16] detail a multi-objective optimization algorithm adopting stochastic models to analyze the influence of failures on an e-health system.

Nguyen et al. [17] propose a hierarchical modeling approach to assess IoMT infrastructures. The approach adopts fault trees and Markov chains, focusing on availability and security issues. The approach evaluates an IoMT architecture composed of cloud and edge computing components. Santos et al. [18] propose an approach based on SPN to assess IoT systems based on cloud computing, focusing on system throughput. Clemente et al. [19] presents a hierarchical modeling approach to evaluate the availability of private cloud systems, taking into account connectivity, virtualization, and storage components. Results indicate the redundancy strategies that improve availability.

Andrade et al. [20] propose a SPN approach to evaluate IoT systems based on fog and cloud computing. The metrics of interest are response time and availability. Results indicate that fog computing components significantly influence availability, but nothing is stated about IoMT architectures. Lima et al. [21] describe a SPN-based technique to carry out a disaster recovery analysis of IoMT systems. The proposed model aims to identify the components with the highest influence on system restoration.

2.2 Performance

Mallick et al. [22] propose a blockchain framework to reduce network bandwidth usage and processing time in IoMT systems adopting fog computing. In that approach, fog computing moves data processing to the edge of the network, minimizing the load on central servers. In addition, blockchain provides a reliable

method for recording transactions. A case study utilizes simulation to assess the framework performance. El Kafhali et al. [23] present a Markov chain model to assess the required number of fog nodes to deal with the workload of IoT devices. The focus is on the improvement of response time and resource utilization. For validation, the authors adopted a simulation tool to compare the results.

Djahafi et al. [24] propose a formal approach using stochastic Petri nets for analyzing IoT systems, taking into account publish/subscribe communication and response time. Experiments indicate the communication broker has a prominent influence on system performance. Ghosh et al. [25] present an IoMT architecture for constrained network bandwidth. A monitoring system based on machine learning is described, which redirects data processing in the fog when the patient is in critical condition. Experiments demonstrate improvements in latency for monitoring of patients requiring close attention.

Nakayama et al. [26] propose a communication architecture for IoMT systems in order to ensure service continuity for patients requiring transport to another facility. The architecture focuses on increasing communication reliability and meeting performance constraints. In [27], the authors present an analytical approach to assess the influence of load balancing on the performance of medical information systems. Stochastic reward nets are adopted.

In [28], the authors detail a fog-based IoT architecture for monitoring patient health. The work focuses on reducing response time using Bayesian networks. The technique adopts a measurement approach to analyze system performance, comparing the proposed technique with a similar approach using cloud computing. Ezhilarasi et al. [29] describe a framework for healthcare services that incorporates three layers: IoT, fog computing, and cloud storage. The framework aims to improve resource utilization.

Unlike previous works, this paper proposes a modeling approach based on SPN and RBD to jointly evaluate performance and availability of IoMT systems using a private cloud. Table 1 depicts an overview of related work compared to our technique, taking into account the adopted strategy and evaluated metrics. The proposed approach provides models that can be adopted as an additional tool by system designers to assess distinct system configurations regarding availability and performability even before implementing the actual infrastructure.

3 Background

This section introduces essential concepts to better understand this work.

3.1 IoT Architecture

IoT allows distinct elements of an environment to be remotely monitored using existing network infrastructures, creating a prominent integration of disparate computing systems. Such an integration commonly results in efficient data collection and processing [30].

Table 1 Related work

Work	Strategy	Metrics
Shen et al. [14]	Markov Matrix	Availability
Rahmani et al [15]	Neural network	Availability
Santos et al. [16]	SPN	Availability
Nguyen et al. [17]	Markov chains	Availability
Santos et al. [18]	SPN	Availability, throughput
Clemente et al. [19]	SPN	Availability
Andrade et al. [20]	SPN	Availability, response time
Lima et al. [21]	SPN	Availability
Mallick et al. [22]	Simulation	Bandwidth, data traffic
El Kafhali et al. [23]	Markov chain	Response time, loss rate, throughput
Djahafi et al. [24]	SPN	Response time
Ghosh et al. [25]	Machine Learning	Latency, energy
Nakayama et al. [26]	Simulation using Mininet	Throughput, bandwidth
Nguyen et al. [27]	SRN	Response time, throughput, discard probability
Gupta et al. [28]	Measurements	Response time
Ezhilarasi et al. [29]	Measurements	Response time
This work	Measurements, SPN and RBD	Availability, downtime and response time

A basic IoT architecture [31] is divided into four layers (Fig. 1): devices, communication, processing and presentation. Devices perform data gathering and contemplate, for instance, sensors and microcontrollers. The communication layer carries out data transfer using standard protocols, such as LoRA, MQTT, and ZigBee, for further processing [32, 33]. The processing layer manipulates data to execute system services, and the presentation layer provides mechanisms for end-user interaction.

3.2 Availability

IoMT systems usually deal with critical data, and, thus, high availability is an important attribute for those systems [34]. For instance, an equipment failure cannot cause

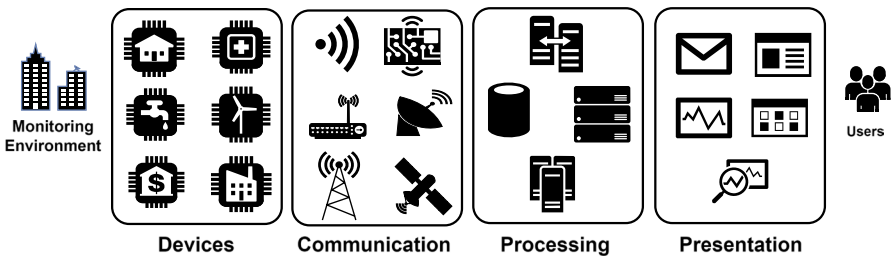

Fig. 1 IoT basic architecture

severe consequences for a patient (e.g., a monitoring device failure cannot lead to a false alarm).

Availability is the probability of a system being in a functioning state [8]. Steady-state availability (A) is commonly utilized, which takes into account the relationship between the system's mean time to failure (*MTTF*) and mean time to repair (*MTTR*) [8]:

$$A = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

$$MTTF = \int_0^\infty R(t)dt \tag{2}$$

$$MTTR = \int_0^\infty 1 - M(t)dt \tag{3}$$

$M(t)$ is the cumulative distribution function representing the probability that a repair will occur within time t. $R(t)$ is the reliability function, which is the probability of a system performing its functions without failures for period t.

3.3 Petri Nets

Petri nets (PN) are a graphical and mathematical modeling tool that can be adopted to represent several system types. For instance, parallelism, concurrency, asynchronous, and non-deterministic activities are naturally expressed in a PN model [35].

A Petri net is a bipartite directed graph in which places denote local states and transitions represent actions. Arcs connect places to transitions and vice versa. Tokens may reside in places denoting a state (e.g., marking). An inhibitor arc represents the unavailability of tokens in places, and the semantics of a PN is defined in terms of a token game (e.g., tokens are generated and consumed due to the firing of transitions).

This work adopts a specific PN extension, namely, generalized stochastic Petri nets (GSPN), which allows the addition of probabilistic delays to timed transitions or zero delays (and guard expressions) to immediate transitions. Stochastic Petri nets (SPN) are commonly adopted to denote all stochastic extensions of PN formalism, and for the sake of simplicity, the term SPN is utilized henceforth.

The state space of an SPN model can be translated into a continuous-time Markov chain (CTMC), and simulation techniques can also be adopted as an alternative to the generation of CTMCs [8].

As an example, Fig. 2 presents a model with a physical machine (left) and a virtual machine (right). A token in place *HostUp* (*VMUp*) indicates the physical device (VM) is operational, and a token in place *HostDown* (*VMDown*) denotes the device is unavailable. The firing of transition *tHostFail* (*tVMFail*) consumes a token from place *HostUp* (*VMUp*) and generates a token in place *HostDown* (*VMDown*), representing the device inoperability. Similarly, the firing

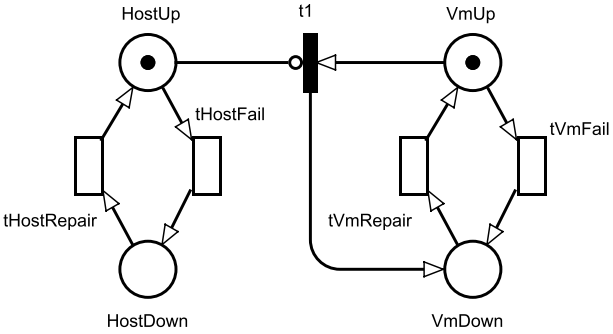

Fig. 2 GSPN example

of transition *tHostRepair* (*tVMRepair*) denotes the maintenance (recovery) of a device. In case of physical machine failure, the VM is also not operational. More specifically, immediate transition $t1$ is enabled due to the inhibitor arc, and the respective firing represents the VM failure (i.e., token generated in place *VMDown*).

As usually adopted in SPN, operator # represents the number of tokens in a place (e.g. #HostUp), and $P\{exp\}$ indicates the probability of inner expression *exp*. These operators are utilized in Sect. 6.

3.4 Reliability Block Diagram

A reliability block diagram (RBD) is a combinatorial modeling formalism that represents conditions that make a system operational regarding the structural relationship between the system components [8]. RBD assumes that each component failure (and maintenance) is independent of other counterparts. This formalism can be graphically represented by a set of blocks (rectangles), in which each block denotes a system component (device). Arcs connect the blocks, and the system is operational if there is at least one path from the beginning node to the end node. In general, the components can be configured in series or parallel. Figure 3 illustrates two examples in which the blocks are arranged in series (Fig. 3a) and in parallel (Fig. 3b).

In the serial arrangement, the system becomes non-operational if just one component fails. For a system with n independent components, reliability or steady-state availability (P_s)is calculated as follows:

$$P_s = \prod_{i=1}^{n} P_i \tag{4}$$

in which P_i is the reliability or steady-state availability of a component i.

Regarding parallel arrangement, the system remains operational as long as at least one of the components is operational. Considering a system with n independent components, reliability or steady-state availability (P_s) is calculated as follows:

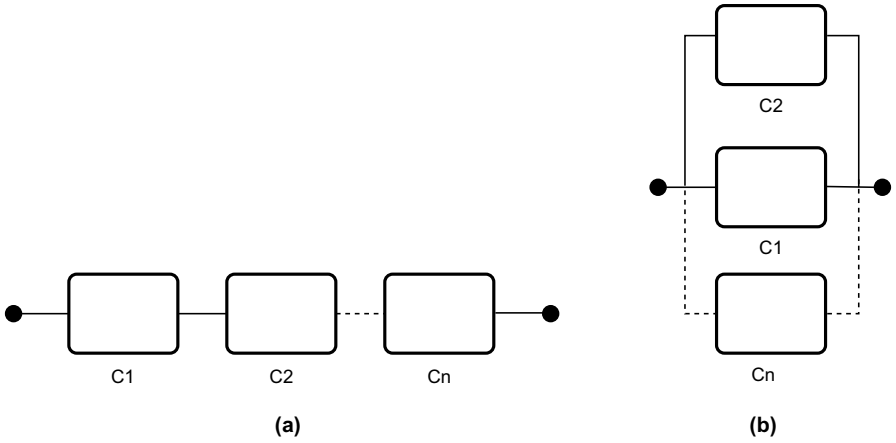

Fig. 3 RBD arrangements. **a** Series, **b** parallel

$$P_s(t) = 1 - \prod_{i=1}^{n} P_i\left(1 - P_i\right) \tag{5}$$

in which P_i is the reliability or steady-state availability of a component i.

4 Methodology

This section presents the adopted methodology (Fig. 4) to evaluate availability and performance of IoMT architectures using a private cloud. The first step concerns system understanding, in which the designer indicates the system components (and the respective interactions). Next, metrics are defined (e.g., availability) in order to quantify system behavior.

Data gathering carries out an initial measurement of system activities (on a prototype or real system). Whenever a measurement is not feasible (e.g., a system in production), the designer may adopt values, for instance, from datasheets.

The fourth step performs model creation, which also takes into account component interactions and the defined metrics. Assuming independent components and availability, reliability block diagrams (RBD) are adopted, as the computational cost to evaluate the system is shorter due to closed-form equations. Regarding complex interactions or performance metrics, a state-based representation, more specifically, stochastic Petri nets, is utilized to represent the system.

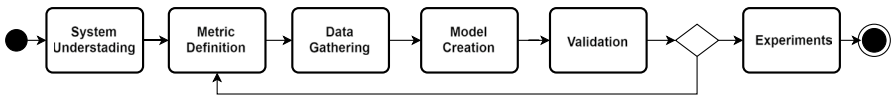

Fig. 4 Methodology

Afterward, model validation is adopted to verify whether the model produces estimates close to a target system. This step adopts statistical methods to ensure the model results are equivalent to the data measured in the real system. If the model does not provide accurate estimates, the designer must review the conceived model.

Finally, the models are adopted to assess distinct system designs. A design may adopt a sensitive analysis (e.g., an effects analysis) to identify the components that have a greater influence on availability or performance.

5 IoMT Architecture

An IoMT architecture defines a smart environment that contemplates electronic devices and sensors to monitor physiological signals from patients that may or may not be hospitalized [36].

The architecture should allow physicians to access and analyze patient data in real-time, making better-informed decisions about patient care. The respective system usually includes communication protocols, data storage, data analytics, visualization tools, and other hardware and software components to enable medical personnel to monitor and remotely manage patient health. Additionally, the architecture can collect data from various sources, such as wearable and mobile devices, providing a comprehensive view of patient status [37].

Figure 5 represents the IoMT architecture adopted in this work, which is based on [38] and has four layers. The sensor layer is composed of sensors responsible for collecting patients' physiological data. The data gathering layer is composed of devices that transmit data provided by sensors using the public network (i.e., the Internet). The private cloud layer is responsible for storing patient data for future assessment. Such a layer deals with the cloud computing infrastructure, which includes the adopted virtual machines. A private cloud is required, as patient data are sensitive and must be managed and securely stored by health facilities.

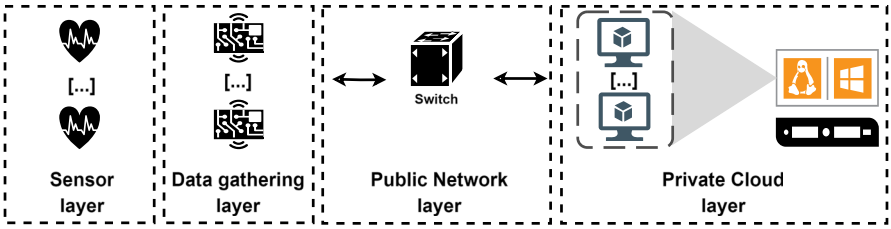

Fig. 5 IoMT basic architecture

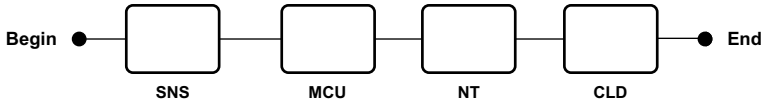

Fig. 6 RBD model: system

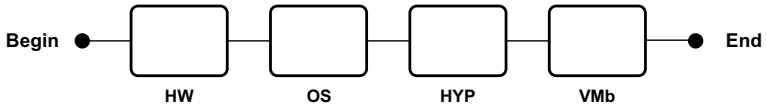

Fig. 7 RBD models private cloud layer

6 Availability and Performability Models

This section presents the proposed availability and performability models. For this work, reliability block diagrams (RBD) are utilized to estimate system availability and downtime. SPN models are adopted to estimate response time under the influence of each layer availability.

6.1 Availability Model

The conceived availability model is based on reliability block diagrams (RBD), which represent an IoTM system adopting the architecture described in Sect. 5. In this work, the system is operational if all components are working. In other words, a single component failure makes the system non-operational. Figure 6 depicts the RBD model, taking into account the sensor layer (SNS), data gathering layer (MCU), public network layer (NT), and private cloud layer (CLD).

A single component is adopted for the SNS, MCU, and NT. For the latter, a network switch is assumed. For the sake of readability, CLD block is an abstraction to another RBD model (Fig. 7), which is composed of hardware (HW), an operating system (OS), a hypervisor (HYP), and a virtual machine (VMb). Such a model can also be reduced to a single block with an approximate MTTF and MTTR [39].

6.2 Performability Model

This section presents the conceived model (Fig. 8) for the unified evaluation of performance and availability assessment. The metric of interest is response time, which contemplates the delay for the private cloud layer in receiving data from the sensor layer. Availability may significantly influence response time, as system components can fail.

For a better understanding, the model is divided into blocks. Taking into account the adopted architecture (Sect. 5) and RBD model, each layer has an availability block (in red) that abstracts the adopted components. Place *XUp* (e.g., PsUp) indicates the layer (or component) is operational, whereas place *XDown*

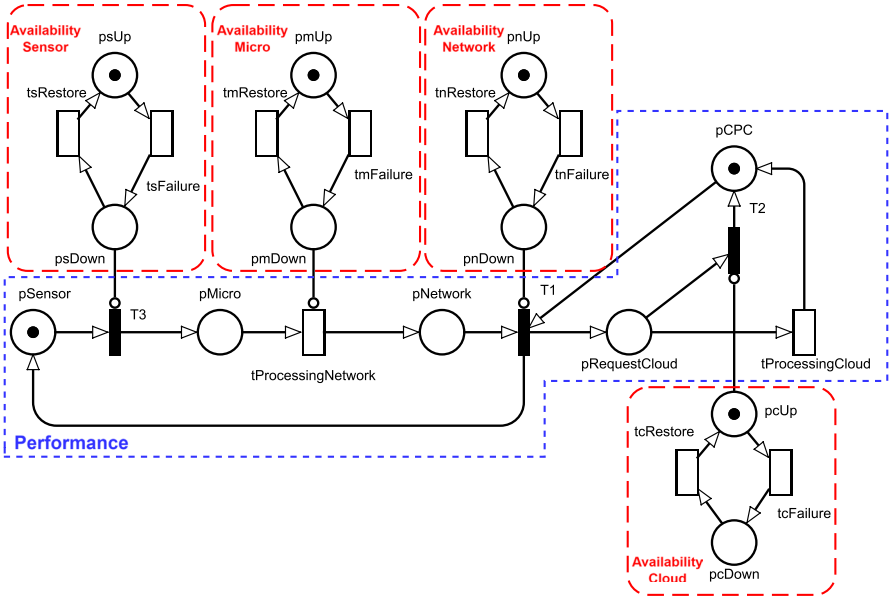

Fig. 8 Integrated availability and performance model

denotes the respective failure. Transitions *tXFailure* and *tXRestore* represent the failure and maintenance of a (set of) devices in the layer, and the respective delays correspond to the device MTTF and MTTR. System availability may be estimated as $P\{(\#psUp > 1)$ and $(\#pmUp > 1)$ and $(\#pnUp > 1)$ and $(\#pcUP > 1)\}$. If any layer is not operational, the system fails.

The performance block (in blue) represents data transmission. A token in place *pSensor* indicates data was obtained by a sensor, and it is immediately (transition $T3$) acquired by the microcontroller (place *pMicro*). The microcontroller and the sensor are connected to the same board, and thus, the respective delay is considered negligible. Transition $T3$ has an inhibitor arc, which does not allow data acquisition when the sensor is down.

The microcontroller then transmits patient data over the network (transition *tProcessingNetwork*), and then, the sensor data is received at the private cloud (firing of transition *tProcessingCloud*). Place (*pCPC*) represents the processing capacity for receiving data in the private cloud. For instance, one token limits the capacity to one request at a time, and additional tokens ($\#pCPC > 1$) allow the assessment of dealing with multiple requests simultaneously. Similar to the sensor, transition $T1$ and $T2$ verify if the components are down. In case of failure, data transmission is not carried out. Regarding data being received in the cloud, data is lost in case of failure (transition $T2$).

Besides, Little's law [8] is adopted to estimate the mean response time (R), which is calculated as follows: $R = \frac{L}{\lambda}$. L is the average number of messages on the system, and λ is the processing throughput. More, specifically,

$L = E\{\#pSensor\} + E\{\#pMicro\} + E\{\#pNetwork\} + E\{\#pRequestCloud\}$ and $\lambda = P\{\#pRequestCloud > 0\} \times 1/W(tProcessingCloud)$.

6.2.1 Redundant Components

Redundant components can be represented by adding tokens in a place *XUp*. For instance, two tokens in place *pcUp* ($\#pcUp = 2$) indicate two working machines: the primary and a spare. Besides, transitions need to adopt infinite-server semantics, which is utilized to represent parallel activities. In this case, the firing rate of a transition is linearly increased according to its enabling degree. The reader is referred to [40] for more details.

7 Experimental Results

This section presents experimental results to demonstrate the feasibility of the proposed technique. Initially, model validation is described, and next, the experiments are presented.

7.1 Experimental Setting

Figure 9 illustrates the evaluation system utilized in this work, which contemplates two physical machines (a client and a server with a virtual machine) and one switch. The client machine, adopting JMeter [41], simulates the microcontroller and also collects data regarding the server response time. The server hosts a virtual machine and is connected to a switch. Message queuing telemetry transport (MQTT) [42] protocol is adopted for the communication between the client and server, as such a protocol is commonly adopted for systems with constrained computing resources [43].

The client machine utilizes an Intel i3-10110U 2.59 GHz processor with 24 GB of RAM and 500 GB of storage. The server adopts an AMD Ryzen 5 3500X 3.59 GHz processor with 16 GB of RAM and 1 TB of storage. Both machines utilize CentOS 7. Apache Cloudstack has been adopted as the private cloud computing platform, and Mosquitto is utilized as the broker in the server for the communication using MQTT.

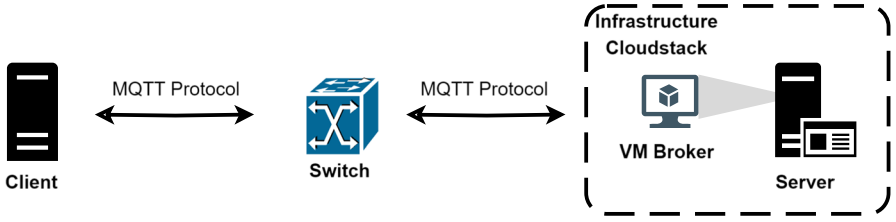

Fig. 9 Evaluation system

7.2 Validation

This section presents the validation of the conceived SPN and RBD models. The main idea is to verify whether the models produce estimates close to a real system. The SPN model has been evaluated using TimeNet [44], and Mercury [45] has been adopted for RBD evaluation.

7.2.1 Availability

Availability validation considers a RBD model composed of a sensor, microcontroller, switch, and a private cloud using series arrangement (Fig. 6). The adopted approach utilizes fault injection [46] in the network interface to represent system failures, and Table 3 presents each component *MTTF* and *MTTR*.

For each component, a thread is created using Algorithm 1 [47], which is a script responsible for injecting failures. *cTime* represents the current monitoring time (in hours), and *maxTime* is the maximum observation period (in hours). In each iteration, a time to failure (*ttf*) is generated using the exponential distribution and the component *MTTF* (line 5). The thread sleeps for the time defined in *ttf* (line 6), and then, the network interface is disabled. Similarly, the algorithm generates a time to repair (*ttf*) (line 8), sleeps, and, next, the component is restored to the working state. *cTime* is incremented (increment(*cTime*)) taking into account the elapsed time (i.e., the sum of *ttf* and *ttr*). Whenever *cTime* is greater than or equal to *maxTime*, the thread concludes the execution.

Algorithm 1 Fault Injection

```
1:  cTime ← 0;
2:  maxTime ← 34;
3:  while cTime < maxTime do
4:      Generate time to failure (ttf);
5:      Sleep(ttf);
6:      Interrupt system;
7:      Generate time to repair (ttr);
8:      Sleep(ttr);
9:      Restore system;
10:     increment(cTime);
11: end while
```

Algorithm 2 [47] depicts the script for monitoring the system state (available or unavailable). A loop performs periodic communications with the network Interface. If a change in the respective state is detected, the system status is recorded to estimate system availability. The observation period was 34 h, which is close to one year, assuming an acceleration factor of 100. In other words, MTTF and MTTR are divided by 100 to reflect the acceleration in failure and repair.

Algorithm 2 Availability Monitor

```
1: status ← up;
2: while True do
3:     if network connected then
4:         if status ≠ up then
5:             status ← up;
6:             Register repair in a log;
7:         end if
8:     else if network not connected then
9:         if status = up then
10:            status ← down;
11:            Register failure in a log;
12:        end if
13:    end if
14: end while
```

Using the collected data and the technique described in [48], a 95% confidence interval is obtained: [0.99329, 0.99493]. Utilizing the RBD model, availability is estimated as 0.99470. Since such a value is contained in the interval, there is no statistical evidence to refute the equivalence between the proposed model and the real system.

7.2.2 Performability

For performance, the proposed approach utilizes the SPN model presented in Fig. 8 without component failures. Using the system depicted in Fig. 9, 30 samples were collected to estimate the mean delays for each timed transition and the system response time. Such a sample size is related to the central limit theorem [49]. The estimated delays are 0.001 s and 0.14133 s for transition *tProcessingNetwork* and *tProcessingCloud*, respectively.

Adopting stationary analysis [8], the proposed model obtains 0.01513 s for the system response time. Utilizing the data collected in the real system, [0.013962, 0.016305] represents the 95% confidence interval for response time using Student's t distribution. As 0.01513 s is contained in the interval, there is no statistical evidence that the model does not represent the real system.

7.3 Experiments

Two experiments are carried out to evaluate system availability and response time. A design of experiments (DoE) approach [49] is adopted, assuming a two-level (2^k) factorial design.

The following sections present results using rank of effects. Effect [49] is the change in response due to a change in the factor level. All ranks are presented

Table 2 Factor and levels

Factor	Level 1	Level 2
MTTF SNS	44,957	67,435.5
MTTF MCU	28,011	42,016.5
MTTF NT	10,000	15,000.0
MTTF HW	8760	13,140.0

Table 3 MTTF and MTTR of model components

Module	Input parameters		
	Component	MTTF	MTTR
Microcontroller	MCU	44,957.0	5.0
Sensor	SNS	28,011.0	5.0
Network	NT	10,000.0	1.0
Private cloud	HW	8760.0	1.66
	SO	2893.0	0.25
	KVM	2990.0	1.0
	VM	217.8	0.94

in descending order, considering the absolute values of all effects. Besides, this work considers main effects and second-order interactions since higher-order interactions are usually negligible.

7.3.1 Availability

This section presents the experiment for assessing the adopted architecture using the conceived RBD models. The experiment evaluates the influence of distinct MTTF values of each component on system availability. To isolate the impact of component failures, MTTR values are kept constant for all treatments.

Table 2 depicts the adopted factors (e.g., MTTF SNS). Level 1 denotes the actual value, and Level 2 represents an improved component with a 50% increase in the mean time to failure. *SNS* represents the sensor, *MCU* denotes the microcontroller, *NT* is the network, and *HW* is the hardware of the private cloud. For each treatment (i.e., a combination of factor levels), a model is created (see Fig. 6) to conduct the availability analysis. Table 3 shows the base MTTF and MTTR values adopted, which are based on [50–52].

Table 4 presents system availability (A) and downtime ($D = [1 - A] \times 8760$) for each treatment in a year (8760 h). Table 5 depicts the rank of effects. *HW* is the most important factor since improving its MTTF, availability increases by 0.06% (4.5 h of downtime). Other factors also have a significant effect, and they

Table 4 Treatments and results—availability

Treatment	MTTF SNS	MTTF MCU	MTTF NT	MTTF HW	Availability	Downtime
1	44,957.0	28,011.0	10,000.0	8760.0	0.9994209	5.0754
2	67,435.5	28,011.0	10,000.0	8760.0	0.9994580	4.7506
3	44,957.0	42,016.5	10,000.0	8760.0	0.9994804	4.5542
4	67,435.5	42,016.5	10,000.0	8760.0	0.9995175	4.2294
5	44,957.0	28,011.0	15,000.0	8760.0	0.9994543	4.7834
6	67,435.5	28,011.0	15,000.0	8760.0	0.9994913	4.4586
7	44,957.0	42,016.5	15,000.0	8760.0	0.9995137	4.2622
8	67,435.5	42,016.5	15,000.0	8760.0	0.9995508	3.9374
9	44,957.0	28,011.0	10,000.0	13,140.0	0.9994841	4.5221
10	67,435.5	28,011.0	10,000.0	13,140.0	0.9995211	4.1973
11	44,957.0	42,016.5	10,000.0	13,140.0	0.9995435	4.0008
12	67,435.5	42,016.5	10,000.0	13,140.0	0.9995806	3.6760
13	44,957.0	28,011.0	15,000.0	13,140.0	0.9995174	4.2301
14	67,435.5	28,011.0	15,000.0	13,140.0	0.9995544	3.9053
15	44,957.0	42,016.5	15,000.0	13,140.0	0.9995768	3.7088
16	67,435.5	42,016.5	15,000.0	13,140.0	0.9996139	3.3840

Table 5 Rank of effects—availability

Availability		Downtime	
Factor/interaction	Effect	Factor/interaction	Effect
MTTF HW	0.000063	MTTF HW	− 0.5533
MTTF MCU	0.000059	MTTF MCU	− 0.5212
MTTF SNS	0.000037	MTTF SNS	− 0.3248
MTTF NT	0.000033	MTTF NT	− 0.2920

are followed by the microcontroller as well as the interaction of the sensor and network.

7.3.2 Performability

This section presents the experiment that jointly quantifies the impact of distinct workloads on availability and response time. In this case, the conceived SPN model (Fig. 8) is utilized.

Table 6 depicts the adopted DoE, assuming 5 factors. *pSensor* denotes the workload, which represents the number of concurrent messages transmitted by the microcontroller to the private cloud. *psUp*, *pmUp*, *pnUp* and *pcUp* contemplate the presence of redundancy (level 2) or not (level 1) regarding the sensor, microcontroller, network, and private cloud, respectively. The factor name also indicates the place in the model, and the level denotes the marking (e.g. $\#psUP = 2$). This experiment also adopts the MTTF and MTTR depicted in Table 3.

Table 6 Factor and levels

Factor	Level 1	Level 2
pSensor	1	5
psUp	1	2
pmUp	1	2
pnUp	1	2
pcUp	1	2

Table 7 Treatments and results—availability and response time

Treatment	pSensor	psUp	pmUp	pnUp	pcUp	Availability	Response time
1	1	1	1	1	1	0.99477030	0.02840677
2	5	1	1	1	1	0.99477029	0.08516914
3	1	2	1	1	1	0.99488092	0.02840835
4	5	2	1	1	1	0.99488092	0.08517690
5	1	1	2	1	1	0.99494783	0.02840930
6	5	1	2	1	1	0.99494783	0.08518161
7	1	2	2	1	1	0.99505847	0.02841088
8	5	2	2	1	1	0.99505847	0.08518938
9	1	1	1	2	1	0.99486976	0.02840817
10	5	1	1	2	1	0.99486976	0.08517614
11	1	2	1	2	1	0.99498039	0.02840975
12	5	2	1	2	1	0.99498039	0.08518391
13	1	1	2	2	1	0.99504732	0.02841071
14	5	1	2	2	1	0.99504732	0.08518861
15	1	2	2	2	1	0.99515797	0.02841229
16	5	2	2	2	1	0.99515797	0.08519638
17	1	1	1	1	2	0.99958689	0.02833791
18	5	1	1	1	2	0.99958689	0.08482677
19	1	2	1	1	2	0.99969805	0.02833948
20	5	2	1	1	2	0.99969804	0.08483450
21	1	1	2	1	2	0.99976528	0.02834043
22	5	1	2	1	2	0.99976528	0.08483918
23	1	2	2	1	2	0.99987646	0.02834200
24	5	2	2	1	2	0.99987646	0.08484691
25	1	1	1	2	2	0.99968684	0.02833931
26	5	1	1	2	2	0.99968683	0.08483374
27	1	2	1	2	2	0.99979801	0.02834088
28	5	2	1	2	2	0.99979800	0.08484147
29	1	1	2	2	2	0.99986525	0.02834183
30	5	1	2	2	2	0.99986525	0.08484615
31	1	2	2	2	2	0.99997644	0.02834340
32	5	2	2	2	2	0.99997644	0.08485388

Table 8 Rank of effects—response time and availability

Response time		Availability	
Factor/interaction	Effect	Factor/interaction	Effect
Psensor	0.056640	pcUp	0.004818
pmUp	0.000007	pmUp	0.000178
Psensor*pmUp	0.000005	psUp	0.000111
psUp	0.000005	pnUp	0.000001
pnUp	0.000004	–	–
Psensor*pnUp	0.000026	–	–

For each treatment, a model based on Fig. 8 is created, and stationary analysis has been utilized to estimate availability and response time. Table 7 depicts the system availability and response time for each treatment, and Table 8 presents the rank of effects. *Psensor* is the most important factor for response time. For instance, comparing treatment 1 (one message) to treatment 2 (5 concurrent messages), the response is increased by 199%. The private cloud is the most important factor for availability. For instance, by adding a spare component to this device (treatment 17), system availability improved, reducing downtime by approximately 20% (from 4.57 h to 3.62 h).

The conceived technique represents an additional tool for designers to assess the influence of each component on system performability and availability.

7.4 Remarks

As presented, results indicate devices with a longer time to fail significantly increase system availability, which is prominent in keeping continuous patient monitoring. Additionally, hot standby redundancy may significantly decrease downtime (around 20%), whenever such a technique is adopted to the components with the highest influence on system operation.

Regarding performability, *Psensor* is an important factor due to its direct influence on response time. Experiments indicate concurrent messages may increase response time by 199%. Redundancy also reduces response time, as system downtime is shorter.

The proposed approach contributes to state-of-the-art as an additional technique for designing IoMT systems. Distinct designs can be assessed, taking account the concomitantly evaluation of dependability attributes (e.g., MTTF) and performance (e.g., response time).

8 Conclusion

This paper presented a modeling approach based on stochastic Petri nets and reliability block diagrams to evaluate IoMT architectures. The models were validated using a real testbed system, demonstrating that our approach provides performance

and availability estimates close to those of a real system. The conceived models make it possible to evaluate distinct system designs before modifying or implementing the actual system or prototype.

Two experiments were presented to show the applicability of our approach. Results demonstrate the influence of distinct designs, and a sensitivity analysis indicates the components with the highest influence on system availability. As a future work, a response time index will be proposed as an alternative technique to identify the device with the highest influence n system response time.

References

1. Sorri, K., Mustafee, N., Seppänen, M.: Revisiting IoT definitions: a framework towards comprehensive use. Technol. Forecast. Soc. Chang. **179**, 121623 (2022)
2. Pawar, N., Bourgeau, T., Chaouchi, H.: Study of IoT architecture and application invariant functionalities. In: 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 667–671. IEEE (2021)
3. Cisco, T., Internet, A.: Cisco: 2020 CISO benchmark report. Comput. Fraud Secur. **2020**(3), 4–4 (2020)
4. Verma, D., Singh, K.R., Yadav, A.K., Nayak, V., Singh, J., Solanki, P.R., Singh, R.P.: Internet of things (IoT) in nano-integrated wearable biosensor devices for healthcare applications. Biosens. Bioelectron.: X **11**, 100153 (2022)
5. Research, P.: Internet of things in healthcare market. https://www.precedenceresearch.com/internet-of-things-in-healthcare-market. Accessed 10 Sept 2023 (2023)
6. Tang, S., Xie, Y.: Availability modeling and performance improving of a healthcare internet of things (IoT) system. IoT **2**(2), 310–325 (2021)
7. Xing, L.: Reliability in internet of things: current status and future perspectives. IEEE Internet Things J. **7**(8), 6704–6721 (2020)
8. Maciel, P.R.M.: Performance, Reliability, and Availability Evaluation of Computational Systems, Volume I: Performance and Background. Chapman and Hall/CRC (2022)
9. Wai, K.T., Aung, N.P., Htay, L.L.: Internet of things (IoT) based healthcare monitoring system using NodeMCU and Arduino UNO. Int. J. Trend Sci. Res. Dev. (IJTSRD) **3**(5), 755–759 (2019)
10. Pokorni, S.J.: Reliability and availability of the internet of things. Vojnotehnicki glasnik/Mil. Tech. Cour. **67**(3), 588–600 (2019)
11. Ruman, M.R., Barua, A., Rahman, W., Jahan, K.R., Roni, M.J., Rahman, M.F.: IoT based emergency health monitoring system. In: 2020 International Conference on Industry 4.0 Technology (I4Tech), pp. 159–162. IEEE (2020)
12. Athira, A., Devika, T., Varsha, K., et al.: Design and development of IoT based multi-parameter patient monitoring system. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 862–866. IEEE (2020)
13. Valentim, T., Callou, G., Vinicius, A., França, C., Tavares, E.: Availability assessment of internet of medical things architecture using private cloud. In: Anais do L Seminário Integrado de Software e Hardware, pp. 13–23. SBC (2023)
14. Shen, Y., Shen, S., Wu, Z., Zhou, H., Yu, S.: Signaling game-based availability assessment for edge computing-assisted IoT systems with malware dissemination. J. Inf. Secur. Appl. **66**, 103140 (2022)
15. Rahmani, A.M., Hosseini Mirmahaleh, S.Y.: Flexible-clustering based on application priority to improve IoMT efficiency and dependability. Sustainability **14**(17), 10666 (2022)
16. Santos, G.L., Gomes, D., Kelner, J., Sadok, D., Silva, F.A., Endo, P.T., Lynn, T.: The internet of things for healthcare: optimising e-health system availability in the fog and cloud. Int. J. Comput. Sci. Eng. **21**(4), 615–628 (2020)
17. Nguyen, T.A., Min, D., Choi, E., Lee, J.-W.: Dependability and security quantification of an internet of medical things infrastructure based on cloud-fog-edge continuum for healthcare monitoring using hierarchical models. IEEE Internet Things J. **8**(21), 15704–15748 (2021)

18. Santos, L., Cunha, B., Fé, I., Vieira, M., Silva, F.A.: Data processing on edge and cloud: a performability evaluation and sensitivity analysis. J. Netw. Syst. Manag. **29**(3), 27 (2021)
19. Clemente, D., Pereira, P., Dantas, J., Maciel, P.: Availability evaluation of system service hosted in private cloud computing through hierarchical modeling process. J. Supercomput. **78**(7), 9985–10024 (2022)
20. Andrade, E., Nogueira, B., Farias Júnior, I.d., Araújo, D.: Performance and availability trade-offs in fog-cloud IoT environments. J. Netw. Syst. Manag. **29**, 1–27 (2021)
21. Lima, L.N., Sabino, A., Barbosa, V., Feitosa, L., Brito, C., Araujo, J., Silva, F.A.: Dependability analysis and disaster recovery measures in smart hospital systems. J. Reliab. Intell. Environ. 1–17 (2024)
22. Mallick, S.R., Goswami, V., Lenka, R.K., Sharma, S., Barik, R.K.: Performance evaluation of queueing assisted IoMT-fog-blockchain framework for healthcare organizations. In: 2022 IEEE 9th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), pp. 1–6. IEEE (2022)
23. El Kafhali, S., Salah, K.: Efficient and dynamic scaling of fog nodes for IoT devices. J. Supercomput. **73**, 5261–5284 (2017)
24. Djahafi, M., Salmi, N.: Using stochastic petri net modeling for self-adapting publish/subscribe IoT systems. In: NOMS 2023—2023 IEEE/IFIP Network Operations and Management Symposium, pp. 1–4. IEEE (2023)
25. Ghosh, A., Saha, R., Misra, S.: Persistent service provisioning framework for IoMT based emergency mobile healthcare units. IEEE J. Biomed. Health Inform. **26**(12), 5851–5858 (2022)
26. Nakayama, F., Lenz, P., Nogueira, M.: A resilience management architecture for communication on portable assisted living. IEEE Trans. Netw. Serv. Manag. **19**(3), 2536–2548 (2022)
27. Nguyen, T.A., Fe, I., Brito, C., Kaliappan, V.K., Choi, E., Min, D., Lee, J.W., Silva, F.A.: Performability evaluation of load balancing and fail-over strategies for medical information systems with edge/fog computing using stochastic reward nets. Sensors **21**(18), 6253 (2021)
28. Gupta, A., Chaurasiya, V.K.: Efficient task-offloading in IoT-fog based health monitoring system. In: 2022 OITS International Conference on Information Technology (OCIT), pp. 495–500. IEEE (2022)
29. Ezhilarasi, M., Kumar, A., Shanmugapriya, M., Ghanshala, A., Gupta, A.: Integrated healthcare monitoring system using wireless body area networks and internet of things. In: 2023 4th International Conference on Innovative Trends in Information Technology (ICITIIT), pp. 1–5. IEEE (2023)
30. Islam, M.M., Nooruddin, S., Karray, F., Muhammad, G.: Internet of things: device capabilities, architectures, protocols, and smart applications in healthcare domain. IEEE Internet Things J. **10**(4), 3611–3641 (2022)
31. Jara, A.J., Zamora, M.A., Skarmeta, A.F.: An architecture for ambient assisted living and health environments. In: International Work-Conference on Artificial Neural Networks, pp. 882–889. Springer, Berlin (2009)
32. Rahmani, A.M., Gia, T.N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., Liljeberg, P.: Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach. Futur. Gener. Comput. Syst. **78**, 641–658 (2018)
33. Loh, F., Mehling, N., Metzger, F., Hoßfeld, T., Hock, D.: LoRaPlan: A software to evaluate gateway placement in LoRaWAN. In: 2021 17th International Conference on Network and Service Management (CNSM), pp. 385–387 (2021). IEEE
34. Joyia, G.J., Liaqat, R.M., Farooq, A., Rehman, S.: Internet of medical things (IoMT): applications, benefits and future challenges in healthcare domain. J. Commun. **12**(4), 240–247 (2017)
35. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
36. Hireche, R., Mansouri, H., Pathan, A.-S.K.: Security and privacy management in internet of medical things (IoMT): a synthesis. J. Cybersecur. Privacy **2**(3), 640–661 (2022)
37. Askar, N.A., Habbal, A., Mohammed, A.H., Sajat, M.S., Yusupov, Z., Kodirov, D.: Architecture, protocols, and applications of the internet of medical things (IoMT). J. Commun. **17**(11) (2022)
38. Vishnu, S., Ramson, S.J., Jegan, R.: Internet of medical things (IoMT)—an overview. In: 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), pp. 101–104. IEEE (2020)
39. Buzacott, J.A.: Finding the MTBF of repairable systems by reduction of the reliability block diagram. Microelectron. Reliab. **6**(2), 105–112 (1967)
40. Balbo, G.: Introduction to stochastic petri nets. In: Lectures on Formal Methods and PerformanceAnalysis: First EEF/Euro Summer School on Trends in Computer Science Bergen Dal, The Netherlands, July 3–7, 2000 Revised Lectures 1, pp. 84–155 (2001)

41. Halili, E.H.: Apache JMeter (2008)
42. Standard, O.: Mqtt version 5.0. Retrieved June 22, 2020 (2019)
43. Alshammari, H.H.: The internet of things healthcare monitoring system based on MQTT protocol. Alex. Eng. J. **69**, 275–287 (2023)
44. Zimmermann, A., et al.: Towards version 4.0 of TimeNET. In: 13th GI/ITG Conference—Measuring, Modelling and Evaluation of Computer and Communication Systems, pp. 1–4 (2006)
45. Silva, B., Matos, R., Callou, G., Figueiredo, J., Oliveira, D., Ferreira, J., Dantas, J., Lobo, A., Alves, V., Maciel, P.: Mercury: An integrated environment for performance and dependability evaluation of general systems. In: Proceedings of Industrial Track at 45th Dependable Systems and Networks Conference, DSN, pp. 1–4 (2015)
46. Hsueh, M.-C., Tsai, T.K., Iyer, R.K.: Fault injection techniques and tools. Computer **30**(4), 75–82 (1997)
47. Gomes, C., Tavares, E., Junior, M.N.d.O., Nogueira, B.: Cloud storage availability and performance assessment: a study based on NOSQL DBMS. J. Supercomput. **78**(2), 2819–2839 (2022)
48. Keesee, W.: A method of determining a confidence interval for availability. Technical report, Naval Missile Center Point (1965)
49. Montgomery, D.C., Runger, G.C.: Applied Statistics and Probability for Engineers. John Wiley & Sons (2010)
50. Tang, D., Kumar, D., Duvur, S., Torbjornsen, O.: Availability measurement and modeling for an application server. In: International Conference on Dependable Systems and Networks, 2004, pp. 669–678. IEEE (2004)
51. Kim, D.S., Machida, F., Trivedi, K.S.: Availability modeling and analysis of a virtualized system. In: 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing, pp. 365–371. IEEE (2009)
52. Novacek, G.: Tips for predicting product reliability. https://circuitcellar.com/cc-blog/tips-for-predicting-product-reliability/. Accessed 10 Sept 2023 (2022)

Thiago Valentim graduated with a degree in Information Systems in 2015 and received his M.Sc. in Applied Computer Science from the Universidade Federal Rural de Pernambuco (UFRPE) in 2019. He is currently a Ph.D. candidate in Computer Science at the Center for Informatics, Universidade Federal de Pernambuco (UFPE), and a Professor at the Instituto Federal de Pernambuco. His research interests include computer networks, information security, Internet of Things, cloud computing, performance and dependability evaluation, Petri nets, and formal models.

Gustavo Callou holds a Ph.D. in Computer Science, which was obtained in 2013 from the Federal University of Pernambuco (UFPE). Between 2010 and 2011, he was a student researcher at the University of Wuppertal, Germany, and between 2023 and 2024, he completed a postdoctoral fellowship at the University of Coimbra, Portugal. He is currently an Associate Professor in the Department of Computing at the Federal Rural University of Pernambuco (UFRPE). He has extensive experience in the field of Computer Science, with a focus on Performance Evaluation and System Optimization. His primary research interests include performance and dependability modeling, cloud computing, data centers, energy efficiency, and sustainability.

Cleunio França graduated with a degree in Computer Engineering in 2010 and received his M.Sc. in Computer Science from the Universidade Federal de Pernambuco (UFPE) in 2013. Currently, he is a Ph.D. candidate at the same university. His research interests include the internet of things, artificial intelligence, machine learning, performance and dependability evaluation, Petri nets, formal models, and real-time systems.

Eduardo Tavares graduated in Computer Science in 2002, and he received his M.Sc. as well as Ph.D. degrees in Computer Science from Universidade Federal de Pernambuco (UFPE) in 2006 and 2010, respectively. Since 2011, he has been a member of Center for Informatics - UFPE, where he is currently Associate Professor. His research interests include performance and dependability evaluation, Petri nets, formal models, and real-time systems.

Authors and Affiliations

Thiago Valentim[1] · Gustavo Callou[2] · Cleunio França[1] · Eduardo Tavares[1]

✉ Thiago Valentim
tvb@cin.ufpe.br

Gustavo Callou
gustavo.callou@ufrpe.br

Cleunio França
cbff@cin.ufpe.br

Eduardo Tavares
eagt@cin.ufpe.br

[1] Centro de informática, Universidade Federal de Pernambuco, Recife, Pernambuco 100190, Brazil

[2] Departamento de computação, Universidade Federal Rural de Pernambuco, Recife, Pernambuco 10587, Brazil