




Performance and availability evaluation of an smart hospital architecture

Laécio Rodrigues¹ · Igor Gonçalves¹ · Iure Fé³ · Patricia Takako Endo² · Francisco Airton Silva¹ 

Received: 30 November 2020 / Accepted: 23 June 2021 / Published online: 3 July 2021

© The Author(s), under exclusive licence to Springer-Verlag GmbH Austria, part of Springer Nature 2021

Abstract

Low latency and high availability of resources are essential characteristics to guarantee the quality of services in health systems. Hospital systems must be efficient to prevent loss of human life. Smart hospitals promise a health revolution by capturing and transmitting patient data to doctors in real-time via a wireless sensor network. However, there is a significant difficulty in assessing the performance and availability of such systems in real contexts due to failures not being tolerated and high implementation costs. This paper adopts analytical models to assess the performance and availability of intelligent hospital systems without having to invest in real equipment beforehand. Two Stochastic Petri Net models were proposed to represent intelligent hospital architectures. One model is used to assess performance, and another to assess availability. The models are pretty parametric, making it possible to calibrate the resources, service times, times between failures, and times between repairs. The availability model, for example, allows you to define 48 parameters, allowing you to evaluate a large number of scenarios. The analysis showed that the arrival rate in the performance model is an impacting parameter. It was possible to observe the close relationship between MRT, resource utilization, and discard rate in different scenarios, especially for high arrival rates. Three scenarios were explored considering the second model. The highest availability results were observed in scenario A, composed of server redundancy (local and remote). Such scenario—with redundancy—presented an availability of 99.9199%, that is, 7.01 h/year of inactivity. In addition, this work presents a sensitivity analysis that identifies the most critical components of the architecture. Therefore, this work can help hospital system administrators plan more optimized architectures according to their needs.

Keywords Smart hospital · Stochastic petri net · Performance · Availability · Internet of Things

Mathematics Subject Classification 65 · 68

1 Introduction

The Internet of Things (IoT) is innovating in all areas of our society, connecting millions of handsets and making everyday life easier for people. The health sector can also benefit from IoT. The use of intelligent devices and sensors can revolutionize people's health in any environment, such as at home or on the streets. In a hospital, it is no different. Vital signs of patients captured by sensors can be transmitted to Intensive Care Units (ICU), bed occupancy rate, employee productivity index, among others. Therefore, there are many possibilities of IoT in the health field [1]. The health crisis caused by the COVID-19 pandemic has highlighted the fragility of health systems worldwide and the need to optimize the use of existing (scarce) resources. To prepare for a pandemic like the one we are facing, hospitals need a strategy to manage their space, staff, and supplies in order to provide the best care to patients [2].

IoT in health has stood out as a trend that allows the creation of new treatments, more precise, and more wealth of information, [3–5]. With these treatments, it is possible to detect the permanent flow of data providing a complete picture of the patient's illness. The agility of the services allows doctors to respond to any change on time. Doctors rely on continuous real-time information. The information is captured through not only one but a wireless sensor network (WSN). WSN integrates a series of spatially distributed autonomous sensors into a network and cooperatively transmits its data through wireless communication. These sensors capture patient data and route them through wireless communication to a gateway responsible for processing that data. Subsequently, the gateway can route the data to several distributed servers, local (edge computing), or cloud computing [4]. Edge computing is a distributed architecture that presents decentralized processing power, enabling mobile computing and IoT technologies. In edge computing, data is processed by devices near the end-user (computer or local server) instead of being transmitted to the cloud [6,7], which is quite attractive considering smart hospitals.

These sensor networks are responsible for monitoring patients and notifying physicians in cases of emergency. However, these systems can not face request failures or high latency since they can compromise a specific person's health. The remote computing resources should be cleverly designed considering multiple and concomitant factors. The health system project design must observe all layers' computing capacity to avoid idle and overloaded machines. This project design conducted with system evaluation is very complex and expensive [8,9]. Such evaluation certainly can bring monetary savings to the health center or hospitals. Therefore, evaluating the performance and availability of smart hospital systems, even before implementing a real infrastructure, is vital [10–12]. Analytical models can mitigate such challenges by producing system behavior forecasts according to probability calculations. Stochastic Petri Nets (SPNs) are analytical models representing complex systems with diverse characteristics, including parallelism and concurrency. SPNs have probabilistic fundamentals and with very reliable results if well applied. We are using Stochastic Petri Nets (SPNs) as our modelling approach [13–22]. SPN is a formal method commonly used to describe performance of systems, covering several aspects, such as concurrency and synchronous mechanisms. Transitions are associated with a random firing delay that follows a stochastic process. Moreover, SPNs are also widely used to model

availability and performance of systems as in the case presented in this paper. Usually the performance evaluation of systems through analytical modeling uses one of the most traditional models: Queuing models, Markov chains, and Petri nets. Queuing models are very powerful but with some limitations when the model has many components. Markov chains and Petri nets (stochastic) are equivalent, however, SPN are much simpler and representative.

This paper proposes to evaluate, using SPN models, the use of resources, the mean response time (MRT), the availability, and the downtime of an architecture of a smart hospital, leading to optimization. In summary, the main contributions of this paper are:

1. An SPN model for evaluating the performance of computing architectures in smart hospital, with the ability to configure up to 13 parameters.
2. An SPN model for assessing the availability of computing architectures in smart hospitals, with the ability to configure average failure and recovery times for 16 components.
3. Case studies using the proposed SPN models serve as guides for system administrators to plan their specific hospital infrastructures.

The rest of this paper is divided as follows: Sect. 2 presents some concepts to understand this work; Sect. 3 discusses the main related works; Sect. 4 presents the architecture of a smart hospital that was used as a basis for designing the models; Sect. 5 presents an SPN model for performance evaluation of smart hospitals; Sect. 6 presents an SPN model for assessing availability of smart hospitals; Sect. 8 outlines some conclusions and future work.

2 Background

This section presents the background needed to understand the paper proposal. The following topics are discussed: IoT, smart hospitals, cloud computing, edge computing, and Stochastic Petri Nets.

2.1 IoT and smart hospitals

The Internet of Things (IoT) is innovating all areas of our society and describes a scenario in which diverse objects are connected and communicating. The IoT is formed by a network of physical objects capable of collecting and transmitting data. This technological innovation has as main objective to connect the objects that we use day-by-day to the Internet, approaching the physical world of the digital. The connection of these objects seeks to bring more convenience, collecting data by sensors, and fast information processing. Such objects can range from smartwatches [23] to refrigerators [24] to automatically order products from the grocery store.

IoT still has a long pathway to evolve, but the main strength of the IoT is the high impact on several aspects of everyday life and behavior of potential users, not only in e-health. For the private user, the most apparent effects of the introduction will be visible in both the working and domestic fields. In this context, assisted living, e-health, and

enhanced learning are only a few examples of possible application scenarios in which the new paradigm will soon play a leading role. Similarly, from the perspective of business users, the most apparent consequences will be equally visible in fields such as automation and industrial manufacturing, logistics, business/process management, intelligent transportation of people and goods [25].

In the eHealth area, there are several examples of connecting medical devices on the Internet to perform different remote medical services such as remote patients' monitoring, elderly persons' supervision, online medical consultations, or even robotic arm control for surgical interventions [26,27]. A real boost in this area was given by the latest miniaturized portable medical devices and gadgets. These devices may be used for continuous measurement of medical parameters (e.g., ECG, blood pressure, temperature), for activity recognition and monitoring, or remotely-made medical evaluations [28]. There are many particular solutions present on the market, but with very restricted accessibility.

Smart hospital is the concept used to refer to digitized hospitals that rely on automated and optimized processes based on IoT. These devices serve to improve existing patient care procedures in addition to introducing new services. Among these capabilities, we can highlight the continuous monitoring of patients. A smart hospital's goals include providing enhanced patient care, remote medical care, and enhancing diagnostic capabilities to ensure patient safety [29]. Smart Hospitals allow the creation of new treatments, more precise and with a greater wealth of information. With smart hospitals, it will be possible to provide a complete picture of the patient's illness, allowing physicians to respond to any change promptly.

2.2 Cloud and edge computing

Cloud computing is a paradigm in continuous development that originated from the combination of several different technologies. It has been defined as "a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers" [30,31]

The basic principle of cloud computing is assigning computing to many distributed computers rather than local or remote services. It is characterized by the efficient utilization of resources, employing virtualization, resource monitoring, and load balancing mechanisms [32]. In modern societies, the majority of essential services are made available in a transparent way. The water supply, electric power, gas and telephone, essential goods in our daily life, have this characteristic. These market models follow the concept of pay for what you use: the paid value for the service is flexible following the necessity of the organization at any time [33]. Cloud computing provides a similar payment model for the utilization of computing services.

There are three models of implementation of Cloud Computing [34]. A private cloud is a cloud infrastructure provisioned for exclusive use by a single organization comprising multiple consumers. In the public cloud model, the cloud infrastructure is provisioned for open use by the general public that remains unique entities, but they

are bound together by standardized or proprietary technology that enables data and application portability. The hybrid cloud model comprises two or more distinct cloud infrastructures (private, community, or public) that remain unique entities. However, the hybrid clouds introduce additional complexity, the distribution of applications by both models [35]. Briefly, among the benefits associated with utilizing the services on the cloud, we could highlight centralized management, the reduction of energetic consumption, and the decrease of inherent costs to the maintenance of traditional infrastructures. The cloud provides a diversity of services that favors the agility of the market [36].

Data is increasingly produced at the edge of the network. Therefore, it would be more efficient also to process the data at the edge of the network. Previous work such as micro datacenter [37], cloudlet [13,38], and fog computing [39] has been introduced to the community because cloud computing is not always efficient for data processing when the data is produced at the edge of the network.

Putting all the computing tasks on the cloud has been proved to be an efficient way for data processing. The computing power on the cloud usually outclasses the capability of the things at the edge. However, compared to the fast-developing data processing speed, the network's bandwidth has come to a standstill. With the growing quantity of data generated at the edge, the speed of data transportation is becoming the bottleneck for the cloud-based computing paradigm. For example, about 5 Gigabyte data will be generated by a Boeing 787 every second [40], but the bandwidth between the airplane and either satellite or base station on the ground is not large enough for data transmission. Consider an autonomous vehicle as another example. One Gigabyte of data will be generated by the car every second, and it requires real-time processing for the vehicle to make correct decisions [41]. If all the data needs to be sent to the cloud for processing, the response time would be too long. Not to mention that current network bandwidth and reliability would be challenged to support many vehicles in one area. In this case, the data needs to be processed at the edge for shorter response time, more efficient processing, and smaller network pressure.

Therefore, edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services, and upstream data on IoT services. Here we define the edge as any computing and network resources along the path between data sources and cloud data centers. For example, a smartphone is an edge between body and cloud, a gateway in a smart home is the edge between home things and cloud, a micro data center, and a cloudlet is an edge between a mobile device and cloud [42].

2.3 Stochastic petri net

Petri Nets [43] are a family of formalisms very well suited for modeling several system types since concurrency, synchronization, and communication mechanisms are naturally represented. This work adopts a particular extension, namely, Stochastic Petri Nets [44], which allows the association of stochastic delays to timed transitions, and the respective state space can be converted into CTMC [45]. SPN models present a strong mathematical foundation, and they are suitable for representing and analyzing

parallel systems with various components and exhibit concurrency and synchronization aspects. Therefore, this formalism represents a superior choice to model cloud computing systems. In SPNs, Places are represented by circles, whereas transitions are depicted as filled rectangles (immediate transitions) or hollow rectangles (timed transitions).

Arcs (directed edges) connect places to transitions and vice versa. Tokens (small filled circles) may reside in places that denote the state (i.e., marking) of an SPN. An inhibitor arc is a particular arc type that depicts a small white circle at one edge instead of an arrow, and they usually are used to disable transitions if there are tokens present in a place. The behavior of an SPN is defined in terms of a token flow. Tokens are created and destroyed according to the transition firings. Immediate transitions represent instantaneous activities, and they have higher firing priority than timed transitions. Besides, such transitions may contain a guard condition, and a user may specify a different firing priority, among other immediate transitions. SPNs also allow the adoption of simulation techniques for obtaining system metrics as an alternative to the generation of a CTMC. The extended stochastic Petri net definition adopted in this work is:

Let $\mathcal{N} = (P, T, I, O, H, \Pi, M_0, Atts)$ be a stochastic Petri net (SPN), where:

- $P = \{p_1, p_2, \dots, p_n\}$ is the set of places, which may contain tokens and form the discrete state variables of a Petri net. $ordp_{\mathcal{N}}$ corresponds to a bijective function ($ordp_{\mathcal{N}} : P \rightarrow \{1, 2, \dots, n\}$) that maps each place to a unique natural number.
- $T = \{t_1, t_2, \dots, t_m\}$ is the set of transitions, which model active components. $ordt_{\mathcal{N}}$ is a bijective function ($ordt_{\mathcal{N}} : T \rightarrow \{1, 2, \dots, m\}$) that maps each transition to a unique natural number.
- $I \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ is a matrix of marking-dependent multiplicities of input arcs, where $I[ordp_{\mathcal{N}}(p_j), ordt_{\mathcal{N}}(t_k)]$ gives the (possibly marking-dependent) arc multiplicity of input arcs from place p_j to transition t_k [$A \subseteq (P \times T) \cup (T \times P)$ —set of arcs]. A transition is only enabled if there are enough tokens in all input places.
- $O \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ is a matrix of marking dependent multiplicities of output arcs, where $O[ordp_{\mathcal{N}}(p_k), ordt_{\mathcal{N}}(t_j)]$ specifies the possibly marking-dependent arc multiplicity of output arcs from transition t_j to place p_k . When a transition fires, it removes the number of tokens specified by the input arcs from input places and adds the number of tokens given by the output arcs to all output places.
- $H \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ is a matrix of marking-dependent multiplicities describing the inhibitor arcs, where $H[ordp_{\mathcal{N}}(p_j), ordt_{\mathcal{N}}(t_k)]$ returns the possibly marking-dependent arc multiplicity of an inhibitor arc from place p_j to transition t_k . In the presence of an inhibitor arc, a transition is enabled to fire only if every place connected by an inhibitor arc contains fewer tokens than the multiplicity of the arc.
- $\Pi \in \mathbb{N}^m$ is a vector that assigns a priority level to each transition. Whenever there are several transitions fireable at one point in time, the one with the highest priority fires first and leads to a state change.
- $M_0 \in \mathbb{N}^n$ is a vector that contains the initial marking for each place (initial state). In this work, $M(p_n)$ denotes the number of tokens of place p_n at marking M .

- $Atts : (Dist, W, G, Policy, Concurrency)^m$ comprises a set of attributes for the m transitions, where
 - $Dist \in \mathbb{N}^m \rightarrow \mathcal{F}$ is a possibly marking dependent firing probability distribution function. In a stochastic timed Petri net, the time has to elapse between the enabling and firing a transition. The actual firing time is a random variable, for which the distribution is specified by \mathcal{F} . We differ between immediate transitions ($\mathcal{F} = 0$) and timed transitions, for which the domain of \mathcal{F} is $(0, \infty)$.
 - $W \in \mathbb{R}^+$ is the weight function, that represents a firing weight w_t for immediate transitions or a rate λ_t for timed transitions. The latter is only meaningful for the standard case of timed transitions with exponentially distributed firing delays. For immediate transitions, the value specifies a relative probability to fire the transition when there are several immediate transitions enabled in marking, and all have the same probability. A random choice is then applied using the probabilities w_t .
 - $G \in \mathbb{N}^n \rightarrow \{true, false\}$ is a function that assigns a guard condition related to place markings to each transition. Depending on the current marking, transitions may not fire (they are down) when the guard function returns false.
 - $Policy \in \{prd, prs\}$ is the preemption policy (*prd*—*preemptive repeat different* means that when a preempted transition becomes enabled again the previously elapsed firing time is lost; *prs*—*preemptive resume*, in which the firing time related to a preempted transition is resumed when the transition becomes enabled again),
 - $Concurrency \in \{ss, is\}$ is the concurrency degree of transitions, where *ss* represents single server semantics and *is* depicts infinity server semantics in the same sense as in queueing models. Transitions with policy *is* can be understood as having an individual transition for each set of input tokens, all running in parallel.

In many circumstances, it might be suitable to represent the initial marking as a mapping from the set of places to natural numbers ($m_0 : P \rightarrow \mathbb{N}$), where $m_0(p_i)$ denotes the initial marking of place p_i and $m(p_i)$ denotes a reachable marking (reachable state) of place p_i . In this work, the notation $\#p_i$ has also been adopted for representing $m(p_i)$. For more detail about SPN concepts and semantic, the reader is referred to [44].

2.4 Sensitivity analysis

Sensitivity analysis is a measure of the local effect of a given input data concerning the output data, aiming to outline the weak links of computer systems, and from then on, seek to adopt a set of techniques that aim to improve these systems in different scenarios [46]. In a way, the sensitivity analysis can bring necessary security and forward the results within the perspective pre-established by the system administrators.

There are many ways to perform a sensitivity analysis. Several methods are available, such as regression analysis, disturbance analysis (AP), variation one by one, Monte Carlo simulation, parametric differential analysis, correlation analysis, and

percentage difference. The distinction of which method to use is a difficult step, making it necessary to establish which computational resources are available and the characteristics of the problems addressed [47,48].

The percentage difference method was chosen to perform the sensitivity analysis of this work. The percentage difference method is an approach that does not use a continuous domain for the parameter input values. This process calculates the percentage difference by varying a minimum value input parameter to its maximum value. It is necessary to use the entire range of possible values for each parameter to calculate the sensitivity of the parameters [49].

The Eq. 1 shows how the sensitivity index is calculated by the percentage difference method. The expression $\max \{Y(\theta)\}$ and $\min \{Y(\theta)\}$ are, respectively, the maximum and minimum values of output, measured when varying the θ parameter over a range of its n possible values of interest. If $Y(\theta)$ is known to vary monotonically, only the extreme values of θ (i.e., θ_1 and θ_n) can be used to calculate $\max \{Y(\theta)\}$, $\min \{Y(\theta)\}$ and as a result $S_\theta \{Y\}$ [50].

$$S_\theta \{Y\} = \frac{\max \{Y(\theta)\} - \min \{Y(\theta)\}}{\max \{Y(\theta)\}} \quad (1)$$

3 Related work

This section presents some related works. Three aspects are analyzed: IoT in health-care, resource utilization, and edge computing in smart hospitals. Qadri et al. [51] recently published an in-depth survey focused on Healthcare IoT (H-IoT) systems, providing an overview of its structure and requirements, discussing the complementing future technologies in H-IoT, such as machine learning, edge computing, blockchain, big-data, software defined networking (SDN). They state that “*the widespread adoption of the H-IoT systems demands compliance with the standard QoS parameters in terms of latency, accuracy, and availability*”. Additionally, authors also say that the usage of mobile nodes deeply affects the H-IoT system, enforcing the relevance of studies that analyze how to improve the network performance and availability as well.

Uslu et al. [52] focused their work on the usage of IoT technologies in smart hospital environments, bringing discussion about optimization factors, challenges, available technologies and opportunities. They present a holistic analysis based on a five layered architecture of IoT-based smart hospital. Authors state that for an IoT-based smart hospital design, the inadequacies that may arise in each layer result in shortcomings, such as insufficient service planning, high information processing times, inadequate reliability and processing performance and others. Based on models, as proposed in our work, it is possible to evaluate different proposals before implement the final solution in the real environment, avoiding some of the shortcomings highlighted by [52].

Some works consider **resource utilization as parameter** for evaluation of proposals. Oueida et al. [53] propose a resource preservation net (RPN) framework using Petri net. The proposed framework is designed to model non-consumable resources and is theoretically described and validated. RPN applies to a real-life scenario where

key performance indicators such as patient length of stay (LoS), resource utilization rate, and average patient waiting time are modeled and optimized. Greco et al. [54] propose a technological and architectural solution based on Open Source big data technologies to perform real-time analysis of wearable sensor data streams. The proposed architecture is composed of four distinct layers: a sensing layer, a pre-processing layer (Raspberry Pi), a cluster processing layer (Kafka's broker and Flink's mini-cluster), and a persistence layer (Cassandra database). Performance evaluation of each layer has been carried out by considering CPU and memory usage for accomplishing a simple anomaly detection task using the REALDISP dataset. Chen et al. [55] propose the Edge-Cognitive Computing-based (ECC-based) smart-healthcare system. This system can monitor and analyze the physical health of users using cognitive computing. It also adjusts the computing resource allocation of the whole edge computing network comprehensively according to the health-risk grade of each user. The experiments show that the ECC-based healthcare system provides a better user experience and optimizes the computing resources reasonably, as well as significantly improving the survival rates of patients in a sudden emergency.

The closest related papers to this work address the topic **edge computing in smart hospitals**. Zhang et al. [5] propose an architecture to connect smart things in smart hospitals based on NB-IoT and introduce edge computing to deal with the requirement of latency in the healing process. Zhang et al. have developed an infusion monitoring system to monitor the real-time drop rate and the volume of remaining drugs during the intravenous infusion. Also, they discuss the challenges and future directions for building a smart hospital by connecting intelligent things. Rahmani et al. [4] exploit the strategic position of gateways at the edge of the network to offer several higher-level services such as local storage, real-time data processing, embedded data mining, etc. Rahmani et al. then propose to exploit the concept of Fog Computing in Healthcare IoT systems by forming a Geo-distributed intermediary layer between smart sensor nodes and the cloud.

Some related papers address SPN models to assess the availability of health computing systems with support for IoT sensors. [56] presents a model of the Stochastic Petri Net (SPN) of a Wearable Health Monitoring System (WHMS) with several sensors, including a corresponding simulation structure implemented in Java. [57] proposed a high-level model that characterizes the behavior of an m-Health system that aimed to identify the probability of a message being delivered at time t , while several components involved may fail. Matheus et al. [58], and Santos et al. [59] propose stochastic models to analyze how failures affect the availability of electronic health systems.

Unlike our work, the studies mentioned above do not explore the performance, availability, and use of resources in conjunction. Some studies also lack performance evaluation and analysis of availability in the context of smart hospitals. Table 1 presents a related work comparison.

4 Architecture overview

Figure 1 presents an architecture of a smart hospital that uses an IoT-based health monitoring system. Sensors connected to the body collect vital patient information.

Table 1 Related work comparison

| Related work | Metric | One by one sensitivity analysis | Model | Use of component dependency |
|--------------|--|---------------------------------|-------------------|-----------------------------|
| [53] | Performance and Resources | No | Petri Net | No |
| [55] | Resources | No | No | No |
| [5] | Resources | No | Block Diagram | No |
| [4] | Performance and Resources | No | Not applied | No |
| [54] | Performance | No | Layered model | No |
| [57] | Performance and Availability | No | Petri Net and RBD | No |
| [58] | Performance | Yes | Petri Net | Yes |
| [59] | Performance and Availability | Yes | Petri Net and RBD | Yes |
| This work | Performance, Availability, and Resources | Yes | Petri Net | Yes |

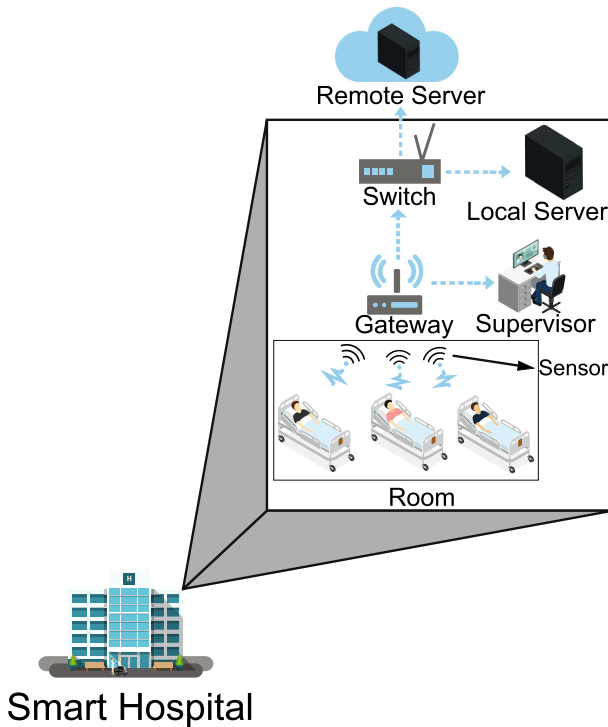


Fig. 1 Overview of the evaluated architecture

This data can also be supplemented with contextual information such as date, time, location, temperature, etc. Knowing the context allows us to identify unusual patterns and make inferences about the situation. Other sensors can also be connected to the systems for transmitting data to the medical team, such as high-resolution images.

This architecture consists of a WSN, a *gateway*, a supervisor, and two servers (local and remote). WSN is responsible for collecting biomedical and contextual signals from sensors located on patients' bodies and in the environment itself. This data can be used for treatment or diagnosis and is transmitted to the gateway via wireless communication protocols. The gateway must support distinct communication protocols and acts as a point of contact between the three elements: WSN, the supervisor, and the switch. The gateway can receive data from different subnets and perform the communication protocol conversion and provide other higher-level services, such as data aggregation, filtering, etc.

The supervisor can be a smartphone or a tablet through which the doctor or nurse monitors the status of patients. The *switch* is responsible for creating routes for the local server and remote server. The local server store the data for later searches. The remote server ensures the scalability of that data, providing more security and redundancy.

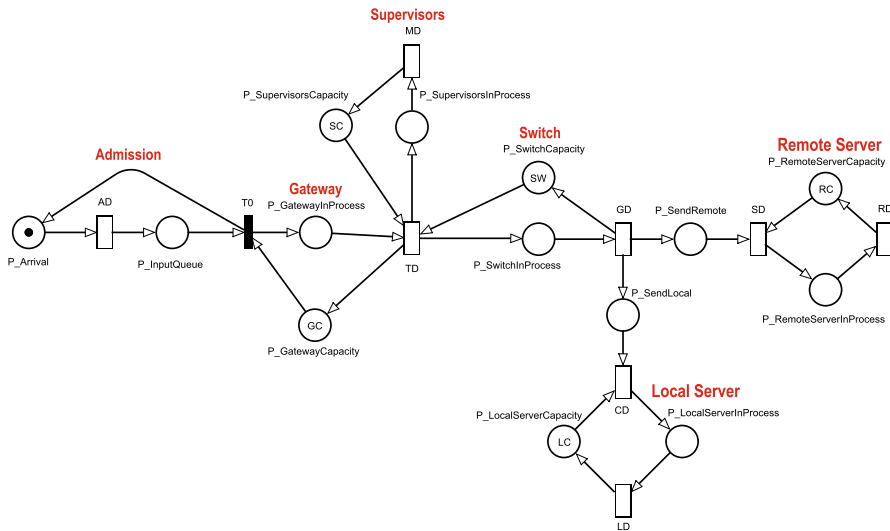


Fig. 2 SPN model for an smart hospital architecture

5 Performance evaluation of the computing architecture in a smart hospital

This section presents the SPN model for a smart hospital and performance metrics and numerical analysis to exemplify the proposal's applicability.

5.1 SPN model for smart hospitals architecture

Figure 2 represents an SPN model for the presented architecture with the following functions: (i) *Admission* that deals with the data arrival; (ii) *Gateway* that forwards the data to the supervisor server; (iii) *Supervisors* who receive the data to perform patient monitoring. Components are represented by graphs, such as places (circles), timed transitions (empty bars), and markings (small black balls). (iv) *switch* is primarily responsible for managing the route to the local server. (v) *Local Server* stores patient data for future analysis and to prevent data loss. (vi) *Remote Server* processes and stores the data in the cloud.

Components are represented by graphs such as places (circles), timed transitions (empty bars), and place markings (filled circles). The Table 2 describes all elements of the model. The timed transitions are parameterized with probability distributions. According to the literature, the system administrator shall report these distributions or perform measurements and characterizations of the system.

Given the model overview, we now describe the flow of tokens among the model components. The Admission sub-net consists of two places $P_Arrival$ and $P_InputQueue$, which represent the delay between input and acceptance of this data in the queue. Tokens generated in $P_Arrival$ represent any type of request that involves data entry to be processed or stored.

Table 2 Description of each model elements including places, transitions and marks

| Type | Element | Description | Server semantics |
|-------------------|-------------------------|---|------------------|
| Places | P_Arrival | Wait for new requests | X |
| | P_InputQueue | Wait for queue availability | X |
| | P_GatewayInProcess | Requests at the gateway queue | X |
| | P_GatewayCapacity | Gateway capacity | X |
| | P_SupervisorsInProcess | Requests on the supervisor queue | X |
| | P_SupervisorsCapacity | Supervisor server capacity | X |
| | P_SwitchInProcess | Requests on the switch queue | X |
| | P_SwitchCapacity | Switch capacity | X |
| | P_LocalServerInProcess | Requests on the local server queue | X |
| | P_LocalServerCapacity | Local server capacity | X |
| | P_RemoteServerInProcess | Requests at the remote server queue | X |
| | P_RemoteServerCapacity | Remote server capacity | X |
| Timed transitions | AD | Arrival delay between arrivals | Single Server |
| | TD | Time to gateway send requests to supervisor | Infinite Server |
| | MD | Time to process the request at the supervisor server | Infinite Server |
| | TD | Delay to the gateway send requests to supervisor and switch | Infinite Server |
| | GD | Delay to switch send requests to the local server | Infinite Server |
| | CD | Delay to requests arrive at the local server | Infinite Server |
| | LD | Delay to local server process the requests | Infinite Server |
| | SD | Delay to requests arrive to remote server | Infinite Server |
| | RD | Delay for remote server process one request | Infinite Server |
| | GC | Maximum capacity of the gateway input queue | X |
| | SC | Maximum capacity of the supervisor server | X |
| | SW | Maximum queue capacity of switch | X |
| | LC | Maximum queue capacity of local server | X |
| Place markings | RC | Remote server capacity | X |

The transition $T0$ represents the receipt of the request. Note that it is an immediate transition, having no associated delay. $T0$ fires as soon as a token exists in $P_InputQueue$ and there is at least one token in $P_GatewayCapacity$. The capacity of the gateway can be interpreted as available distribution channels, given by the GC marking.

When $T0$ fires, the gateway subnet is reached. A token is taken from $P_InputQueue$ and $P_GatewayCapacity$. A token is returned to $P_Arrival$, allowing a new trigger. A token is then added to the $P_GatewayInProcess$ location. The number of tokens in $P_GatewayInProcess$ represents the queuing of requests in the gateway. Queuing occurs when there is no available capacity to serve the newly arrived request. If there is available capacity on the supervisor server and switch (place $P_SupervisorsCapacity$ and $P_SwitchCapacity$), the TD transition is triggered to both simultaneously. TD represents the delay for the gateway to send a request to the supervisor server and for a switch.

If there is enough resource on the supervisor server, $P_SupervisorsInProcess$ will contain the number of requests in the queue for processing. The time that requests remain processing on a node depends on the MD transition. Such transitions have infinite server semantics, so each request is processed independently. It is important to note that the processing time depends significantly on the computational capacity of the supervisor server nodes. Thus, the MD transition must be configured with processing time for a single job on a specific resource type. After GD has triggered, the request follows both sides, the local server and the remote server, simultaneously. Next, it is necessary to wait for the arrival time of the data on the local server, and this time is represented by the CD transition. After the request arrives, it goes ahead to be processed.

The time that requests remain processing on the local server depends on the LD transition. LD represents the service time of the local server. Such a transition has the *infinite server* semantics, so each request is processed independently if there is a resource available in $P_LocalServerCapacity$. Again, it is essential to note that the processing time depends significantly on the computational capacity of the node (VM or container, for example) and the database used to store it. Thus, the LD transition must be configured with processing time for a single job on a specific resource type.

If there is no resource on the remote server ($P_RemoteServerCapacity$), there will be queuing in place $P_SendRemote$. If so, there is a time associated with sending a request to the remote server (transition SD) and service time on the remote server (transition RD). These transitions are also infinite server semantics, so each request is processed independently.

The time between arrivals is assigned to the transition AD . We consider that times between shots are exponentially distributed; this assumption can be modified by changing this distribution. The transition AD considers only the time the requests entered the system; that is, the network losses are not taken into account.

The proposed model allows us to evaluate several scenarios since the appraiser can configure up to 13 parameters (8 transitions and 5 markings), as shown in Table 2. Any change in one of these parameters can significantly impact the mean response time of the system and, consequently, the cost of infrastructure. The variation of the scenarios

considering a significant number of factors makes this model the main contribution of this work.

Various scenarios can be evaluated according to the definition of the parameters. The parameters must be defined based on the actual system settings. Transitions should have times similar to what you would expect if it were a real system. Capacity settings can be modified as needed, as they define the necessary settings for the system to function. The queues and discards that occur in the model create a view on which capacities must be changed in order for the system to work better. Thus, it is always important to evaluate more than one scenario to verify the best configurations so that the model and in the future the real system does not fail.

5.2 Performance metrics

This section defines metrics to evaluate the architecture of a smart hospital based on the proposed model. In this work we calculate four metrics: mean response time (*MRT*), discard probability (*Discard_Probability*), number of discarded requests against time (*Discard_Number*) and probability of resource utilization (*Utilization*<*ServerName*>).

The MRT can be obtained from the Little's Law [60] that relates the average number of requests in progress in a system (*RequestsInProgress*), the arrival rate of new requests (*ArrivalRate*), and the MRT. The arrival rate is the inverse of the arrival delay: $ArrivalRate = \frac{1}{AD}$. Little's Law requires a stable system, meaning that the arrival rate is lower than the service time. The MRT is obtained by Eq. 2.

$$MRT = RequestsInProgress \times AD \quad (2)$$

Therefore, it is also possible to calculate MRT by the equation:

$$MRT = \frac{RequestsInProgress}{ArrivalRate} \quad (3)$$

The Eq. 4 define *RequestsInProgress*. The number of requests in progress in the system is calculated by summing the number of tokens in each place representing a request in progress. In Eq. 4, *Esp(Place)* represents the statistical expected value of tokens in "Place", where $Esp(Lugar) = (\sum_{i=1}^n P(m(Lugar) = i) \times i)$. In other words, *Esp(Place)* indicates how many tokens occupy that place.

$$\begin{aligned} RequestsInProgress = & Esp(P_GatewayInProgress) \\ & + Esp(P_SupervisorsInProgress) \end{aligned} \quad (4)$$

Equation 5 defines the probability of discarding (*Discard_Probability*). There must be a token in the input queue (*P_Arrival*) to calculate the discard, and there are no more resources available in the gateway. $P(Place = n)$ computes the probability of n nodes in "Place".

$$\text{Discard_Probability} = (P((P_InputQueue = 1) \wedge (P_GatewayCapacity = 0))) \times 100 \quad (5)$$

Equation 6 defines the number of requests discarded in a given period of time T . To get the discarding number in the T period, simply multiply the discard probability by *ArrivalRate* and the time T .

$$\text{Discard_Number} = \text{Discard_Probability} \times \text{ArrivalRate} \times T \quad (6)$$

Finally, we also calculate the probability of resource utilization, where the equation is given by the number of tokens of the place corresponding to the moment of execution divided by the total capacity of those resources. We do this for each resource component of the model. Note that the capacity in question is given by the marking of the place corresponding to that resource.

The gateway utilization equation is given by:

$$\text{UtilizationGateway} = \frac{\text{Esp}(P_GatewayInProcess)}{GC} \times 100 \quad (7)$$

The equation for supervisor server utilization is given by:

$$\text{UtilizationSupervisors} = \frac{\text{Esp}(P_SupervisorsInProcess)}{SC} \times 100 \quad (8)$$

The equation for switch utilization is given by:

$$\text{UtilizationSwitch} = \frac{\text{Esp}(P_SwitchInProcess)}{SW} \times 100 \quad (9)$$

The equation for local server utilization is given by:

$$\text{UtilizationLocalServer} = \frac{\text{Esp}(P_LocalServerInProcess)}{LC} \times 100 \quad (10)$$

The equation for remote server utilization is given by:

$$\text{UtilizationRemoteServer} = \frac{\text{Esp}(P_RemoteServerInProcess)}{RC} \times 100 \quad (11)$$

5.3 Numerical analysis

This section presents eight numerical analyzes of the model. Table 3 displays the values assigned to the timed transitions and SPN model markings. Again, we vary the value of the transition corresponding to the arrival delay (AD) from 1.0 to 10.0 ms with increments of 0.5 ms. The other parameters remained fixed. All the results of the analysis with the model are presented in the graphs of Fig. 3a–e.

Table 3 Assigned values for transitions and markings of the SPN model

| Type | Element | Values |
|-------------------|---------|-------------------------------|
| Timed transitions | AD | [1.0–10.0] (Increment of 0.5) |
| | TD | 3.5 |
| | MD | 20.3 |
| | GD | 2.5 |
| | CD | 1.0 |
| | LD | 20.3 |
| | SD | 4.0 |
| | RD | 20.3 |
| Markings | GC | 8 |
| | SC | 8 |
| | SW | 8 |
| | LC | 16 |
| | RC | 16 |

Figure 3a displays the results for MRT. The MRT increases until $AD = 3.0$ ms, with $MRT = 140$ ms; after that, it decreases. There is a slight increase in $AD = 4.0$ ms, and then it starts to decay steadily. Figure 3d shows the gateway utilization, where can be noticed requests queuing with 85% of utilization in $AD = 3.0$ ms.

Figure 3b presents the discard probability of requests. At the most critical point (1.0 ms), the probability is around 75%. It is possible to note that this probability decreases as the AD increases (between 1.0 and 5.5 ms) until it stagnates to zero when the AD reaches 6.0 ms. Considering the knee ($AD = 3.0$ ms), the probability of dropping requests is less than 40%.

Figure 3c displays the number of discards in a particular time T. The period T considered in this text was 10.0 ms. As AD increases, the number of discarded data decreases. From 6.5 ms, it is possible to observe that the number of discards reaches zero. At the most critical point ($AD = 1.0$), the number of discards is approximately 760 requisitions. For such a scenario, there will be no losses if the AD exceeds 6.0 ms. In the MRT knee ($AD = 3.0$ ms), we have approximately 100 missed requests.

Figure 3d shows the utilization level of the Gateway and the Switch. For $AD = 1.0$ ms, the gateway utilization reaches 100%. With AD values above 1.5 ms, the value drops continuously, reaching values less than 10% when the AD reaches approximately 10.0 ms. As explained earlier, the switch has a lower percentage of utilization since the Gateway bottleneck does not allow requests to arrive at the same frequency on the switch. In the time interval 7.5–10.0 ms, the Gateway and the Switch have the same utilization level.

Figure 3e shows the utilization level of the supervisor server, the local server, and the remote server. The supervisor with $AD = 1.0$ ms has utilization of approximately 62%, with some variations until its downswing after the AD reaches 3.5 ms. For $AD = 10.0$ ms, its utilization reaches approximately 36%. Both the local server and the supervisor have similar behaviors. Such machines start at the utilization of 31%, and for $AD = 10.0$ ms, their utilization reaches approximately 3%. The capacity of the

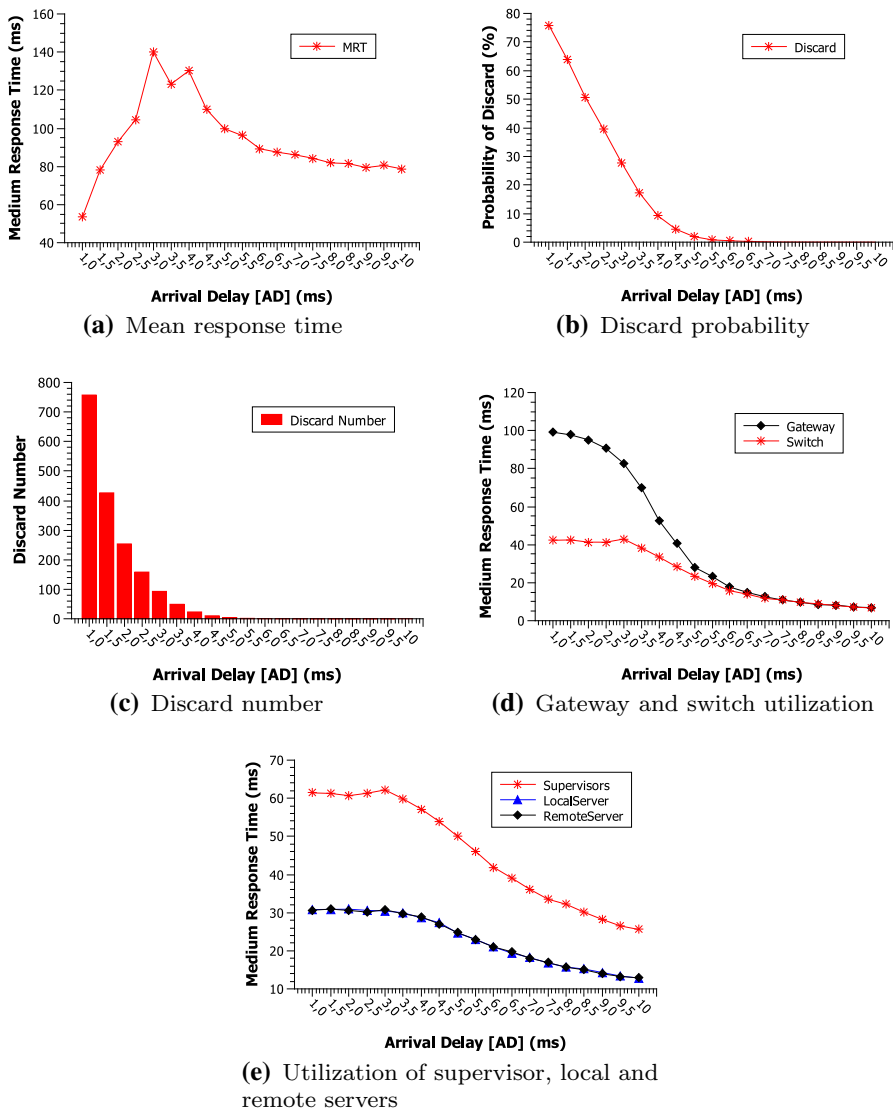


Fig. 3 Performance evaluation results of model

local server (16 cores) and the remote server (16 cores) is greater than the capacity of the supervisor server (8 cores).

5.4 Model validation

This section presents the validation of the proposed SPN model. This validation was to compare the MRT calculated by the model and the MRT collected by real experiments. Figure 4 presents an outline of the practical performed experiment. As in the proposed architecture, validation simulates the presence of various communicating components

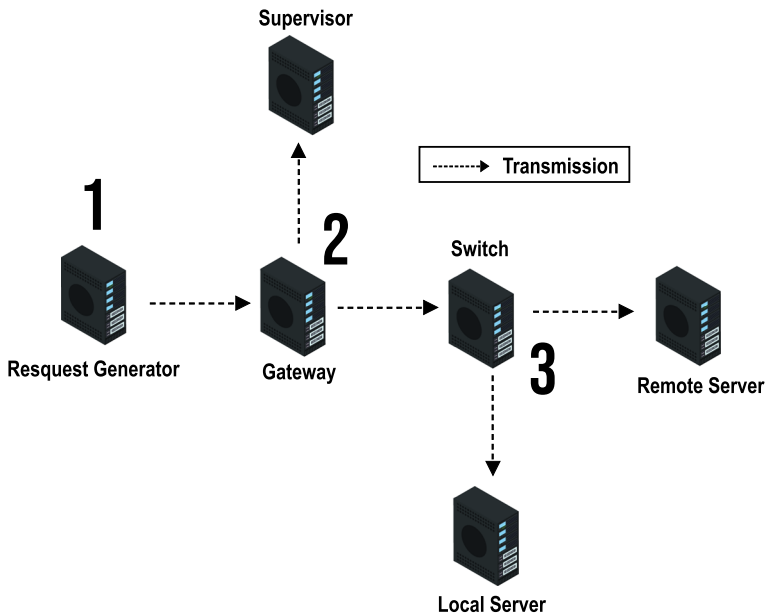


Fig. 4 Practical experiment outline to validate the model

(requests generator, gateway, switch, supervisor server, local data center, and remote data center). The figure presents three steps (1, 2 and 3) that refer to the moments of data distribution. Step 1 refers to the arrival of data to the system that occurs through the generation and sending of requests to the gateway. Step 2 refers to the transmission of data to the supervisor's machine which is responsible for monitoring the patients and the transmission to the central. Step 3 ensures the distribution of data through the switch to the local server and the remote server, so that both servers can process the data. All components process each request, so it is possible to get five requests as being transmitted to all components. A synthetic system was developed to perform the validation by generating requests obeying an exponential distribution.

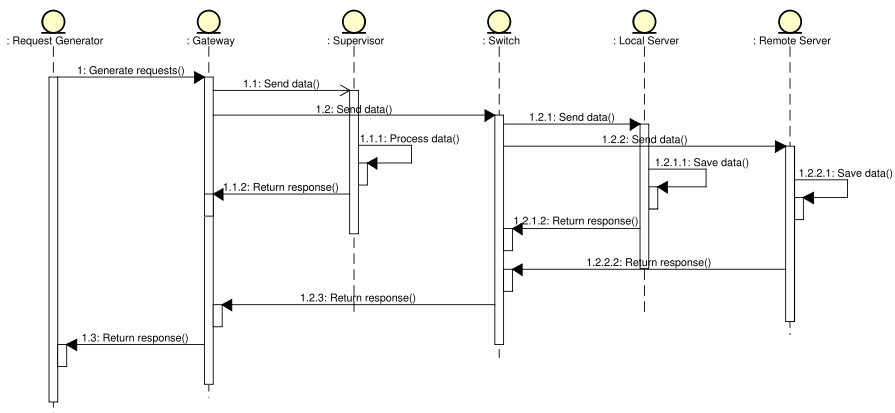
Table 4 presents the configurations of the computers used to perform the validation along with their profiles and functions. Computers can be divided into categories as follows: those responsible for forwarding data (gateway and switch); those responsible for processing data (supervisor) and those responsible for saving data (local server and remote server).

The synthetic system was a simple Java Collections API sorting algorithm (Java.util) that receives and processes a vector of numbers. However, the synthetic system can perform any service. At the beginning of the validation, an initial experiment is required to collect each step's mean times to feed the model transitions. The synthetic system generates logs during the process.

Figure 5 presents a sequence diagram for the experiment, with the steps for sending and processing the data. Each request used in the experiment consists of a file containing a 4×4 size matrix filled with random numerical values. The request generator simulates \times number of requests that send requests to the gateway within a given range

Table 4 Computers used in the experiment

| Computer configuration | Profile | Function |
|-----------------------------|-------------------|--|
| Intel Core i5 2.5 Ghz 12 GB | Request generator | Generate requests to be processed on servers |
| Intel Core i3 3.7 Ghz 8 GB | Gateway | Receive, group and forward data |
| Intel Core i3 3.7 Ghz 4 GB | Supervisor | Receive requests and process them in their cores |
| Intel Core i3 3.7 Ghz 4 GB | Switch | Receive and forward data |
| Intel Core i5 3.4 Ghz 8 GB | Local Server | Receive and save data locally |
| Intel Core i5 3.4 Ghz 8GB | Remote Server | Receive and save data remotely |

**Fig. 5** Practical experiment sequence diagram

(AD). The gateway receives the requests and forwards them to the switch and the supervisor. In the supervisor, this matrix is processed, showing the result to the user. The switch forwards the array to both local and remote servers. The array is saved in a database on both the local and remote servers.

The switch waits for confirmations from the local server and the remote server. When they arrive, the switch sends a confirmation to the gateway, sending another confirmation to the request generator, thus generating a kind of verification. The verification ensures that data has been transmitted to all components and no processing problems were encountered.

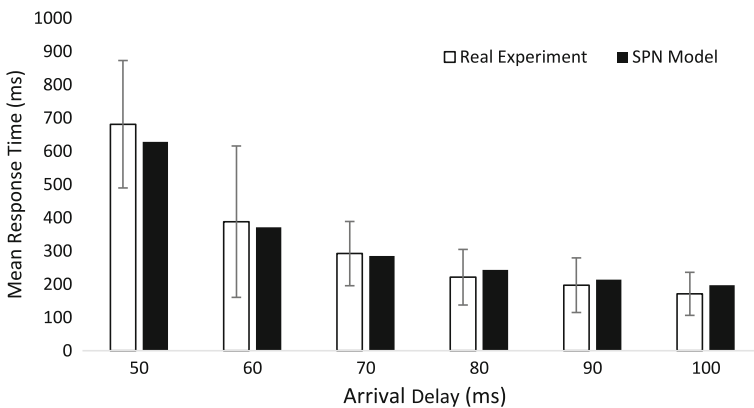
Table 5 presents the parameters that were obtained from the initial experiments mentioned above that served to align the model. The parameters used in the model that mirror the real experiment were related to the transitions and location markings. This configuration represents that each component has only four threads to handle data transmission and four threads to handle request processing.

5.4.1 Validation results

Table 6 presents the MRTs and ADs used in the experiment, where MRTs are related to each request released. Note that different quantities of requests would impact neg-

Table 5 Parameters used in the model that mirror the real experiment

| Type | Element | Values |
|-------------------|------------------------|--------------------------------|
| Timed transitions | AD | 50.0–100.0 (Increment of 10.0) |
| | TD | 4.5 |
| | MD | 4.5 |
| | GD | 2.0 |
| | CD | 2.5 |
| | SD | 80.0 |
| | LD | 42.0 |
| | RD | 42.0 |
| | P_GatewayCapacity | 8 |
| Markings | P_SupervisorsCapacity | 8 |
| | P_SwitchCapacity | 8 |
| | P_LocalServerCapacity | 16 |
| | P_RemoteServerCapacity | 16 |

**Fig. 6** Comparison between model and experiment

actively on the MRT due to long queues, increasing the time to respond to the requests. Different values can be easily configured in the models for analysis of different scenarios. For presenting an example, in this work, for each AD, 30 requests were triggered and the model was also configured to trigger only 30 requests.

Figure 6 presents the results comparing the real experiment MRTs with those of the model. For each AD, the MRT values of the experiment and model were very close.

After verifying that the samples had a normal distribution, it was decided to apply the T test of one sample to the six comparisons. We adopted the T-Test of a sample to compare the MRT generated by the model with the MRT obtained in the experiment. To check if the null hypothesis that the averages are equal, the p-value was observed. Table 7 shows the P values considering a 95% confidence interval. In all cases, the

Table 6 Times obtained in the practical experiment

| Request | MRT (ms) | | | | | |
|---------|-------------|------------|------------|------------|------------|------------|
| | AD = 100 ms | AD = 90 ms | AD = 80 ms | AD = 70 ms | AD = 60 ms | AD = 50 ms |
| 1 | 345 | 228 | 234 | 225 | 217 | 249 |
| 2 | 318 | 235 | 117 | 199 | 200 | 255 |
| 3 | 259 | 230 | 180 | 237 | 130 | 270 |
| 4 | 226 | 155 | 164 | 281 | 170 | 365 |
| 5 | 180 | 118 | 145 | 164 | 136 | 418 |
| 6 | 226 | 129 | 226 | 228 | 156 | 498 |
| 7 | 196 | 134 | 291 | 213 | 111 | 569 |
| 8 | 117 | 169 | 276 | 249 | 138 | 640 |
| 9 | 105 | 233 | 292 | 163 | 174 | 663 |
| 10 | 162 | 139 | 312 | 203 | 224 | 696 |
| 11 | 112 | 146 | 335 | 258 | 192 | 732 |
| 12 | 112 | 124 | 359 | 156 | 203 | 779 |
| 13 | 143 | 74 | 250 | 218 | 252 | 808 |
| 14 | 119 | 85 | 287 | 279 | 250 | 839 |
| 15 | 106 | 142 | 389 | 379 | 305 | 864 |
| 16 | 140 | 201 | 170 | 395 | 359 | 882 |
| 17 | 109 | 202 | 103 | 350 | 409 | 860 |
| 18 | 109 | 200 | 160 | 416 | 498 | 895 |
| 19 | 185 | 125 | 157 | 311 | 491 | 848 |
| 20 | 177 | 118 | 84 | 341 | 521 | 713 |
| 21 | 110 | 159 | 108 | 353 | 475 | 769 |
| 22 | 131 | 210 | 103 | 398 | 474 | 795 |
| 23 | 156 | 324 | 154 | 470 | 528 | 722 |
| 24 | 209 | 253 | 225 | 134 | 603 | 726 |
| 25 | 145 | 301 | 235 | 224 | 648 | 805 |
| 26 | 198 | 334 | 277 | 280 | 677 | 756 |
| 27 | 254 | 367 | 208 | 370 | 758 | 713 |
| 28 | 175 | 387 | 205 | 402 | 745 | 739 |
| 29 | 216 | 253 | 275 | 399 | 788 | 757 |
| 30 | 215 | 160 | 336 | 466 | 812 | 826 |

p-value is greater than 0.05, indicating that the results generated by the model are statistically equivalent to the experiment.

After having a generic model validated, a hospital system administrator can make use of such model to represent the target environment and evaluate different scenarios in order to help the decision making process. As our SPN model is focused on performance and availability aspects, the hospital system administrator will be able to represent and test different solutions, changing parameters of the model, in order

Table 7 T test result

| AD | P value |
|-----|---------|
| 50 | 0.135 |
| 60 | 0.677 |
| 70 | 0.683 |
| 80 | 0.186 |
| 90 | 0.301 |
| 100 | 0.066 |

to improve the service process time or to decrease the downtime metric, for instance, without additional costs during this phase.

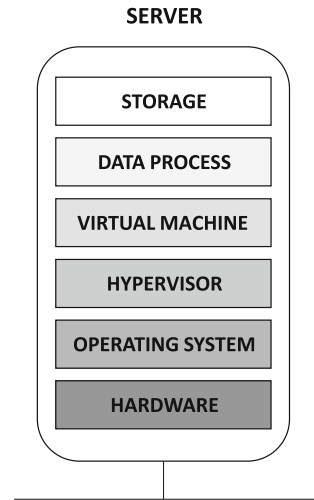
Each set of parameters represents an equipment present in the system. Therefore, based on the results provided by the models, the hospital system administrator will have a set of solutions in which they can invest in order to enhance the hospital services as a whole.

5.4.2 Threats to validity

Following, there are some points considered as threats to the validity of the validation performed on the model.

- **Synthetic data generator** A system was developed to generate and monitor the behavior of the system in order to compare it with the model’s result. Another alternative would be to use a real hospital system, however, we did not have access to this type of system. To mitigate this issue, we used to model a highly distributed architecture and used a generic matrix processing application. We believe that as the application has such a generic characteristic, it can be as representative as a specific hospital system.
- **Low-capacity machines** The machines used in the experiment were simple and conventional machines. The computers are the same as those used in the research laboratory. Most machines had only 4GB of memory. Although acceptable, the use of these computers does not match very well with the possible reality of a hospital data center. In hospitals, it is expected that powerful servers will be adopted to centralize the processing of data within the hospital itself. In addition, the used computers did not undergo an accurate formatting, but, it was only guaranteed that other software was not running during the experiment. However, as the model is configurable for any type of computational elements, simple or powerful machines can also be represented.
- **Single LAN network** The machines were, as previously mentioned, in a research laboratory using a conventional LAN network. This controlled environment may not represent a hospital network that tends to be more sophisticated and with many computers connected. However, we tried to make the network free from interference.
- **Simple benchmark application** The application used in the validation experiment is a simple matrix processing. All matrices generated were 4 × 4 in size with random

Fig. 7 Architecture of the servers based on layers



numbers, that is, of equal sizes. The transposition processing of these matrices was extremely fast given the low amount of calculations required. Perhaps if the workload was much greater, the results would be a little different.

6 Assessing the availability of a smart hospital

This section presents an SPN model to evaluate the smart hospital system's availability by considering the previously presented architecture. However, this section details the architecture a little more in terms of server' layers.

6.1 Layered architecture

Figure 7 presents the layered architecture of the servers (remote and local), including (*hardware, operating system, hypervisor, virtual machine, data process and storage*). We must highlight that the software components *data process* and *storage* are responsible for processing and storing the sensor data, respectively. The upper layers will only be available in the lower layers are working.

6.2 SPN model

Figure 8 presents a second SPN model for the architecture of a smart hospital focused on availability. The SPN model is composed of *switch (SW)*, *gateway (GA)*, *sensors (SE)*, *supervisor (SU)*, *local server (LS)* and *remote server (RS)*. The *local server* and *remote server* have several interconnected components for the system to work. The *hardware (HW)*, *operating system (OS)*, *hypervisor (HV)*, *virtual machine (VM)*, *data process (DP)* and *storage (S)*, are part of the composition of LS and RS. Each component has an MTTF (mean time to failure) and an MTTR (mean time to repair).

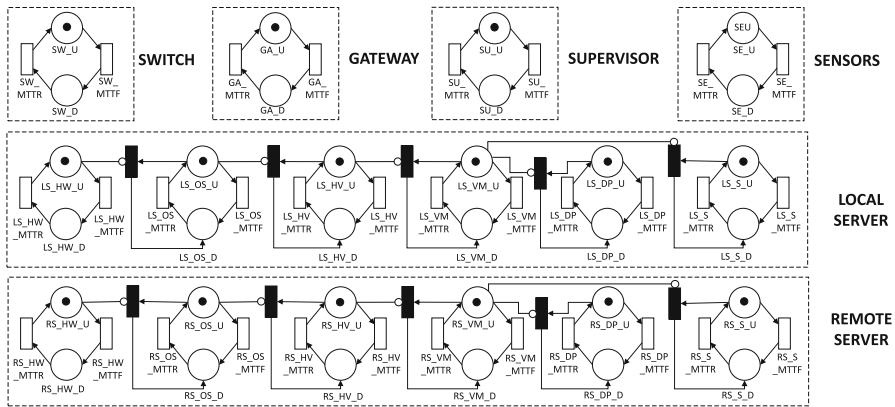


Fig. 8 SPN model of availability of a smart hospital

Immediate transitions are used to connect the components that compound the *local server* and *remote server*. These transitions ensure that if one component fails, the dependent components will also fail.

The number of active sensors is given by the number of tokens in SE_U (sensors up). The number of tokens in place SE_D (sensors down) represents the number of passive sensors. The change between the active and inactive state for each sensor is triggered by the transitions: SE_MTTT (sensors MTTT) and SE_MTTR (sensors MTTR). We consider that the gateway is working with a token at the place GA_U (gateway up) and not working with a token at place GA_D (gateway down). GA_MTTT (MTTT from gateway) and GA_MTTR (MTTR from the gateway) change the states of the gateway. Similar to gateway, we consider that switch is working when it has a token in the local SW_U (switch up); and not working with a token at SW_D (gateway down). The active/inactive changing of the switch are triggered by: SW_MTTT (MTTT from gateway) and SW_MTTR (MTTR from gateway) transitions. The supervisor is working when it has a token in place SU_U (supervisor up) and not working with a token at SU_D (supervisor down). The active/inactive changing of Supervisor are triggered by: SU_MTTT (MTTT of supervisor) and SU_MTTR (MTTR of supervisor) transitions.

The Local Server will be operational when it has tokens in the LS_DP_U (data process up) and LS_S_U (storage up) places. We consider that *local server* is not working when it has a token in one of the following locations: LS_HW_D (hardware down) or LS_OS_D (operating system down) or LS_HV_D (hypervisor down) or LS_VM_D (virtual machine down) or LS_DP_D (data process disabled) or LS_S_D (storage down). Active/inactive changes are triggered by: LS_HW_MTTT, LS_OS_MTTT, LS_HV_MTTT, LS_VM_MTTT, LS_DP_MTTT and LS_S_MTTT—for mean time to failure—and LS_HW_MTTR, LS_OS_MTTR, LS_HV_MTTR, LS_VM_MTTR, LS_DP_MTTR and LS_S_MTTR—for mean time to repair.

The Remote Server will be working when it has tokens in the locations: RS_DP_U (data process up) and RS_S_U (storage up). We consider that the Remote Server is not working when it has a token in one of the following places: RS_HW_D

Table 8 Guard conditions that restrict transition firing

| Server | Transition | Condition |
|--------|------------|------------------------|
| Local | LS_OS_MTTR | $P\{\#LS_HW_U > 0\}$ |
| | LS_HV_MTTR | $P\{\#LS_OS_U > 0\}$ |
| | LS_VM_MTTR | $P\{\#LS_HV_U > 0\}$ |
| | LS_DP_MTTR | $P\{\#LS_VM_U > 0\}$ |
| | LS_S_MTTR | $P\{\#LS_VM_U > 0\}$ |
| Remote | RS_OS_MTTR | $P\{\#RS_HW_U > 0\}$ |
| | RS_HV_MTTR | $P\{\#RS_OS_U > 0\}$ |
| | RS_VM_MTTR | $P\{\#RS_HV_U > 0\}$ |
| | RS_DP_MTTR | $P\{\#RS_VM_U > 0\}$ |
| | RS_S_MTTR | $P\{\#RS_VM_U > 0\}$ |

(hardware down) or RS_OS_D (*operating system* down) or RS_HV_D (*hypervisor* down) or RS_VM_D (*virtual machine* down) or RS_DP_D (*data process* down) or RS_S_D (*storage* down). The active/inactive changes are triggered by: RS_HW_MTTF, RS_OS_MTTF, RS_HV_MTTF, RS_VM_MTTF, RS_DP_MTTF, and RS_S_MTTF—for mean time to failure—and RS_HW_MTTR, RS_OS_MTTR, RS_HV_MTTR, RS_VM_MTTR, RS_DP_MTTR and RS_S_MTTR—for mean time to repair.

Table 8 presents guard conditions for the main transitions. Guard conditions ensure that a specific condition only triggers transitions. In our model, conditions are adopted to enable components dependency. As an example, the operating system can be repaired only if the hardware is up.

6.3 Case study

This section presents a case study based on the proposed model, including an availability analysis and a sensitivity analysis.

6.3.1 Defining scenarios and metrics

Figure 9 shows the servers in three proposed scenarios (A, B, C). Scenario A depicts the entire system's availability metric with a local and remote server in active mode. Scenario B depicts the system availability metric only with the local server running. Scenario C depicts the system availability metric only with the remote server running.

Two metrics were adopted for this evaluation: availability and downtime per year. The availability equation represents a sum of probabilities of the number of tokens in each state. Thus, P stands for probability, and # stands for the number of tokens in a given location. Table 9 presents the metrics used to obtain system availability. Downtime (D) can be obtained by $D = (1 - A) \times 8760$, where A represents the availability of the system and 8760 the number of hours in the year.

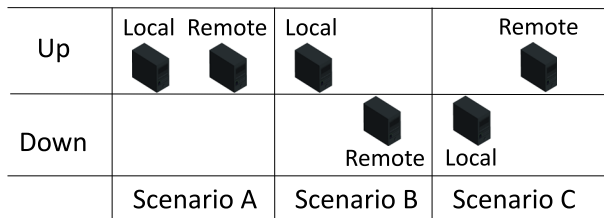


Fig. 9 Server states in each scenario

Table 9 Detailed adopted metrics

| Scenario | Availability metric |
|----------|---|
| A | $A = P\{((\#RD_HW_U > 0) \text{ AND } (\#RD_OS_U > 0) \text{ AND } (\#RD_HV_U > 0) \text{ AND } (\#RD_VM_U > 0) \text{ AND } (\#RD_DP_U > 0) \text{ AND } (\#RD_S_U > 0)) \text{ OR } ((\#LD_HW_U > 0) \text{ AND } (\#LD_OS_U > 0) \text{ AND } (\#LD_HV_U > 0) \text{ AND } (\#LD_VM_U > 0) \text{ AND } (\#LD_DP_U > 0) \text{ AND } (\#LD_S_U > 0)) \text{ AND } (\#GA_U > 0) \text{ AND } (\#SW_U > 0) \text{ AND } (SE_U = SEU) \text{ AND } (\#SUP_U > 0)\}$ |
| B | $A = P\{((\#LD_HW_U > 0) \text{ AND } (\#LD_OS_U > 0) \text{ AND } (\#LD_HV_U > 0) \text{ AND } (\#LD_VM_U > 0) \text{ AND } (\#LD_DP_U > 0) \text{ AND } (\#LD_S_U > 0)) \text{ AND } (\#GA_U > 0) \text{ AND } (\#SW_U > 0) \text{ AND } (SE_U = SEU) \text{ AND } (\#SUP_U > 0)\}$ |
| C | $A = P\{((\#RD_HW_U > 0) \text{ AND } (\#RD_OS_U > 0) \text{ AND } (\#RD_HV_U > 0) \text{ AND } (\#RD_VM_U > 0) \text{ AND } (\#RD_DP_U > 0) \text{ AND } (\#RD_S_U > 0)) \text{ AND } (\#GA_U > 0) \text{ AND } (\#SW_U > 0) \text{ AND } (SE_U = SEU) \text{ AND } (\#SUP_U > 0)\}$ |

6.3.2 Results

Some input parameters are needed to evaluate the proposed model. Component failure and repair times were extracted from [61] and [62]. Parameters for the switch and gateway parameters were extracted from the Cisco website. The adopted gateway was the Cisco 4461 router,¹ and the switch was the Cisco Catalyst 1000 Series switch for 24-port model.² Table 10 presents all adopted MTTF and MTTR values.

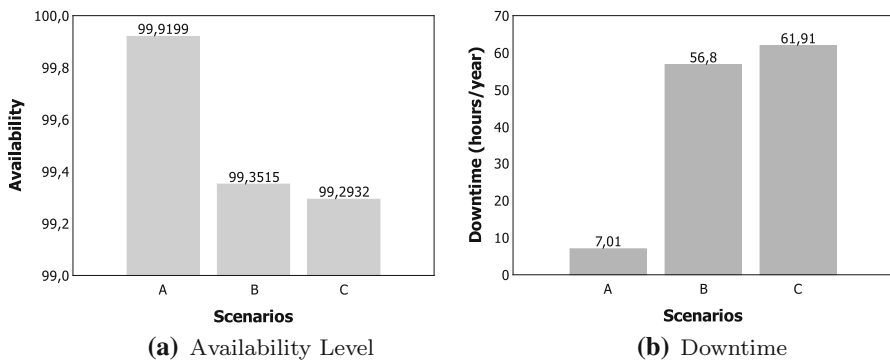
Figure 10 presents the results of availability and downtime for a smart hospital considering the three scenarios mentioned above. The results were obtained by simulation using the Mercury [63] tool. Scenario A presents the best availability (99.9199%), followed by scenario B (99.3515%) and scenario C (99.2932%). Scenario A has 7.01 h/year of downtime, while scenario B has 56.8 h/year, and scenario C, 61.91 h/year. Scenario A has the highest availability because it has a redundancy of servers (local server and remote server), so if one of the two servers fails, the other can supply the system's needs. Scenario B was more available than C, influenced by the MTTR of hardware (see Table 10). The MTTR in the local server is about 3h and 8h in the remote one.

¹ Available at: https://www.cisco.com/c/en/us/products/collateral/routers/4000-series-integrated-services-routers-isr/data_sheet-c78-732542.html.

² Available at <https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-1000-series-switches/nb-06-cat1k-ser-switch-ds-cte-en.html?oid=otren019232>.

Table 10 Input parameters (MTTR/MTTF) adopted in the case study

| Component | MTTF (h) | MTTR (h) |
|-----------------------------|-------------|--------------|
| Switch | 698220 | 8 |
| Gateway | 480770 | 8 |
| Sensors | 300000 | 1 |
| Supervisor | 44957 | 1 |
| Hardware (LS) | 4765.793684 | 3.4702988846 |
| Hardware (RS) | 8760 | 8 |
| Operational System (LS, RS) | 2800 | 1 |
| Hypervisor (LS, RS) | 2900 | 1 |
| Virtual Machine (LS, RS) | 2880 | 0.0958333333 |
| Device Manager (LS, RS) | 700 | 1 |
| Data Base (LS, RS) | 1440 | 1 |

**Fig. 10** Levels of availability and downtime in relation to the health monitoring system for the proposed scenarios

The next step was to perform the sensitivity analysis. The MTTF and MTTR were varied in intervals of 10%, until reaching 50% more and less the base value. Table 11 presents the sensitivity indexes for each scenario in descending order. Those transitions that have the most significant impact on availability appear at the top. The *data process* (DP) software component presented the highest sensitivity index in the three scenarios, meaning that such a component should be replicated to avoid system unavailability.

Considering scenario A, some additional analyses were performed. To better illustrate the influence of the parameters, Fig. 11 represents the variation in the availability based on parameter variation. Figure 11b, d, e and f show the MTTF parameters, where availability tends to increase proportionally to MTTF. Figure 11a and c show that increasing MTTR the availability decreases. The variation of the parameters provides an overview of the system's behavior. The availability is significantly influenced by the failure and repair rates of the components *data process*, *storage*, and *operating system*. Since all these components are software and are usually less expensive to replicate than hardware, it is an interesting result.

Table 11 Sensitivity indexes for all scenarios

| Base architecture | | Local server architecture | | Remote server architecture | |
|-------------------|-----------------------|---------------------------|-----------------------|----------------------------|-----------------------|
| Variable | Index | Variable | Index | Variable | Index |
| RS_DP_MTTR | 1.52×10^{-5} | LS_DP_MTTR | 2.34×10^{-3} | RS_DP_MTTR | 2.26×10^{-3} |
| LS_DP_MTTF | 1.51×10^{-5} | LS_HW_MTTF | 1.95×10^{-3} | RS_DP_MTTF | 1.88×10^{-3} |
| LS_S_MTTR | 1.11×10^{-5} | LS_DP_MTTF | 1.90×10^{-3} | RS_OS_MTTF | 1.70×10^{-3} |
| RS_DP_MTTF | 1.09×10^{-5} | LS_OS_MTTF | 1.70×10^{-3} | RS_S_MTTR | 1.52×10^{-3} |
| RS_OS_MTTF | $.06 \times 10^{-5}$ | LS_S_MTTR | 1.61×10^{-3} | RS_HV_MTTF | 1.20×10^{-3} |
| LS_OS_MTTF | 1.03×10^{-5} | LS_HV_MTTF | 1.20×10^{-3} | RS_HW_MTTF | 1.16×10^{-3} |

7 Future research directions

IoT and cloud-based have been integrated into healthcare. Such integration has led to changes in many aspects of the health industry. Nowadays, the connected physical devices that the elderly take care of themselves autonomously. In addition, IoT and cloud-based connect healthcare professionals and specialists worldwide, and they can observe and consult remotely quickly and conveniently. However, some challenges need to be faced to integrate IoT and cloud-to-cloud in healthcare in full. Following are some questions that prevents the development of IoT and cloud-to-cloud in healthcare; possible solutions for these issues are also discussed. All questions have some relationship with the evaluation of performance and availability in this work.

System Development Process - Accessibility and communication speed are two points that encourage organizations and businesses to implement IoT and cloud computing in healthcare. However, research from Cisco³ in 2017 unveiled that complete projects only occupied 26% of all the IoT and cloud computing in healthcare projects. On the other hand, 60% of projects faced difficulties at the proof-of-concept stage. CISCO pointed out that forming partnerships with other partners was a significant point for a successful project. Companies have to be careful intending to implement IoT and cloud computing in healthcare projects. They might begin with specified projects that reflect patient needs or business objectives.

Resource Management When three separate concepts (IoT, cloud computing, and healthcare) are included in one system, the resources management is the main concern [64]. For fog computing, the resource management process can be even more challenging due to the decrease in computing power and available storage compared with cloud computing. When multiple IoT devices are included in the system and collected data are being transmitted and processed in the cloud computing layer; the systems must have the ability to reduce redundant data to prevent them from using all valuable resources. Another scenario that shows the importance of resource management in IoT and cloud computing for healthcare systems is that these systems usually involve many users who share the same resources. Thus, resource management is critical to guarantee the smallest delay. As a result, when implementing IoT and cloud com-

³ <https://newsroom.cisco.com/press-release-content?articleId=1847422>.

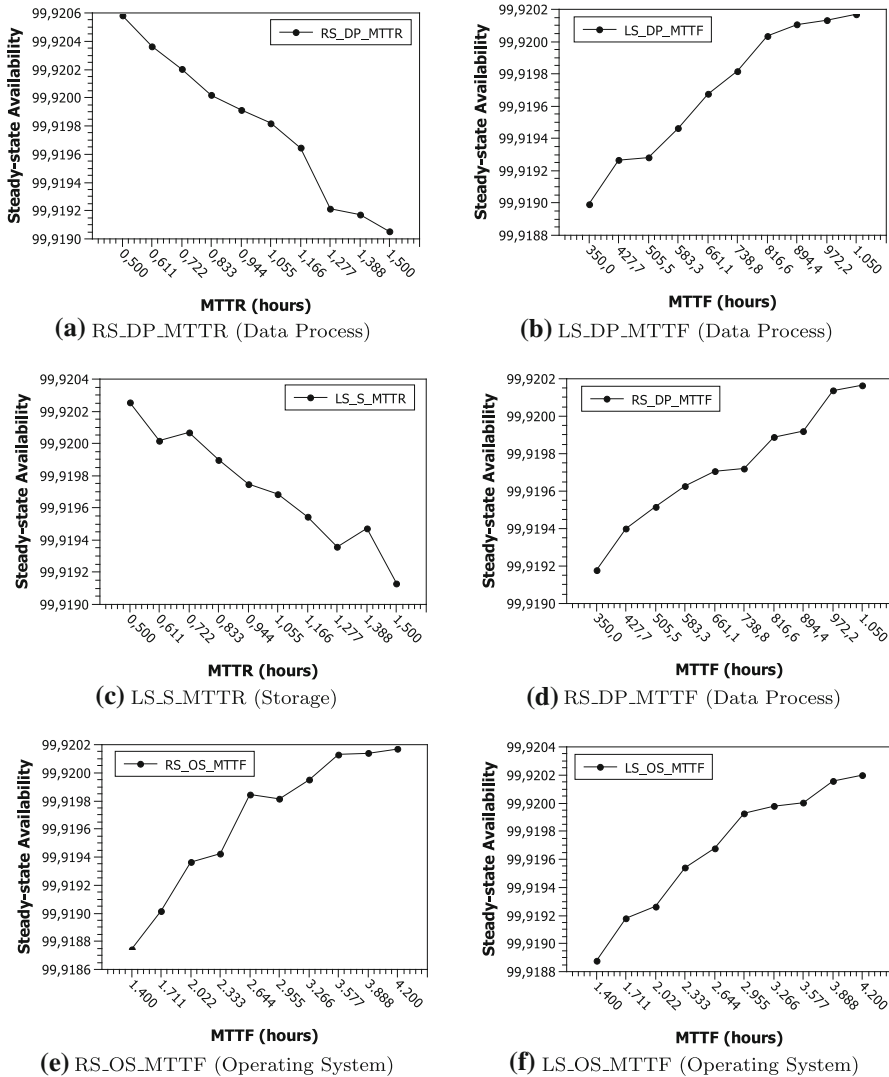


Fig. 11 Availability results by varying MTTR/MTTF

puting in the healthcare system, different factors that affect resource allocation must be carefully analyzed. Quality of service (QoS) is another important factor related to resource management because poor resource management will lead to bad QoS.

Huge Volumes of Data In recent years, a huge volume of data collected from sensors and wearable devices has caused concern in computing resources and the time-intensive process to analyze all of the data. Thus, organizations need to apply emerging technologies such as fog computing and big data to keep up with this massive influx of data.

Device Layer and Network Layer A few examples of devices from the health-care system are medical devices, connected sensors, fog nodes, gateways, and mobile devices that collect, analyze and transmit the data to the cloud. In order to prevent attacks at the device layer, security measures such as identity authentication, authorization management, whitelisting, application sandboxing, secure booting, protection of data during the collection and transmission, fault tolerance, password encryption, and secure pairing protocols must be evaluated and implemented. Moreover, the nature of IoT devices (memory, processing power, battery capacity, network range, embedded operating systems) should also be considered when security algorithms are conducted. The network layer is in charge of establishing suitable communication techniques between sensors, smart devices, fog nodes, and cloud computing that use several network protocols (such as Wi-Fi, Bluetooth, ZigBee). This layer targets attacks, including Man-in-the-Middle attack, eavesdropping, sinkhole attack, and Sybil attack. The network layer can be protected by applying secured routing mechanisms and message integrity verification techniques as well as point-to-point encryption techniques.

Cloud and Fog Layers The cloud and fog layers provide computing power and storage for data collected by the device layer. This layer is frequently attacked by Structured Query Language (SQL) Injection, Denial-of-service (DoS) attack, sniffing attack, malicious code injection, cross-site scripting (XSS), brute-force attack, phishing attack, trojan horses, and viruses. Both the cloud service providers and businesses that implement IoT and cloud computing in healthcare systems have to take adequate and efficient approaches to protect their systems from potential attacks.

8 Conclusion

This paper has adopted stochastic Petri nets to assess the performance and availability of a smart hospital system without having to invest in real equipment previously. The models are quite parametric, making it possible to calibrate the capacity of the resources, the service times, the times between failures, and the times between repairs.

The first model allows the configuration of 13 parameters, making it possible to evaluate many different scenarios. The analysis showed that the arrival rate is an essential parameter in the system. It was possible to observe the close relationship between MRT, resource utilization, and discard rate in different scenarios, especially for high arrival rates.

Three scenarios were explored considering the second model. The highest availability results were observed in scenario A, composed of server redundancy (local and remote). Such scenario—with redundancy—presented an availability of 99.9199 %, that is, 7.01 h/year of inactivity. Sensitivity analysis was also explored to identify the components that significantly impact the system's availability. Thus, the impact of different MTTF and MTTR values for each component was measured. Sensitivity analysis has shown that specific components have a higher impact on system availability. The data process software component, for example, showed a high sensitivity index in the three scenarios, meaning that such components must be replicated to avoid system unavailability.

Therefore, this work can assist system administrators in identifying the most suitable equipment in terms of cost and efficiency. As future work, we intend to perform more numerical analyses, evaluating different applications. Besides, we intend to combine both models (performance and availability) in the same model and extend the possible scenarios.


References

1. Ubirajara M (2017) Como a iot está mudando os hospitais e o mercado de saúde
2. Wong J, Goh QY, Tan Z, Lie SA, Tay YC, Ng SY, Soh CR (2020) Preparing for a covid-19 pandemic: a review of operating room outbreak response measures in a large tertiary hospital in Singapore. *Canad J Anesthesia* 1–14
3. Amir-Mohammad Rahmani, Nanda Kumar Thanigaivelan, Tuan Nguyen Gia, Jose Granados, Behailu Negash, Pasi Liljeberg, and Hannu Tenhunen. Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 826–834. IEEE, 2015
4. Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018
5. Haibin Z, Jianpeng L, Bo W, Yijie X, Jiajia L (2018) Connecting intelligent things in smart hospitals using nb-iot. *IEEE Internet Things J* 5(3):1550–1560
6. Bradon Butler. What is edge computing and how it's changing the network, 2017
7. Weisong S, Jie C, Quan Z, Youhuizi L, Lanyu X (2016) Edge computing: vision and challenges. *IEEE Internet Things J* 3(5):637–646
8. BADRINARAYANANGOPALAKRISHNAN and SRIDHAR SESHADRI. Estimating potential operational cost savings by migrating on-premises to cloud: A study using amazon tco
9. Dinh C Nguyen, Khoa D Nguyen, and Pubudu N Pathirana. A mobile cloud based iomt framework for automated health assessment and management. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6517–6520. IEEE, 2019
10. Chia-Chen C, Ya-Fen C (2013) Smart healthcare environment: design with rfid technology and performance evaluation. *J Med Biol Eng* 33(4):427–432
11. Xiao Chen, Nigel Thomas, and Michael Harrison. Performance evaluation of scheduling policies in a smart hospital environment. In *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 585–592. IEEE, 2011
12. Angel G-S, Rodriguez-Hernandez Miguel A, Victor S (2015) Evaluation of quality of service in smart-hospital communications. *J Med Imaging Health Inform* 5(8):1864–1869
13. Airton SF, Sokol K, Matheus R, Danilo O, Teresa M, Alessandro M, Paulo M (2017) Mobile cloud performance evaluation using stochastic models. *IEEE Trans Mob Comput* 17(5):1134–1147
14. Francisco Airton Silva, Matheus Rodrigues, Paulo Maciel, Sokol Kosta, and Alessandro Mei. Planning mobile cloud infrastructures using stochastic petri nets and graphic processing units. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 471–474. IEEE, 2015
15. Igor G, Laécio R, Airton SF, Anh NT, Dugki M, Jae-Woo L (2021) Surveillance system in smart cities: a dependability evaluation based on stochastic models. *Electronics* 10(8):876
16. Francisco Airton Silva, Iure Fé, and Glauber Gonçalves. Stochastic models for performance and cost analysis of a hybrid cloud and fog architecture. *The Journal of Supercomputing*, pages 1–25, 2020
17. Carlos B, Laécio R, Brena S, Iure F, Anh NT, Dugki M, Jae-Woo L, Airton SF (2021) Stochastic model driven performance and availability planning for a mobile edge computing system. *Appl Sci* 11(9):4088
18. Daniel C, Laécio R, Takako EP, Sokol K, Airton SF (2020) Edge servers placement in mobile edge computing using stochastic petri nets. *Int J Comput Sci Eng* 23(4):352–366
19. Daniel Carvalho, Laécio Rodrigues, Patricia Takako Endo, Sokol Kosta, and Francisco Airton Silva. Mobile edge computing performance evaluation using stochastic petri nets. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020

20. Lucas S, Benedito C, Iure F, Marco V, Airton SF (2021) Data processing on edge and cloud: a performance evaluation and sensitivity analysis. *J Netw Syst Manage* 29(3):1–24
21. Leoni SG, Demis G, Judith K, Djamel S, Airton SF, Takako EP, Theo L (2020) 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
22. Laécio Rodrigues, Patricia Takako Endo, and Francisco Airton Silva. Stochastic model for evaluating smart hospitals performance. In *2019 IEEE Latin-American Conference on Communications (LATIN-COM)*, pages 1–6. IEEE, 2019
23. Ngu Anne H, Po-Teng T, Manvick P, Christopher C, Walker S (2018) Smartwatch-based iot fall detection application. *Open J Internet Things (OJIOT)* 4(1):87–98
24. Juan Luis Torralbo-Muñoz, Sandra Sendra, Lorena Parra, and Jaime Lloret. Smartfridge: The intelligent system that controls your fridge. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 200–207. IEEE, 2018
25. Luigi A, Antonio I, Giacomo M (2010) The internet of things: a survey. *Comput Netw* 54(15):2787–2805
26. Gheorghe Sebestyen, Gavril Saplacan, and Lorand Krucz. Cardionet—a distributed e-health system for patients with cardio-vascular diseases. In *Workshop on medical informatics*, part of ICCP, 2010
27. Lili Y, Shuang-Hua Y, Linda P (2013) How the internet of things technology enhances emergency response operations. *Technol Forecast Soc Chang* 80(9):1854–1867
28. Gheorghe S, Adrian T, Robert A (2012) Monitoring human activity through portable devices. *Carpathian J Electron Comput Eng* 5(1)
29. Dimitrios G (2017) Smart hospitals—part 1: Designing the future
30. Peter M, Tim G, et al (2011) The nist definition of cloud computing
31. Rajkumar Buyya, Chee Shin Yeo, and Sri Kumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *2008 10th IEEE international conference on high performance computing and communications*, pages 5–13. Ieee, 2008
32. S Mohana Saranya and M Vijayalakshmi. Interactive mobile live video learning system in cloud environment. In *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 673–677. IEEE, 2011
33. Carina Nobre Gomes. *Estudo do paradigma: computação em nuvem*. PhD thesis, 2012
34. Dijiang Huang, Xinwen Zhang, Myong Kang, and Jim Luo. Mobicloud: building secure cloud framework for mobile computing and communication. In *2010 fifth IEEE international symposium on service oriented system engineering*, pages 27–34. Ieee, 2010
35. Wei-Jen W, Yue-Shan C, Win-Tsung L, Yi-Kang L (2013) Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments. *J Supercomput* 66(2):783–811
36. Michael Miller. *Cloud computing: Web-based applications that change the way you work and collaborate online*. Que publishing, 2008
37. Albert G, James H, Maltz David A, Parveen P (2008) The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Comput Commun Rev* 39(1):68–73
38. Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010
39. Ammar Awad Mutlag, Mohd Khanapi Abd Ghani, Net al Arunkumar, Mazin Abed Mohamed, and Othman Mohd. Enabling technologies for fog computing in healthcare iot systems. *Future Generation Computer Systems*, 90:62–78, 2019
40. Changqing X, Xi J, Linghe K, Chi X, Peng Z (2019) Lane scheduling around crossroads for edge computing based autonomous driving. *J Syst Architect* 95:1–8
41. Martin Holubčík, Gabriel Koman, Michal Varmus, and Milan Kubina. A model approach for the formation of synergy effects in the automotive industry with big data solutions: Application for distribution and transport service strategy. In *Smart Technology Trends in Industrial and Business Management*, pages 467–488. Springer, 2019
42. Weisong S, Jie C, Quan Z, Youhuizi L, Lanyu X (2016) Edge computing: vision and challenges. *IEEE Internet Things J* 3:637–646
43. Murata T (1989) Petri nets: properties, analysis and applications. *Proc IEEE* 77(4):541–580
44. Marsan A (1995) Modelling with generalized stochastic Petri nets. Wiley series in parallel computing. Wiley, Hoboken

45. Trivedi K (2002) Probability and statistics with reliability, queueing, and computer science applications. Wiley Interscience Publication, 2nd edn
46. Francesca C, Stefano T, Andrea S (1999) Tackling quantitatively large dimensionality problems. *Comput Phys Commun* 117(1):75–85
47. Francesca C, Stefano T, Andrea S, Marco R (2004) Sensitivity analysis in practice: a guide to assessing scientific models. Wiley, Hoboken
48. Francesca P, Keith B, Jim F, Jim H, Jonathan R, David S, Thorsten W (2016) Sensitivity analysis of environmental models: a systematic review with practical workflow. *Environ Model Softw* 79:214–232
49. Hoffman F, Gardner R (1983) Evaluation of uncertainties in environmental radiological assessment models. *Radiol Assess*
50. de Souza Matos Júnior R (2016) Identification of availability and performance bottlenecks in cloud computing systems: an approach based on hierarchical models and sensitivity analysis. PhD thesis, Federal University of Pernambuco, Center for Informatics, Graduate in Computer Science, Recife
51. Ahmad QY, Ali N, Bin ZY, Athanasios V V, Won KS (2020) The future of healthcare internet of things: a survey of emerging technologies. *IEEE Commun Surv Tutor* 22(2):1121–1167
52. Çalıř Uslu B, Okay E, Dursun E (2020) Analysis of factors affecting iot-based smart hospital design. *J Cloud Comput* 9(1):1–23
53. Soraia O, Yehia K, Moayad A, Yaser J, Thar B (2018) An edge computing based smart healthcare framework for resource management. *Sensors* 18(12):4307
54. Luca G, Pierluigi R, Fatos X (2019) An edge-stream computing infrastructure for real-time analysis of wearable sensors data. *Future Generat Comput Syst* 93:515–528
55. Min C, Wei L, Yixue H, Yongfeng Q, Iztok H (2018) Edge cognitive computing based smart healthcare system. *Future Generat Comput Syst* 86:403–411
56. Pantelopoulou A, Bourbakis N (2009) Spn-model based simulation of a wearable health monitoring system. In: 31st annual international conference of the IEEE EMBS
57. Araujo J, Silva B, Oliveira D, Maciel P (2014) Dependability evaluation of a mhealth system using a mobile cloud infrastructure. 2014 IEEE international conference on systems, man, and cybernetics (SMC), pp 1348–1353
58. Tigre M, Santos G, Lynn T, Sadok D, Kelner J, Endo P (2018) Modeling the availability of an e-health system integrated with edge, fog and cloud infrastructures. *IEEE ISCC*, pp 416–421
59. Guto Leoni Santos, Patricia Takako Endo, Matheus Felipe Ferreira da Silva Lisboa, Leylane Grazielle Ferreira da Silva, Djamel Sadok, Judith Kelner, Theo Lynn, et al. Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures. *Journal of Cloud Computing*, 7(1):16, 2018
60. John DC Little and Stephen C Graves. Little's law. In *Building intuition*, pages 81–100. Springer, 2008
61. Elton Araujo, Jamilson Dantas, Rubens Matos, Paulo Pereira, and Paulo Maciel. Dependability evaluation of an iot system: A hierarchical modelling approach. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 2121–2126. IEEE, 2019
62. Matheus Felipe Ferreira da Silva Lisboa, Guto Leoni Santos, Theo Lynn, Djamel Sadok, Judith Kelner, Patricia Takako Endo, et al. Modeling the availability of an e-health system integrated with edge, fog and cloud infrastructures. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00416–00421. IEEE, 2018
63. Bruno Silva, Rubens Matos, Gustavo Callou, Jair Figueiredo, Danilo Oliveira, Joao Ferreira, Jamilson Dantas, Aleciano Lobo, Vandi Alves, and Paulo Maciel. 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*, 2015
64. Nabil S (2014) Making use of cloud computing for healthcare provision: Opportunities and challenges. *Int J Inf Manag* 34(2):177–184

Affiliations

Laécio Rodrigues¹ · Igor Gonçalves¹ · Iure Fé³ · Patricia Takako Endo² · Francisco Airtton Silva¹ 

✉ Francisco Airtton Silva
faps@ufpi.edu.br

Laécio Rodrigues
laecio8andrade@ufpi.edu.br

Igor Gonçalves
igorcg47@ufpi.edu.br

Patricia Takako Endo
patricia.endo@upe.br

¹ Universidade Federal do Piauí (UFPI), Picos, Piauí, Brazil

² Universidade de Pernambuco (UPE), Pernambuco, Brazil

³ 3th Brazilian Army Engineering and Construction Battalion (BEC), Picos, Piauí, Brazil