Pós-Graduação em Ciência da Computação

Guto Leoni Santos

Modeling the availability and performance of the integration between edge, fog and cloud infrastructures

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Guto Leoni Santos

Modeling the availability and performance of the integration between edge, fog and cloud infrastructures

A M.Sc. Dissertation presented to the Center for Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Concentration Area: Computer Network
Advisor: Prof. Dr. Judith Kelner
Co-advisor: Prof. Dr. Patricia Takako Endo

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S237m Santos, Guto Leoni
Modeling the availability and performance of the integration between edge, fog and cloud infrastructures/ Guto Leoni Santos – 2019.
80 f.: il., fig., tab.

Orientadora: Judith Kelner.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências.

1. Redes de computadores. 2. Computação em Nuvem. 3. Computação nas Bordas. I. Kelner, Judith (orientadora). II. Título.

004.68 CDD (23. ed.) UFPE- MEI 2019-092

Guto Leoni Santos

“Modeling the availability and performance of the integration be-tween edge, fog and cloud infrastructures'”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 19/02/2019.

BANCA EXAMINADORA

_____

Prof. Dr. Nelson Souto Rosa
Centro de Informática / UFPE

_____

Prof. Dr. Theodore Gerard Lynn
Business School da Dublin City University

_____

Profa. Dra. Judith Kelner
Centro de Informática / UFPE
(**Orientadora**)

I dedicate this work to my parents, Marcos and Cicera, who have always supported me and allowed me to follow my dreams.

ACKNOWLEDGEMENTS

Firstly, I would like to thank God for giving me the chance to enter the GPRT/UFPE, to give me patience and strength in the face of difficulties and to be able to complete this work that has so much stressed me.

I would like to thank my parents (Marcos and Cicera), who have always been a source of inspiration and have always done everything I could to study and do what I love. Mom, thank you for giving me great advice always, for making me laugh at the hardest times and for taking care of me so well. Father, thank you for always being with me, for being hard at the right times, for presenting me the best songs, and for always taking care of me. I love you both so much.

I would also like to thank my girlfriend, Ana Carolina, who has always been with me during this research, which, I repeat, so much stressed me. Thank you very much for always listening to me, for giving me wonderful advice, for making me smile, for making everything easier and for continuing with myself after all the stress of this work. I love you so much.

I would like to thank my advisor, Judith Kelner, for the accompaniment, patience and advice during the work; without you, none of this would have been possible. And also to her husband, Djamel Sadok, who always gave me excellent advice and helped me a lot in the course of the research.

Special thanks to my co-advisor, Patricia Takako Endo, who, from my graduation, was present in my life, giving me strength, teaching me a lot and that helped me a lot in this work. Thank you for all the advice and tips, for being patient and for always believing in me.

I thank everyone at the GPRT who, in some way, have helped in this work, directly or indirectly. Thank you to Demis Gomes, for doing all the master's disciplines with me and for doing those stressful jobs with me. I learned a lot from you, man. Many thanks to Daniel Bezerra, Gibson Nunes, Jairo Matheus (not Matheus Vilaça), Leylane Graziele, Carolina Cani, and Pedro Dreyer for providing lots of laughter and epic moments (the order of those mentioned is not relevant).

I would like to thanks to all those who have passed in my life before this work, I have not forgotten you and you have helped me to get here (see acknowledgements of my TCC).

Finally, I would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for funding this work through grant PROEX - 0487.

"Life is never completely without its challenges".
Stan Lee (GRANATO, 2018).

ABSTRACT

The Internet of Things has the potential of transforming health systems through the collection and analysis of patient physiological data via wearable devices and sensor networks. Such systems can offer assisted living services in real-time and offer a range of multimedia-based health services. However, service downtime, particularly in the case of emergencies, can lead to adverse outcomes and in the worst case, to death. In this dissertation, we propose an e-health monitoring architecture based on sensors that relies on cloud and fog infrastructures to handle and store patient data. Furthermore, we propose stochastic models to analyze availability and performance of such systems including models to understand how failures across the Cloud-to-Thing continuum impact on e-health system availability and to identify potential bottlenecks. To feed our models with real data, we designed and built a prototype and executed performance experiments. Our results identify that the sensors and fog devices are the components that have the most significant impact on the availability of the e-health monitoring system, as a whole, in the scenarios analyzed. Our findings suggest that in order to identify the best architecture to host the e-health monitoring system, there is a trade-off between performance and delays that must be resolved.

Keywords: Cloud computing. Edge computing. Fog computing. Availability and performance evaluation. E-health systems.

RESUMO

A Internet das Coisas tem o potencial de transformar sistemas *e-health* através da coleta e análise de dados fisiológicos do paciente através de dispositivos vestíveis e redes de sensores. Tais sistemas podem oferecer serviços de monitoramento em tempo real e oferecer serviços de saúde baseados em multimídia. No entanto, o tempo de inatividade do serviço, particularmente no caso de emergências, pode levar a resultados adversos e, no pior dos casos, à morte. Nesta dissertação, foi proposta uma arquitetura de monitoramento de e-health baseada em sensores que dependem de infra-estruturas de *cloud* e *fog* para manipular e armazenar dados de pacientes. Além disso, modelos estocásticos foram propostos para analisar a disponibilidade e o desempenho de tais sistemas, incluindo modelos para entender como as falhas desde os dispositivos *edge* até a nuvem afetam a disponibilidade do sistema *e-health* e para identificar possíveis gargalos. Para alimentar os modelos com dados reais, um protótipo foi projetado e construído com o intuito de executar experimentos acerca do desempenho do sistema *e-health*. A partir dos resultados, é possível identificar que os sensores e dispositivos *fog* são, de maneira geral, os componentes que mais impactam na disponibilidade do sistema *e-health* nos cenários analisados. Além disso, é possível concluir que para identificar a melhor arquitetura para hospedar um sistema *e-health*, é necessário encontrar um equilíbrio entre o desempenho e a latência.

Palavras-chaves: Computação em Nuvem. Computação nas Bordas. Computação em Névoa. Avaliação de disponibilidade e desempenho. Sistemas de Saúde.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AT	Arrival Time
AWS	Amazon Web Services
BICSI	Building Industry Consulting Service International
BPM	Beat per Minute
CoAP	Constrained Application Protocol
CTMC	Continuous Time Markov Chain
EC2	Elastic Compute Cloud
GSPN	Generalized stochastic Petri nets
HTTP	Hypertext Transfer Protocol
IBB	interval between beats
IoT	Internet of Things
IT	Information Technologies
MQTT	Message Queuing Telemetry Transport
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
NAS	Network Attached Storage
NIST	National Institute of Standards and Technology
RBD	Reliability Block Diagrams
QoS	Quality of Service
Rest	Representational State Transfer
RFID	Radio-Frequency Identification
SAN	Storage Area Network
SPN	Stochastic Petri nets

ST	Service Time
TIA	Telecommunications Industry Association
TP	Throughput
WAN	Wide Area Network

CONTENTS

1 INTRODUCTION

Over recent years, the Internet of Things (IoT) has changed the way we interact with objects around us. The idea of the "connected world", where simple things may sense, actuate, and interact with others automatically is now possible in the emerging context of IoT (WORTMANN; FLÜCHTER, 2015). The consolidation of IoT leverages the low cost of microprocessors and other hardware that are embedded into things, and the possibility of connecting them to the Internet (AL-FUQAHA et al., 2015). Thus, these objects can potentially send and receive information to devices located anywhere in the world.

This paradigm allows the utilization of smart objects (with communication and processing capacities) in several scenarios, and consequently new applications and business models can emerge. For example, IoT is intrinsically related with smart cities (RAMIREZ et al., 2017). Here, a set of different sensors are scattered across a city collecting information about several of its aspects including for example temperature, humidity levels, traffic density, air quality, among others. Such information can be used for decision making and management improvement, in order to ensure the quality of the services offered for the city population (ZANELLA et al., 2014). Another application is in the e-health sector (ISLAM et al., 2015), where sensors can be used in a hospital to monitor, collect and process vital patient data more efficiently. In addition, placing sensors inside the home, on personal items, allows the monitoring of relevant aspects about patients health condition and could be used to trigger alarms or prompt for remote actions by appropriate assistance procedures (AMENDOLA et al., 2014). IoT can be also applied in a smart home context, where it can be used from locking and security systems for residences (HO et al., 2016) to more sophisticated residential automation systems (JACOBSSON; BOLDT; CARLSSON, 2016).

However, as IoT is generally characterized by small devices (things), its applications suffer from limited computational (storage, network, and processing) capacity, and consequential issues regarding reliability, performance, and security (BOTTA et al., 2014). The integration between IoT infrastructure with cloud computing data centers is proposed to mitigate these IoT weaknesses and also to expand the IoT coverage. According to (MARINESCU, 2017), **cloud computing** provides "unlimited" computing resources on demand, such as networks, servers, storage, applications, and services, where the customer will pay according to usage. Thus, the sensors and objects present in smart environment can send data to cloud infrastructures with the purpose of processing and storage, freeing the IoT devices from this task.

Note that the integration between IoT and cloud computing may lead to problems for delay sensitive applications. For instance, in traffic monitoring and e-health systems, where data need to be analyzed and is streamed uninterruptedly, a high delay to receive

data can compromise performance and availability of associated applications. A case in the point where this problem emerges is when the IoT devices send data to the cloud data center (via the Internet), which is often geographically distant. In this context, Edge computing was proposed to cope with this limitation. **Edge computing** gives smart end-devices local computing capability, through gateways and devices with limited computing capacity (M. FELDMAN R. et al., 2017). Thus, not all data needs to be sent to the cloud, and can be stored and processed temporarily by devices at the edge.

Although edge computing provides computational capability to the final user devices, this computational power is limited to more robust applications, which require greater computational power. Examples of such demanding applications include medical analysis based on machine learning algorithms, where classifiers are used to categorize patient data. Another example can be found in the smart cities context where large amounts of data are continuously generated and require further processing and storage (ZANELLA et al., 2014). In summary, the fog computing paradigm brings cloud characteristics to the edge of the network. **Fog computing** provides virtualized computational resources; such as storage, processing, and network, to the edge of the network, and consequently, more closer to edge devices (TSAI et al., 2017). The fog layer is located between the edge devices (sensors, gateways, and micro-controllers) and the cloud (offering"unlimited" resources of distributed data centers) (M. FELDMAN R. et al., 2017).

As a result, cloud computing appears as a mature technology capable of offering enhanced processing and storage power, while also benefiting IoT applications from scalability, high performance, and availability; while fog computing extends the cloud paradigm to the edge of the network, enabling a new breed of applications and services (VAQUERO; RODERO-MERINO, 2014; BONOMI et al., 2012). Figure 1 shows an illustrative scenario integrating edge and fog devices with a cloud computing infrastructure.

This integration improves the performance, integrity, and the availability of the IoT applications, and provides a complete and robust environment for these applications. The integrated cloud and fog infrastructures can truly act as an unlimited and optimized computational platform for IoT applications. As such the availability of cloud services and fog devices becomes more crucial in this scenario. Now, IoT applications' availability can be impacted by several additional factors, such as failures of the own IoT and fog devices, and also by interruptions on cloud data center infrastructure (including outages on Information Technologies (IT) components, and those related to cooling and power infrastructures). The impact of these failures can be exacerbated by the complexity, interdependencies and interconnectivity of the overall integrated system.

Moreover, depending on the nature of an IoT application, unavailability (even if it only lasts few seconds) can lead to great misfortunes. Let's consider the adoption of IoT systems in the e-health application to facilitate the daily monitoring of people with chronic medical conditions. IoT systems can bring many new opportunities to medical IT,

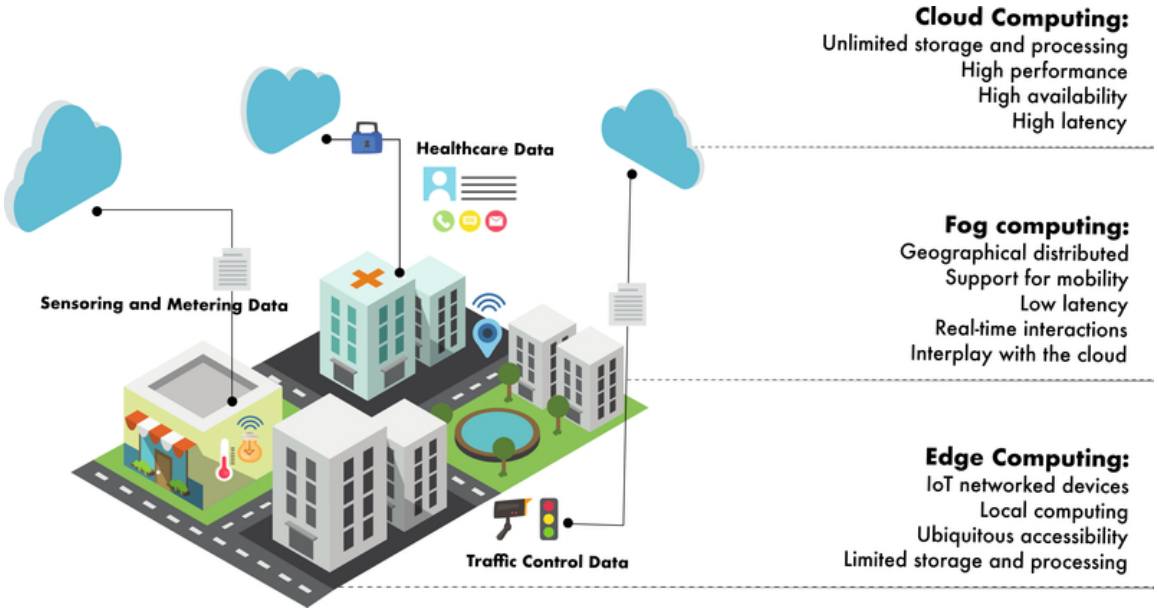


Figure 1 – IoT system integrated with edge,fog and cloud applied in a smart city scenario

allowing the collection of patients' vital data via wearable devices and sensor networks, offering multimedia-based health services, and providing assisted-living services in real-time (BOTTA et al., 2016). However, as it is considered a critical application, if the service is unavailable when a patient suffers a sudden health emergency, this could lead to a loss of life; for this type of application, every second is precious to save a life.

Despite the ease of integration between cloud and fog computing systems (usually as they use the same resources types, such as networking and computing, and share same mechanisms, such as virtualization), the integration/extension is a non-trivial one in that there exist some fundamental differences between the two. Fog paradigm has been conceived to address applications and services that do not fit well the cloud computing paradigm (BONOMI et al., 2014). For instance, authors in (BONOMI et al., 2014) state that fog computing better suits applications that require very low and predictable latency, geo-distributed applications, fast mobile applications, and large-scale distributed control systems. Due to the complexity of the integration between IoT, fog and cloud computing devices, a number of problems can arise and impact the performance and availability of services hosted on such infrastructures. Thus, mechanisms that aim to evaluate metrics of interest about IoT applications are relevant.

In this context, mathematical models are needed to evaluate some relevant metrics regarding the integration between edge, fog, and cloud; and the applications running in infrastructure. Models can be created to evaluate, for instance, the availability of applications, in a scalable way and at a low cost. In addition, they should identify failure causability, and identify the most critical infrastructure components used by an application (ENDO et al., 2017).

1.1 MOTIVATIONS

Although the integration between edge, fog, and cloud computing provides a complete and more robust architecture for IoT applications, several problems may emerge. Taking into account the security (YI; QIN; LI, 2015), replicating application data across multiple layers can represent a risk as more attack points are added for possible misuse (MOHSIN et al., 2017). Another aspect that deserves attention is application performance(YANG; LI; CHENG, 2014). The current architecture contains three separate layers where the data can be processed: edge, fog, and cloud. Depending on the type of application, edge nodes can send the data to be processed in the fog, and stored in the cloud (BORTHAKUR et al., 2017). Another possibility is that fog nodes may act as gateways that store a set of data to be sent to the cloud. In summary, there are several possibilities that have to be analyzed carefully by the application designer (RAHMANI et al., 2018).

Other issues may arise regarding the availability of the IoT application being deployed. Despite fog and cloud computing offering greater availability and resilience (BAKER et al., 2013) (LINDSTRÖM et al., 2014), they can also be viewed as vulnerabilities or potential points of failure. Thus, in addition to device/end point failure, we now also need to pay attention to fog node and cloud infrastructure failures. While cloud and fog integration is relatively well known and shares common technologies, the integration/extension with IoT is a non-trivial task, mostly due to massive device heterogeneity and service requirements. Furthermore, the availability has a direct impact on the performance of these applications, once requests cannot be processed during applications downtime. In other words, attempting to increase the availability of IoT applications by exploring and understanding the cause of failures, can result in the improvement of performance of their applications.

Modeling techniques can be used to represent states and behaviors of complex systems. Such models can be solved analytically or by simulations in order to analyze metrics of interest without the need to consider the complete system (ARDAGNA et al., 2014). In addition, it is possible to vary model parameters and configurations in a simple way and to see the impact on their output. Thus, it is possible to create models that represent IoT applications, which rely on edge, fog and cloud computing infrastructures, in order to evaluate their availability and performance.

Hence, taking into account the scenario proposed, we have the research problem: *"How can we evaluate questions about the availability and performance of e-health systems that rely on edge, fog, and cloud computing infrastructures?"*. Therefore, this dissertation proposes a set of models to analyze IoT applications, with focus on their availability and performance. These models take into account the integration of edge, fog, and cloud architecture. The interest metrics which will be evaluated in the models are availability and metrics related to performance of the applications, that are relevant aspects to the final user of these applications.

In addition, a simple prototype is proposed in order to execute some experiments with the purpose of obtaining real data to feed the stochastic models.

1.2 OBJECTIVES

Considering the motivations described previously, the main objective of this dissertations is to propose a set of models to represent IoT applications that rely on edge, fog, and cloud applications. The models proposed can cover several domain applications and can be adapted for a specific scenario according to the necessities. We propose models to evaluate the availability and performance of cloud data centers, fog, and edge devices; and integrate them. In addition, the models can be used to identify bottlenecks of IoT applications. The e-health application scenario is evaluated as case study. The specific goals of this dissertation are:

- Define an architecture to represent IoT applications in edge, fog, and cloud context;
- Propose models to evaluate the availability of IoT applications, based on edge, fog, and cloud infrastructures;
- Perform sensitivity analysis to find the critical components that influence the availability of IoT applications;
- Build a prototype with the purpose of measuring real data to feed the performance models; and
- Propose performance models and integrate them with availability models in order to evaluate the impact of downtime in performance of IoT applications.

1.3 ORGANIZATION OF THE DISSERTATION

The reminder of this document is organized as follow:

Chapter 2 describes some basic concepts about edge, fog, and cloud computing integration. The overall architecture presents a number of technologies where each one complements the weaknesses of others. This chapter also describes the basic concepts about each of the three related technology, namely, edge, fog and cloud. Next, some main concepts about availability are explored and a brief description of modeling techniques is given, with the focus on techniques used in this dissertation. Finally, we present the concepts relevant to sensitivity analysis.

Chapter 3 first presents the scenario explored in this dissertation. As a result, the models about availability of cloud data centers are presented. Next, the integration between edge, fog, and cloud models is detailed. As a result, the performance model to represent IoT applications is described. Finally, this chapter presents the prototype built to measure data from our models.

Chapter 4 shows the results obtained from cloud availability models. Results about availability and sensitivity analysis are presented. Some related works are discussed as well. Finally, some considerations are made regarding these results.

Chapter 5 describes the results obtained for the integrated scenario. First, the results obtained from our prototype are presented. In other words, the availability and sensitivity analysis results from integrated model are shown. After feeding the performance model with results obtained from the prototype, the performance results are explained. Finally, related works and some considerations about results are offered.

Chapter 6 shows the conclusions of this dissertation, the main contributions, and future works.

2 BACKGROUND

This chapter describes some basics concepts needed to understand the models proposed in this dissertation and their evaluations. Here, we present some IoT, fog and cloud computing definitions, the role of each one in a scenario, and how they can be integrated. Next, we introduce some concepts about availability of systems. In addition, some modeling techniques that can be used to represent complex systems are explained. Finally, sensitivity analysis techniques applied to identify critical components in systems are presented.

2.1 EDGE, FOG, AND CLOUD COMPUTING INTEGRATION

2.1.1 Internet of Things (IoT)

The first IoT definition was mentioned in 1999, by Kevin Ashton, relating it with Radio-Frequency Identification (RFID) systems. In summary, IoT was associated to identification systems where objects are identifiable uniquely using RFID (LI; XU; ZHAO, 2015). However, this concept has evolved over the years, new technologies and communication protocols have been incorporated into IoT; new types of objects have been connected, and new scenarios and business models emerged. Currently, IoT is capable of interconnecting billions or trillions of heterogeneous objects through the Internet (AL-FUQAHA et al., 2015). Authors in (BUYYA; DASTJERDI, 2016) define that the IoT paradigm

> "*promises to make "things" including consumer electronic devices or home appliances, such as medical devices, fridge, cameras, and sensors, part of the Internet environment. This paradigm opens the doors to new innovations that will build novel type of interactions among things and humans, and enables the realization of smart cities, infrastructures, and services, for enhancing the quality of life and utilization of resources.*"

Therefore, IoT gives to simple objects capacity of see, hear, think, and perform tasks by having them "talk" together, with the purpose to share relevant information and take coordinated decisions (AL-FUQAHA et al., 2015). These capabilities allow the emergence of sizable IoT scenarios and applications. IoT supports situated sensing, i.e. the ability to collect information about natural phenomena, or user habits and offer these as a high level tailored service. These applications can be classified in three major domains (BORGIA, 2014):

- **Industrial domain**: monitoring industrial plants, farm registration management, shopping operation, real-time vehicle diagnostic, etc.
- **Smart city domain**: road condition monitoring, traffic management, energy management, sustainable mobility, comfortable living, etc.

- **Health well-being domain**: medical equipment tracking, smart hospital services, elderly assistance, (remote) monitoring medical parameters, etc.

Nevertheless, the IoT utilization may suffer some limitations and problems. Over the years, several hardware platforms were developed to connect sensors (including Arduino, WiSense, BeagleBone, etc). There is clearly the presence of a wide range of heterogeneous devices. Consequently, many software platforms are utilized to provide IoT functionality. This heterogeneity (both software and hardware) makes it difficult to seamlessly implement IoT in several scenarios (AL-FUQAHA et al., 2015). In addition, the hardware used by these objects is very limited in terms of the supported computational resources (CPU, memory, and storage). This is understandable as such devices need to be inexpensive to be used on a large scale. According to Cisco, as many as 20 billion objects are expected to see global deployment by the end of 2020 (EVANS, 2011). This clearly will generate huge amounts of data which cannot be processed or stored on IoT devices because of their limiting capabilities. In this context, cloud computing provides the computational capacity required for IoT devices.

2.1.2 Cloud computing

Currently, essential services (such as water, energy, gas and telephony) are provided in "unlimited" form such that customers can easily gain access to these utilities and pay according to their usage. Under the cloud computing paradigm, computing has become a similar utility service. Cloud computing provides simple access to computational resources from anywhere in the internet (BUYYA et al., 2009). According to National Institute of Standards and Technology (NIST) (MELL; GRANCE et al., 2011),

> "*cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*"

For the cloud clients, this paradigm is very attractive because it delivers "unlimited" computing resources under a pay-as-you-go business model. Thus, customers benefit from reduced costs associated with infrastructure and the management of equipment and are free to concentrate on their business (BUYYA et al., 2009) (TOOSI; CALHEIROS; BUYYA, 2014).

Cloud computing and IoT can be seen as two complementary technologies, where cloud provides to IoT applications the computational resources (e.g., processing and storage) needed to devices (BOTTA et al., 2014). For instance, in smart city scenarios that generate a huge data quantity, this data can be sent to a cloud infrastructure for storage for posterior analysis using sufficient computational power.

The cloud computing essential characteristics are defined in (MELL; GRANCE et al., 2011):

- **On-demand self-service:** the computational resources are provided according with customer demand, without the need for human interaction and automatically.
- **Broad network access:** the resources are available over the network and can be accessed by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- **Resource pooling:** a provider's computing resources are made available in the form of a pool, where these resources (physical or virtual) are allocated and reallocated among customers according to demand.
- **Rapid elasticity:** the capabilities can be released elastically and provisioned, according to demand. For customers, resources appear to be unlimited and can be appropriated any time and in any quantity.
- **Measured service:** The utilization of resources can be monitored, controlled, and reported by the customers, maintaining the transparency of for both the provider and consumer of the utilized service.

Overall, the infrastructure of cloud computing is formed by geo-distributed data centers (TOOSI; CALHEIROS; BUYYA, 2014). A common data center can be composed into three main subsystems: power subsystem, cooling subsystem, and IT subsystem. The power subsystem is made up of equipment needed to power the other data center equipment in an uninterrupted manner. The cooling subsystem is responsible for removing the heat generated by the data center equipment in order to avoid damages of IT equipment. Finally, IT equipment is made up of servers, storage units and network equipment, where applications and cloud services execute (BARROSO; CLIDARAS; HÖLZLE, 2013). Figure 2 illustrates the relationship of these subsystems. In this dissertation we focus on the **TI!** (**TI!**) infrastructure, since it is the main subsystem of the data center and where applications and services are executed.

Currently, there are some standards that define the fundamental aspects, recommendations, and best practices to plan and build data centers, taking into account their availability. The ANSI/BICSI-002 (DESIGN, 2011), defined by the Building Industry Consulting Service International (BICSI), and the TIA-942 (ASSOCIATION et al., 2006), elaborated by the Telecommunications Industry Association (TIA), are the main existing data center standards. Both standards classify data centers according to a number of levels with increasing availability: TIA-942 refers to them as tiers (from I to IV), while BICSI-002 defines these as classes (ranging from 0 to 5). As the two first classes of BICSI-002 are compatible with the first tier of TIA-942, reaching similar availability, in this dissertation

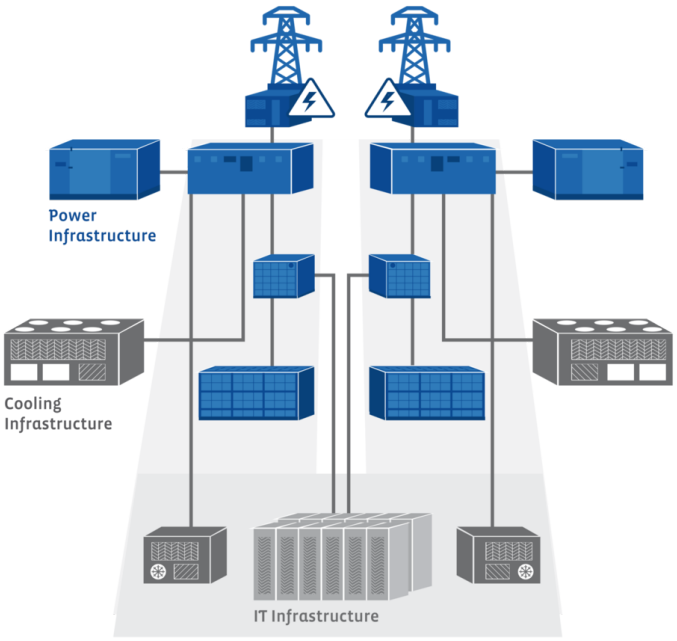


Figure 2 – Subsystems of a generic data center

we will follow the architecture based on TIA's tiers, from I to IV. In this dissertation, we consider the TIA-942 to standard the cloud data centers.

Basically under the TIA-942 standard, a tier is different from another one due to the number of used redundant components (i.e., N means no redundancy; and $N + 1$ means component redundancy) and distribution paths (i.e. single or multiple paths that may be active or passive). Tier classification ranges from I to IV. Higher tiers inherit requirements from lower ones and are less susceptible to system disruptions (Tier II), may avoid system disruptions (Tier III), or are fault tolerant (Tier IV). Higher tiers provide greater availability, which results in higher costs and operational complexities. Therefore, the tier selection depends on the business requirements, such as minimum service availability, deployment costs, and downtime financial consequences.

Even though cloud computing provides the computational resources needed to IoT application, this integration is not trivial, and some problems may emerge. Problems related to security, privacy, legal and social aspects, and data generation are the most critical challenges in this integration (BOTTA et al., 2016).

Another important point that is also impacted the hosted applications performance. Applications with low latency requirements (e.g., online games, streaming applications, augmented reality applications) may have performance affected by the delay caused by the communication with the data center through the Internet (since data centers are, eventually, geographically distant from IoT devices) (BONOMI et al., 2012; BOTTA et al., 2016). This led to the emergence of the of the idea of bringing more computational capabilities to devices located closer to user devices. As a result, the edge computing paradigm was born to bridge this gap between IoT devices and cloud data centers.

2.1.3 Edge computing

With the proliferation of IoT applications, a huge amount of data is generated at the network edge. Not all data can be sent to the cloud especially in the case of applications sensitive to delay. As a result, local processing is necessary. For example, in the autonomous vehicles scenario, around a gigabit data will be generated by a car every second, and such data needs to be processed in real-time to make correct decisions (SHI et al., 2016). Cloud computing cannot support bandwidth demanding IoT application as it becomes a bottleneck over existing internet wide area network links. With the edge computing paradigm, data processing occurs in part at the network edge, rather than completely in the cloud (SHI; DUSTDAR, 2016).

According to NIST (M. FELDMAN R. et al., 2017), "edge is the network layer encompassing the smart end devices and their users to provide, for example, local computing capability on a sensor, metering or some other devices that are network-accessible". In this dissertation, we are concerned with IoT devices located at the edge of the network (for instance, sensors connected to microcontrollers) as end devices taking part in edge computing.

Although edge devices offer a computing capability to local edge devices, normally these devices are individually very limited. Edge devices have limited computational capabilities (e.g., CPU, memory), and when these devices are mobile, the battery factor may limit further these (MAO; ZHANG; LETAIEF, 2016) (LUAN et al., 2015). While some works aim to optimize resources of edge networks (SARDELLITTI; SCUTARI; BARBAROSSA, 2015) (YOU et al., 2017), a new paradigm emerges with the purpose of bringing the cloud computing characteristics to the edge network: fog computing.

2.1.4 Fog computing

With the development of smart homes, smart cities, connected vehicular, smart agriculture, among others, the IoT has received a lot of attention in the last years, and is considered a pillar of the future of the Internet. IoT applications relies on the support of cloud computing, providing the infrastructure (hardware and software) needed by smart/edge devices. However, IoT applications usually require geo-distribution, location-awareness and low latency (YI; LI; LI, 2015). To solve this problem, fog computing has emerged as a practical solution to enable the smooth convergence between cloud and edge devices for content delivery and real-time data processing (LUAN et al., 2015).

According to NIST (IORGA et al., 2018):

> *Fog computing is a layered model for enabling ubiquitous access to a shared continuum of scalable computing resources. The model facilitates the deployment of distributed, latency-aware applications and services, and consists of fog nodes (physical or virtual), residing between smart end-devices and centralized (cloud) services. The fog nodes are context*

> *aware and support a common data management and communication system. They can be organized in clusters - either vertically (to support isolation), horizontally (to support federation), or relative to fog nodes' latency-distance to the smart end-devices.*

Similarly to cloud computing, fog computing is a virtualized platform that provides computation services, both hardware and software, between smart devices and traditional cloud data centers (YI; LI; LI, 2015) (BONOMI et al., 2012).

The fog nodes are intermediate compute elements situated between smart end-devices and cloud. These nodes may be either physical or virtual elements. Though fog is an extension of cloud, not all nodes are resource-rich. Some of these objects have limited computation power, memory, and storage (e.g., smart TVs/set-top-boxes, gateways, switches, routers, notebooks, and end-devices) (M. FELDMAN R. et al., 2017) (YI et al., 2015).

Frequently, fog and edge computing are considered interchangeable in some works. However, there are key differences between them. Edge computing focus is towards the things side and is defined by the exclusion of fog and cloud. While fog works closely with the cloud, and focuses more on the infrastructure side. In addition, fog devices tend to be limited devices, while at the same time it also addresses storage, networking, and data processing (M. FELDMAN R. et al., 2017) (SHI et al., 2016).

Despite fog and cloud computing offer greater availability and resilience, once the architecture becomes more complex, the number of vulnerability points and points of failures increases. The availability aspect of the application can be impacted. Some IoT applications are highly sensitive to downtime. For example, medical applications, where vital data of patients are evaluated, need to work in an uninterrupted manner. A single failure that can be translated into a downtime, even if for a few seconds, may have serious consequences.

Therefore, the availability study of the architecture is necessary to understand how to keep these IoT applications working as well correctly as possible.

2.2 AVAILABILITY CONCEPTS

To evaluate the availability of systems is an important aspect for those who offer services. There is a need to improve this constantly (ENDO et al., 2017). A study of which factors impact availability, and understanding how to mitigate such factors is important to increase availability at a low cost.

However, it is worth emphasizing that availability is a concept that is part of a larger one: that of the dependability. The dependability of a system is the ability of that system to reliably provide a given service (ANDRADE et al., 2017). Dependability is a concept that encompasses six mains non-functional requirements (AVIZIENIS et al., 2001):

- **Reliability**: the probability that a system will be provided a service without interruption, until a certain moment;

- **Availability**: the capacity of the system readiness for provide correct service;
- **Safety**: is the absence of catastrophic consequences for end users and the environment in which the system is running;
- **Confidentiality**: is the absence of unauthorized disclosure or information leakage;
- **Integrity**: capacity of the system does not undergo improper changes in its working;
- **Maintainability**: capacity to undergo repairs and modifications.

In this dissertation, we focus only on availability analysis. The availability can be defined as a measure of delivery of a correct service over time, even if the system does not achieve its full capacity (ANDRADE et al., 2017) (FRANCO; BARBOSA; ZENHA-RELA, 2014). Unlike reliability, which evaluates the probability of having a system working until a first failure, availability takes into account the repair of system (maintainability concept). The calculation of availability is given by Equation 2.1. Service uptime means that the system is running and providing the service normally. Service total time represents the service uptime plus the downtime of service. This downtime can represent time to detect a failure and the repair time (ENDO et al., 2017).

$$serviceAvailability = \frac{serviceUptime}{serviceTotalTime} \tag{2.1}$$

The time that the system works normally (uptime service) also is called time to failure. So, given a period of time, e.g., one day, one week, one year; it is possible to calculate the Mean Time to Failure (MTTF). This value is equivalent to uptime service in Equation 2.1. In addition, the total service can be defined as the sum between MTTF and the Mean Time to Repair (MTTR). MTTR is a average of the time to repair the system, and consequently, the time that the system is inoperable (ENDO et al., 2017) (MELO et al., 2017). Equation 2.2 shows the calculus of availability using MTTF and MTTR concepts.

$$serviceAvailability = \frac{serviceUptime}{serviceTotalTime} = \frac{MTTF}{MTTF + MTTR} \tag{2.2}$$

Thus, the value of availability is given by a probability value, ranging between zero and one. This value can be expressed as number of nines (calculated through Equation 2.3). For example, if in a period of one year, a system presents 0.9994% of availability, this system has 3.2218 nines of availability, and was available 0.9994% of time and unavailable the remaining 0.0006 of the time.

$$numberOfNines = 2 - \log(1 - availability) \tag{2.3}$$

To estimate the availability of a system is a hard task, due to the complexity it can present. A system can be composed of several components, and estimate its availability can take a long time in terms of processing. In addition, changes to its configuration or adding more components onto a system can be hard to manually to track their effect on availability (SIEWIOREK; SWARZ, 2017). Thus, several modeling techniques have arisen to assist in assessing the availability of various types of systems.

2.3 MODELING TECHNIQUES

Understanding the behavior of given aspects of complex systems has always been of interest to researchers. Several areas require mechanisms for easy representation and evaluation to under-stand and verify their various aspects. In such situations, it is possible to build a model representation of the real systems, which receives as input some basic system parameters and calculates the behavior of the system under given conditions. Since the model represents the behavior of the real system, it is possible to observe how the modification of input parameters changes the state of the real system (KOROLYUK; KOROLYUK, 2012). Therefore, theses analytical models are a powerful tool for analyzing the behavior of complex systems.

Many of the existing modeling techniques have their foundations in probability theory. For example, in the analysis of the execution of an algorithm, under some input parameters, the algorithm may take longer to run, while under other conditions, it may be quicker to execute. For many problems, probabilistic analysis of the algorithm is likely to be more useful (TRIVEDI, 2008).

In this dissertation, we use modelling approaches to evaluate IoT applications. Metrics, such as availability or those related to performance, can be estimated with theses modelling techniques. These approaches can be classified as: non-combinatorial or state-space models, and combinatorial or non-state-space models (TRIVEDI; SATHAYE; RAMANI, 2005). These approaches differ according to their modelling power, ease of use, and system complexity, and will be explained below.

- **Non-combinatorial or state space models**: these models create the state space structure that represents all states that a system can reach in its operation (e.g., failures and repair operations). They are efficient for representing and analyzing system operations. Markov chains, markov regenerative processes, semi-markov processes, petri nets, Stochastic Petri nets (SPN), and stochastic reward nets are examples of state space models (DANTAS et al., 2012). However, when the system has a large number of states, these approaches face a problem called state explosion, making the built model difficult to solve. State explosion happens when a system has a very large number of states, and their model becomes more complex to evaluate, taking a long of time to solve (KUNTZ, 2006).

- **Combinatorial or non-state-space models**: These techniques allow the representation of the relationship between components of systems and subsystems. As they are not based on the states of the system, they do not suffer from the problem of state explosion, and can represent systems with hundreds to thousands of components. Fault tree, Reliability Block Diagrams (RBD), and reliability graphs are examples of combinatorial models. Their disadvantage is that they do not allow the modeling of complex system behaviors (e.g., concurrent processes, simultaneous faults, etc) (TRIVEDI; SATHAYE; RAMANI, 2005) (ENDO et al., 2017).

In this dissertation, we use RBD and SPN to represent and analyze our scenarios. Their combination captures the inter-dependencies of components and complex behaviors of our system, and will be explained next.

2.3.1 Reliability diagram blocks (RBD)

RBD is a graphical representation of a system's success logic using block structures (VERMA; SRIVIDYA; KARANKI, 2010). RBD can be evaluated using analytical methods to obtain system reliability and availability.

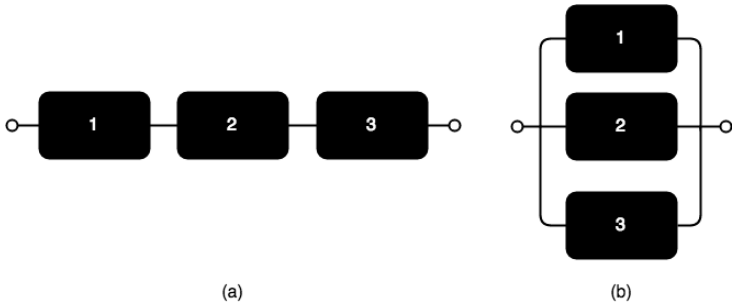


Figure 3 – RBD configurations

When all components of the system are strictly required for its operation, a failure of one of them causes the overall system to fail. In this case, the components are arranged in series, as shown in Figure 3.a. On the other hand, if the isolated failure of one component does not interrupt or shut down the system (i.e., the component is redundant), the blocks are disposed in parallel, as shown in Figure 3.b.

The availability can be defined as service uptime over total service time, where total time is described as the sum of service uptime and service downtime. These concepts can be associated with the average behavior of the system for the purpose of availability calculation.

As shown in Section 2.2, the availability of each component, A_x, is calculated by division of the MTTF and the Mean Time Between Failures (MTBF) of each component

(Equation 2.4). The MTBF also is defined as the sum of MTTF and MTTR, indicating the time between the detection of a failure and the detection of the next failure.

$$A_x = \frac{MTTF_x}{MTBF_x} = \frac{MTTF_x}{MTTF_x + MTTR_x} \tag{2.4}$$

In this way, the availability of the overall system, in case of series combination, A_s, can be calculated via Equation 2.5.

$$A_s = \prod_{x=0}^{N} A_x \tag{2.5}$$

While the availability of overall system with components arranged in parallel can be calculated using the Equation 2.6.

$$A_p = 1 - \prod_{x=0}^{N} A_x \tag{2.6}$$

Considering that reliability, $R(t)$, is defined by Eq. 2.7, the reliability of the overall system where the component are arranged in series, $R_s(t)$, can be calculated using Equation 2.8. Nonetheless, when the components of system are arranged in parallel, the reliability, $R_p(t)$, can be calculated using Equation 2.9.

$$R_x(t) = \mathrm{e}^{-\lambda_x t} \tag{2.7}$$

Where, λ depicts the failure rate.

$$R_s(t) = \prod_{x=0}^{N} R_x(t) = \prod_{x=1}^{N} \mathrm{e}^{-\lambda_x t} \tag{2.8}$$

$$R_p(t) = 1 - \prod_{x=0}^{N} (R_x(t)) = 1 - \prod_{x=1}^{N} (\mathrm{e}^{-\lambda_x t}) \tag{2.9}$$

Once the reliability is calculated, it is possible to obtain the MTTF of the system until a given instant t using the follow equation (HØYLAND; RAUSAND, 2009):

$$MTTF = \int_0^{\infty} R(t)dt \tag{2.10}$$

Finally, the MTTR of the overall system can be calculated by placing estimated availability and MTTF in the Eq. 2.4.

RBDs are commonly used due to their simplicity, but they are not suitable to model behavioral aspects of a system (DANTAS et al., 2015). Therefore, Petri Nets can be used in conjunction with RBDs in order to have a set of comprehensive models addressing aspects of complex systems.

2.3.2 Petri nets

Petri nets are a tool to model dynamic systems with features such as concurrency, synchronization, communication mechanisms, deterministic, mutual exclusion, and conflict (ANDRADE et al., 2017).

Formally, a Petri net N can be defined as a 5-tuple $N = (P, T, I, O, M_0)$, where (MARSAN, 1988)(STOJIC, 2017):

- $\boldsymbol{P}$ is a finite, nonempty set of places $P = (p_1, p_2, ..., p_m)$,
- $\boldsymbol{T}$ is a finite, nonempty set of transitions $T = (t_1, t_1, ..., t_m)$,
- $\boldsymbol{I}$ a set of input arcs $I \subset P \times T$,
- $\boldsymbol{O}$ a set of input arcs $O \subset P \times T$,
- $\boldsymbol{M_0}$ is a initial marking (set of tokes assigned to places) $(m_{01}, m_{02}, ..., m_{0n})$ of Petri net.

A set of tokens assigned to places is called markup, and a markup represents a state of the system. Transitions represent actions, which change from one state of the system to another. The arcs connect the places to the transitions and vice versa, and they have weights (GUIMARAES; MACIEL; JUNIOR, 2015). Petri nets also can have a weight function w that assigns a positive integer into arcs of the net (ZUBEREK, 1991).

Considering a place $P1$ connected to a transition $T1$ by an arc with weight w, the transition $T1$ will be enabled if the number of tokens in $P1$ is equal to w. Transitions may have guard functions, which are additional conditions for triggering transitions. In original Petri nets, there are only immediate transitions, that are represented by black rectangles. This type of transition fires at the moment they are enabled.

A variation of the traditional Petri net was proposed with aim to introduce "*temporal specifications such that the future evolution of the model, given the present marking, is independent of the marking history*"(MARSAN, 1988). For this, the sojourn times in markings must be random variables. Now, the an array $\Lambda = (\lambda_1, \lambda_n, ..., \lambda_n)$ is added into the 6-tuple with firing rates associated with the stochastic transitions. A variation of SPN, called Generalized stochastic Petri nets (GSPN) was proposed adding the concept of immediate transition, where it fires immediately after being enabled (STOJIC, 2017). However, in most cases, the analysis of GSPN models involves this reduction to a SPN model, then only the SPN gained importance, and we will use this term until the end of the dissertation.

The stochastic transitions can have two different firing semantics: single server and infinite server. In a single-server semantic, the firing of a transition are sequential and after each firing, a new delay is sampled in the case of the same transition activated again. In

infinite server semantics, a set of tokens present in one place is processed in parallel, each with its own delay (BALSAMO; MARIN; STOJIC, 2015).

These SPN components are illustrated in Figure 4. A SPN can be converted to a Continuous Time Markov Chain (CTMC) and resolved using numerical or analytic methods, or through simulation. For more information on SPN see (BAUSE; KRITZINGER, 1996).

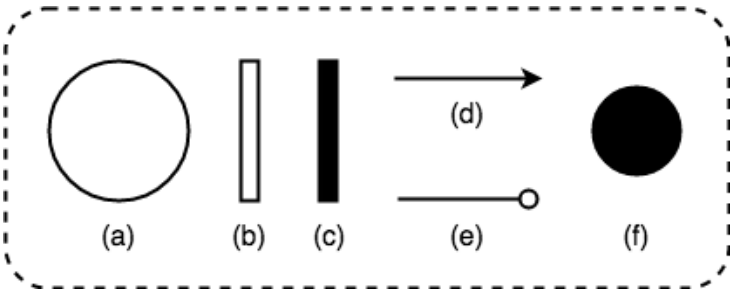


Figure 4 – SPN components (a)Places, (b) sthocastic transtions, (c) immediate transitions, (d) directed arcs, (e) inhibitor arcs, (f) token

We use RBD to model some dependencies present in servers hosted on cloud and in the fog devices due to their simplicity and efficiency of computation (DANTAS et al., 2015). However, it can not be used to represent systems with more complex behavior. Thus, we use SPN to represent cloud, fog, and edge infrastructures in order to evaluate aspects as performance and availability of IoT applications due to its modeling power to represent dynamic systems with complex behaviors such as concurrency, synchronization, and randomness (XIE et al., 2016).

2.3.3 Sensitivity Analysis

Numerical models are used in many fields to predict the response of complex systems. However, as computer power increases the complexity of the models increases as well. Generally, as the complexity of models increases the models output uncertainty increases also due to randomness present in input parameters (VU-BAC et al., 2016). Therefore, sensitivity analysis is a technique used to investigate how the variation of input parameters can influence the output of numerical models (PIANOSI et al., 2016).

The sensitivity analysis techniques can be divided into two main categories: qualitative and quantitative (PIANOSI et al., 2016). Qualitative methods provide a visual inspection of the model predictions, for example through plots or representations of further distributions of input parameters. On the other hand, sensitivity analysis via quantitative methods associates input factors to a reproducible and numerical evaluation of its relative influence, through a set of sensitivity indices.

There are many different quantitative methods such as differential sensitivity analysis, factorial design, importance factors, or percentage difference (HAMBY, 1994). Differential analysis can be used for several analytical methods such as Markov Reward Models,

Stochastic Petri Nets, and Queuing Networks. However, this technique may not properly evaluate the sensitivity either in non-continuous domains or to methods for which is not possible to derive closed-form equations. In these scenarios, the application of percentage difference approach can overcome the problems (ANDRADE et al., 2017).

Percentage difference technique calculates the sensitive index for each input parameter of a model from its minimum value to its maximum value, utilizing the entire parameter range of possible values. Equation 2.11 shows the calculus of the index S, where $min\{Y(\theta)\}$ and $max\{Y(\theta)\}$ are, respectively, the minimum and maximum output values that are computed when varying the input parameter θ over the range of the values of interest (ANDRADE et al., 2017).

$$S_\theta(Y) = \frac{max\{Y(\theta)\} - min\{Y(\theta)\}}{max\{Y(\theta)\}} \tag{2.11}$$

2.4 CONCLUDING REMARKS

We present in this chapter a set of concepts needed to understand the models proposed in this dissertation. We present the concepts about the IoT and how applications can benefit from the edge, fog and cloud infrastructures. Our e-health system architecture used as base to create the models are based on these technologies. After, dependability concepts are presented since we are concerned to evaluate the availability of e-health systems. Afterwards, we present the theoretical explanation of the modeling techniques that we used in this dissertation. Finally, we present the concept of sensitivity analysis and its importance to evaluate parameters of models.

3 PROPOSAL

In this chapter we describe the proposed models. First, we present the IoT application that we chose to model. We focus on e-health system context due its requirements such as reliability, interoperability, and low-latency response (RAHMANI et al., 2018). However, our models can be applied to other IoT scenarios with similar requirements.

We start with a description of the modeled architecture in this dissertation. This architecture represents a generic e-health system that relies on edge, fog, and cloud infrastructures. Afterwards we describe the cloud models, while focusing on the IT subsystem, where the applications will be hosted. We will explain the IT infrastructures variation suggested to increase availability. Then we show how to integrate the cloud models with the availability edge and fog infrastructures. Next, we detail the performance model proposed to assess the performance metrics about the system. Finally, we will describe the prototype that will be used to measured data from feed the performance model.

3.1 E-HEALTH MONITORING SYSTEM

The popularization of smart end devices, coupled with the advances in information and network technologies in recent years, have made possible the development of cheaper and more affordable health systems. IoT has played an important role in the evolution of these systems, providing low cost sensors to monitor many aspects of a patient´s life (CHIUCHISAN; COSTIN; GEMAN, 2014).

As stated previously, edge devices can use cloud computing to improve the availability and performance of medical applications. For instance, Sierra wireless[1] enables the connection between IoT devices and cloud computing infrastructure to collect and analyze real time data from hospitals and home health monitoring devices (CHIUCHISAN; COSTIN; GEMAN, 2014). As an e-health monitoring system monitors a patient continuously, it collects vast amounts of data that needs to be analyzed in relative real-time without interruption. Significant delays in receipt of data can compromise the efficacy of an e-health application and impact the patient's well-being and health. In this case, according to (MELL; GRANCE et al., 2011), fog is positioned to play a significant role in the ingestion and processing of the data close to the source as it is being produced.

In this dissertation, we propose an architecture to represent the behavior of an e-health monitoring system that relies on sensors, fog devices (such as Raspberry Pi[2]) and cloud infrastructure (public or private cloud services) to process and store patients' vital signs data. Figure 5 presents our proposed architecture. We assume that patients have sensors that collect the relevant vital data and these sensors are coupled to a microcontroller (such

[1] https://www.sierrawireless.com
[2] https://www.raspberrypi.org/

as an Arduino[3]). In this case, use assume a microcontroller as a device which aggregate the data of sensors and has connectivity to send them to be processed in most powerful devices (cloud or fog infrastructures) (LIMAYE; ADEGBIJA, 2018). We also assume that we have two different applications that consume collected data: *(a)* a fog application, and *(b)* a cloud application. We refer to these applications as web applications.

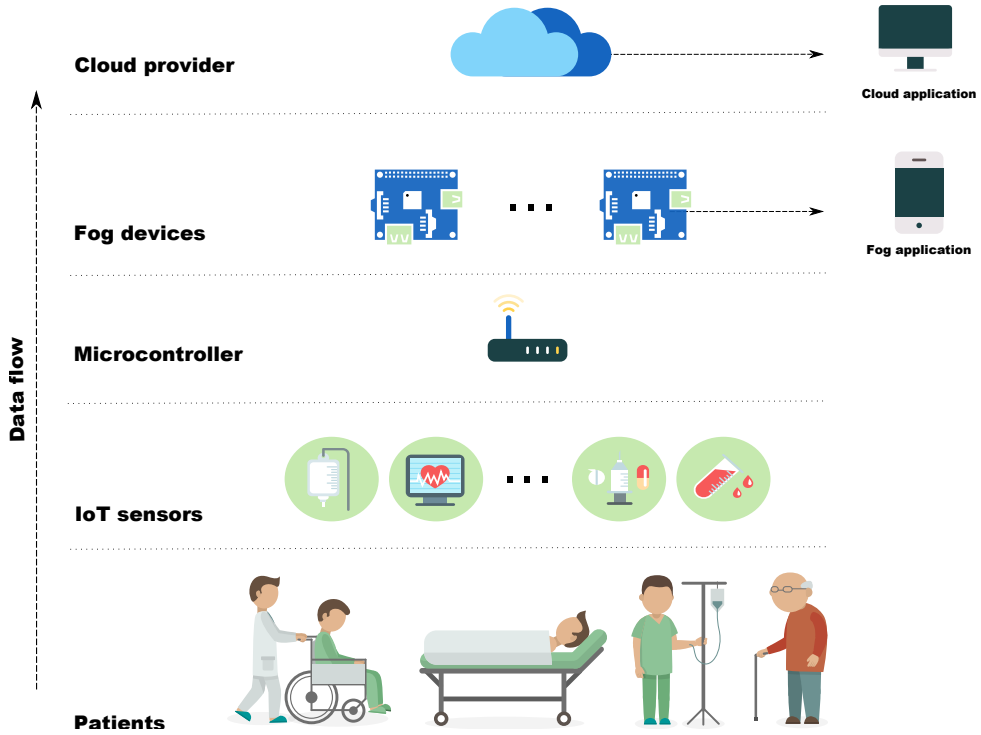


Figure 5 – An e-health monitoring system architecture

The fog application checks the normality of the data and, in the case of an anomaly, it may instigate an action, e.g., generate a call to an emergency service. Physiological data may be sent to the cloud application for further processing, e.g., to train a machine learning algorithm to examine a patient's condition over time and to compare one patient against a larger population to help doctors to provide better treatment[4].

Given the architecture showed in Figure 5, we consider three different availability scenarios (see Figure 6) in which our e-health monitoring system is implemented:

- **Scenario 1**: The e-health monitoring system availability depends on all components of the system, i.e., a failure in any of them results in a system failure. This scenario does not present redundancy and the two applications are complementary, i.e., when one of them fails, the system becomes unavailable. Data is sent to fog devices to perform some pre-processing that does not require large computational capacity. Later, the data is sent from the fog device to a cloud server to complete the processing and stored. A similar scenario is presented in (LI; OTA; DONG, 2018);

[3] https://www.arduino.cc/
[4] http://www.nvidia.com/object/deep-learning-in-medicine.html

- **Scenario 2**: The e-health monitoring system relies only on the cloud application and infrastructure to send patient vital signs data. As such, the system availability estimation does not take in account the fog application and fog device; and
- **Scenario 3**: This scenario is similar to Scenario 2 but here the system relies only on the fog application and infrastructure to receive the patient data; the e-health monitoring system availability estimation does not consider the cloud application and cloud infrastructure.

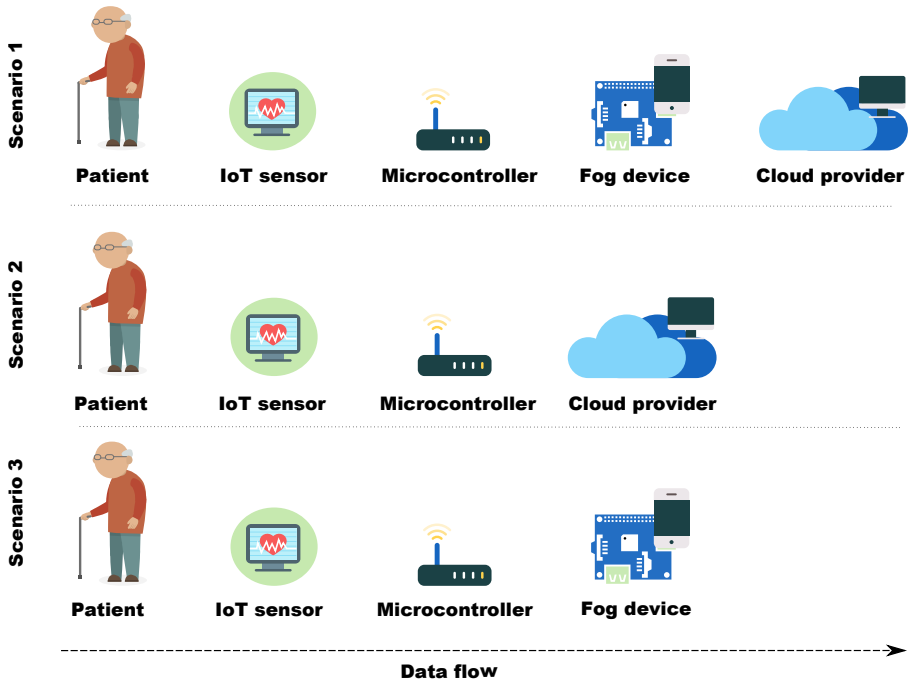


Figure 6 – E-health monitoring system scenarios

We will model these different scenarios through SPN and RBD. First, we will model the cloud data center with different configurations, with the purpose of increasing the availability of services hosted in the cloud. Next we will integrate the cloud with fog and edge devices using simplified building blocks in SPN.

These models can be used for hospitals to design new services and improve the patient experience. For example, the hospital may provide a ambient assisted living service for the patients, where a set of sensors in the patient house collect the data about the patient health, thus the doctor can evaluate these data that is stored in the cloud computing infrastructures. Another possible scenario is to automatically monitor (through cloud-hosted services) several patients in a hospital, without the need for a professional to be accompanied by each patient in person (RAHMANI et al., 2018). However, these applications must the be designed carefully to provide the service properly in order to avoid critical problems for the patients. “E-health system providers” need to evaluate the architecture

of the service in a thorough manner, preferably even before the service is operational, to avoid possible complications about its operation. For this purpose, stochastic models can be used to evaluate different e-health system architectures and plan the greatest configuration for meet the e-health system providers requirements.

3.2 AVAILABILITY MODELS

3.2.1 Cloud models

In this dissertation, we are considering an application running on top of the IT infrastructure in the data center. We focus on the IT infrastructure, once it is the main subsystem of the data center and our models are based on the TIA-942 classification. As explained in Subsection 2.1.2, this standard classifies the data centers in tiers, according with availability level.

Figures 7 and 8, depict a Tier I and Tier IV data center IT subsystem, respectively. A data center IT subsystem is basically composed of servers, storage, and network components. The storage is illustrated as Network Attached Storage (NAS) Disk Array. Network is represented by Edge, Core, Aggregation routers, and Access switch. The Storage Area Network (SAN) is a network component used in Tier IV to connect array disks to servers. As one may note, the Tier I IT subsystem does not consider any component redundancy, being susceptible to system disruptions from planned and unplanned failures. On the other hand, Tier IV is a fully redundant architecture ($2N$).

To model the applications running in a server, we used a RBD to represent dependency between its components. The IT infrastructure is comprised by network components, storage, and servers. However, the behavior of the IT infrastructure components are modeled using SPN. These models are described next.

3.2.1.1 RBD model of service

Figure 9 shows the RBD that represent a server with the application instance running. This RBD model is composed of : hardware (HW), operating system (OS), virtual machine (VM) and the application (APP) instance that is running on this server. Each block has the MTTF and MTTR values respective to its component. It is worth mentioning that any application can be modeled, once the MTTF and MTTR values are known.

Solving the RBD ((VERMA; SRIVIDYA; KARANKI, 2010)), we obtain the MTTF and MTTR values of the entire service running in a server. These values will be added in transitions referring to the server in our SPN model (Figure 11), described next.

3.2.1.2 IT Subsystem Tier I

In this dissertation, we disregarded some components of the infrastructure illustrated in Figure 7. The Edge router Wide Area Network (WAN) was disregarded because is used

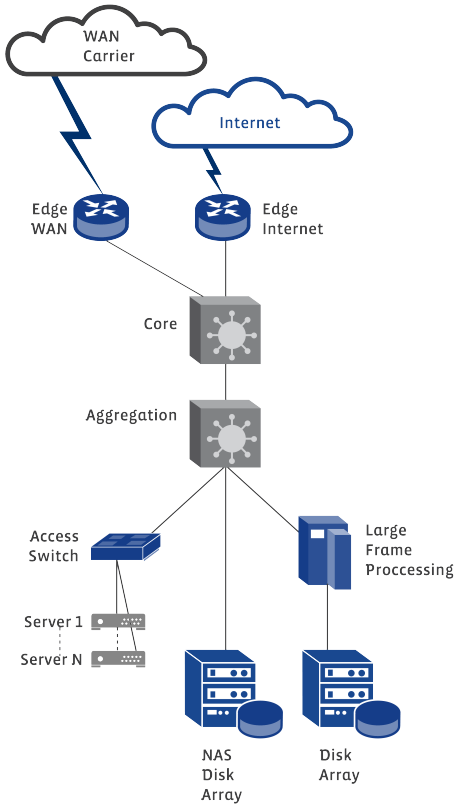


Figure 7 – IT architecture - Tier I

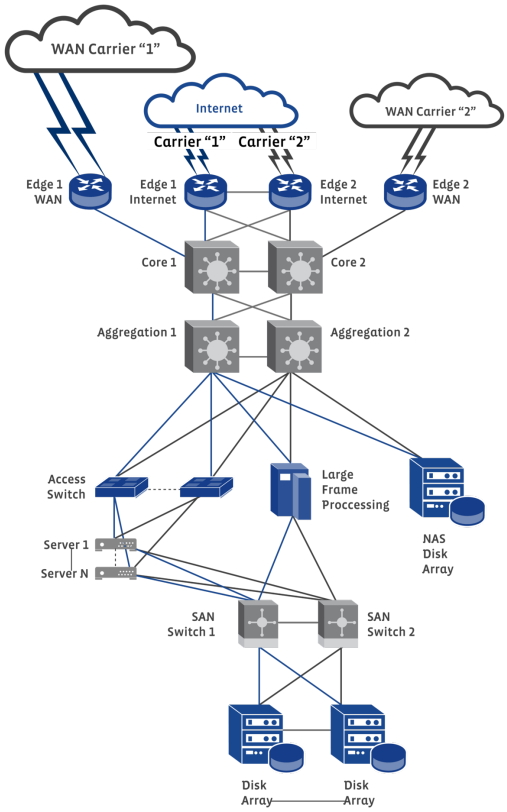


Figure 8 – IT architecture - Tier IV

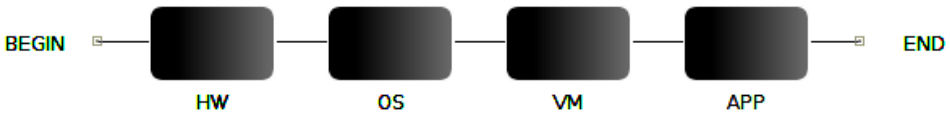


Figure 9 – RBD model of service, based on (ARAUJO et al., 2014a)

for corporate networks, and does not affect the availability of applications hosted in cloud data center. We also disregard the large frame processing and disk array attached to it, because is used for backup, and does not impact the availability of applications.

Each network component is modeled in SPN using a building block with two places and two transitions. Places means the state of components (UP or $DOWN$) and transitions represent actions of these components (fail and repair). For example, the Core router is modeled as shown in figure 10. The place $IT\_CORE\_1\_UP$ represents when the Core router is UP, while the place $IT\_CORE\_1\_DOWN$ represents when this component is down. The failure of this component is modeled by transition IT_ET_3, that consumes a token in place $IT\_CORE\_1\_UP$ and produces a token in place $IT\_CORE\_1\_DOWN$. Core router repair is modeled by transition IT_ET_4, and follows the inverse path of the failure transition. The other network components are similarly modeled, each one with own building block, as shown in Figure 10.

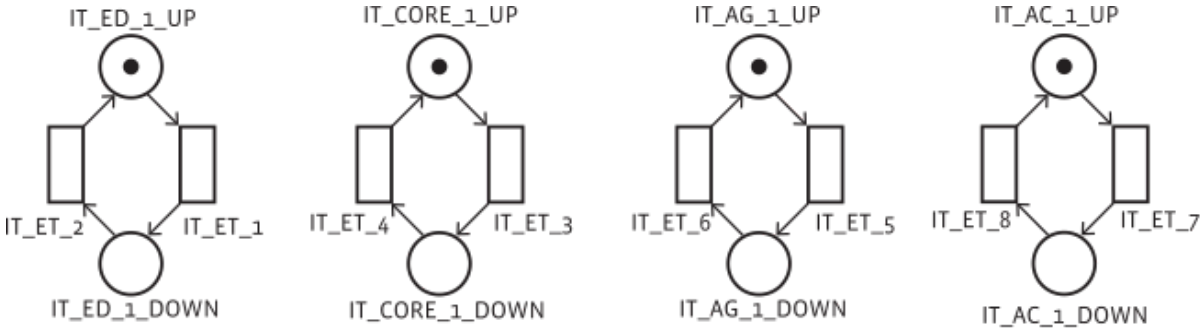


Figure 10 – SPN network components tier I

NAS and servers differ from network components due to the addition of more places and transitions, as shown in Figure 11. This is due to the fact that when one network component fails, servers and NAS become unavailable. Thus, we used an additional place and two immediate transitions to model this behavior.

For example, servers are modeled using a building block with places up/down and transitions failure/repair, like network components. When any network device fails, the immediate transition IT_IT_1 fires making the servers unavailable (consuming one token from the place $IT\_SERV\_1\_UP$ and producing one token in the place $IT\_SERV\_1\_UN$. When the network devices is repaired, the immediate transition IT_IT_2 fires, and the server become available again (consuming one token from the place $IT\_SERV\_1\_UN$ and producing one token in the place $IT\_SERV\_1\_UP$. NAS is modeled in a similar way, but disregards the access switch, since the NAS is connected directly to the aggregation switch (Figures 7 and 8).

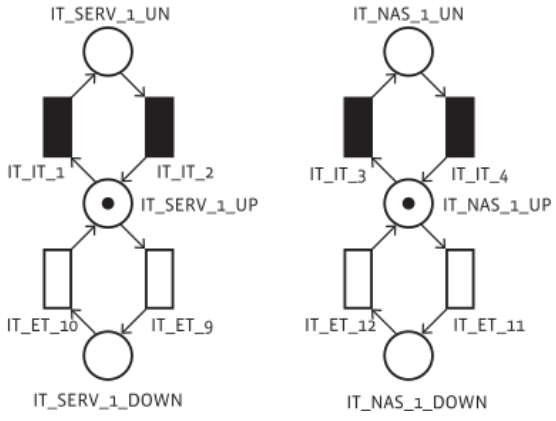


Figure 11 – Storage and server tier I

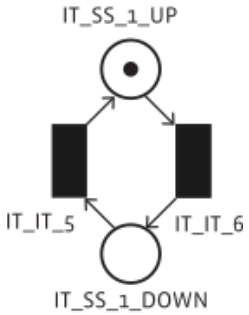


Figure 12 – Building block regarding IT infrastructure status

The behavior mentioned above is assured by guard functions, described in Table 1. As can be seen in Figure 7, only Edge router, Core router, Aggregation router, and Access Switch are connected to servers. So, if one these components fails, servers will become unavailable. This behavior is modeled by guard function associated with transition IT_IT_1.

On the other hand, if all components are running, the servers are available, and this behavior is ensured by guard function associated with transition IT_IT_2. Guard functions present in immediate transition associated with transition IT_IT_3 and associated with transition IT_IT_4 are similar, but only Edge router, Core router, Aggregation router are connected to NAS, therefore the failure of one of these components will make the NAS unavailable. This behavior is modeled by the guard function associated with transition IT_IT_3. The guard function present in transition IT_IT_4 models these components' repair, allowing NAS to become available again.

Table 1 – Guard functions of IT subsystem - tiers I to IV

Transition	**Guard Function**
IT_IT_1	$((\#IT\_ED\_1\_UP = 0) OR (\#IT\_CORE\_1\_UP = 0)$ $OR(\#IT\_AG\_1\_UP = 0) OR (\#IT\_AC\_1\_UP = 0))$
IT_IT_2	$((\#IT\_ED\_1\_UP > 0) AND (\#IT\_CORE\_1\_UP > 0)$ $AND(\#IT\_AG\_1\_UP > 0) AND (\#IT\_AC\_1\_UP > 0))$
IT_IT_3	$((\#IT\_ED\_1\_UP = 0) OR (\#IT\_CORE\_1\_UP = 0)$ $OR(\#IT\_AG\_1\_UP = 0))$
IT_IT_4	$((\#IT\_ED\_1\_UP > 0) AND (\#IT\_CORE\_1\_UP > 0)$ $AND(\#IT\_AG\_1\_UP > 0))$
IT_IT_5 $(tier\ I\ and\ II)$	$((\#IT\_NAS\_1\_UP > 0) AND (\#IT\_SERV\_1\_UP > 0))$
IT_IT_6 $(tier\ I\ and\ II)$	$((\#IT\_NAS\_1\_UP = 0) OR (\#IT\_SERV\_1\_UP = 0))$

Figure 12 shows the building block that represents the status of the IT infrastructure. When the IT infrastructure is working (token in place $IT\_SS\_1\_UP$), if servers or storage fails, the immediate transition IT_IT_6 fires, making the system unavailable (token in place $IT\_SS\_1\_DOWN$). When server and storage become running again, the immediate transition IT_IT_5 fires, making IT infrastructure available again. The guard functions of these transitions are described in Table 1.

3.2.1.3 IT Subsystem Tier II

According with TIA-942 standard, the main difference between tier I and II is the dual Internet access link of Edge router. If one link fails, the second one will still keep the data center Internet connection. To represent this redundancy in our model, we just added a token in place $IT\_IT\_ED\_1\_UP$. So, if one link fails, one token will go to place $IT\_IT\_ED\_1\_DOWN$ but there will still be another token in the UP place.

3.2.1.4 IT Subsystem Tier III and IV

According TIA-942 specification, the differences between tiers III and IV are about low-level components such as cabling redundancy and component location. However, since we

are modeling components at a higher level, we disregard these differences. Therefore, tiers III and IV are modeled in a similar way.

As can see in Figure 8, in tiers III and IV, all components are redundant in order to keep the data center available, in case of unplanned outages. In addition, in tier III and IV there are more components compared to tier I. Two SAN switches connect servers to disk arrays to provide an additional storage. Similarly to previous tiers, in tiers III and IV large frame processing and edge router are disregarded. To model redundant components, there are two tokens in *UP* places and infinite server policy is used in the transitions. The SPN model of tiers III and IV is presented in Figure 13.

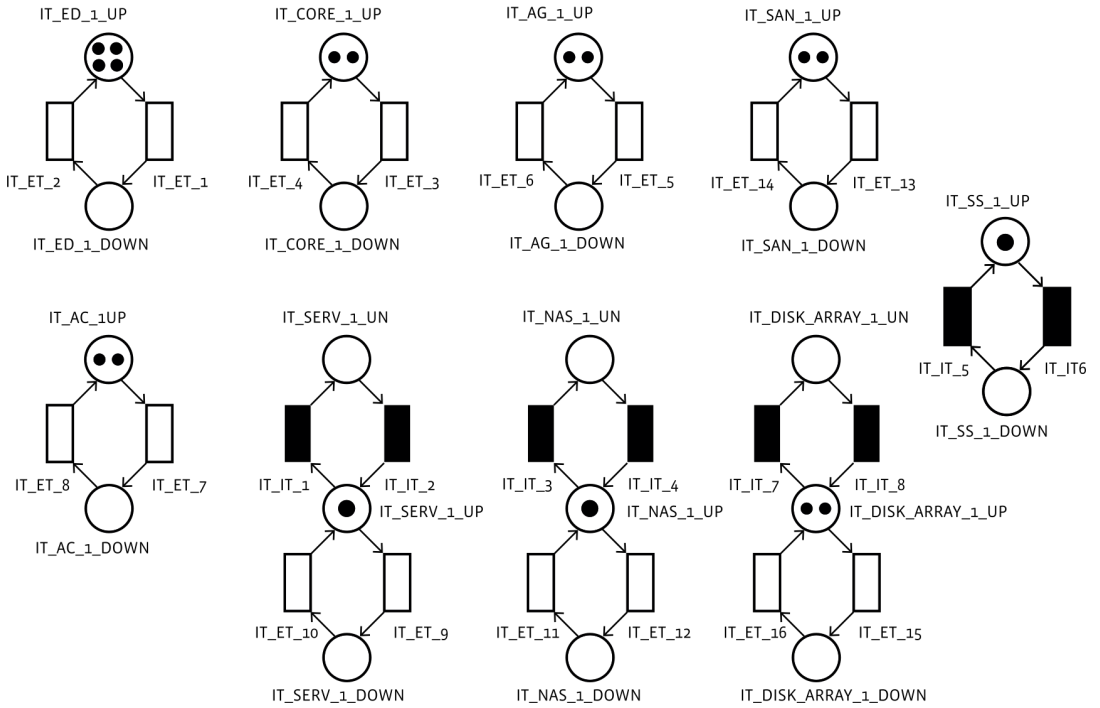


Figure 13 – SPN model of IT infrastructure - tiers III and IV

As the model is very similar to tier I, all stochastic and immediate transitions of tier I are also used in tier III and IV models. SAN switch is represented by a building block with two places $IT\_SAN\_1\_UP$ and $IT\_SAN\_1\_DOWN$, and repair/-failure transitions IT_ET_14 and IT_ET_13, like other components. The component that represents new array disk is similar to server and NAS components. The place $IT\_DISK\_ARRAY\_1\_UP$ represents when new disk array is *UP*, while place $IT\_DISK\_ARRAY\_1\_DOWN$ represents when it is *DOWN*.

Stochastic transitions IT_ET_15 and IT_ET_16 model repair and failure of new disk array, respectively. However, when both SAN switches fail, the second array disks will be unavailable ($IT\_DISK\_ARRAY\_1\_UN$), and this behavior is assured by the guard function presents in immediate transition IT_IT_7. This transition consumes tokens of place $IT\_DISK\_ARRAY\_1\_UP$ and produces a token in place $IT\_DISK\_ARRAY\_1\_UN$. When a single SAN switch is repaired, the second array disk become available again,

as modeled in guard function present in immediate transition IT_IT_8. This transition consumes all tokens present in place $IT\_DISK\_ARRAY\_1\_UN$ and produces a token in place $IT\_DISK\_ARRAY\_1\_UP$. These guard functions and respective transitions are presented in Table 2.

As tier III and IV have an addition array disks to storage, guard functions presents in immediate transitions IT_IT_5 and IT_IT_6 change. Now, when at least one of the storage way is working and servers are working, system is available. On the other hand, when servers are not working or both storage way are not working, system becomes unavailable. These guard functions are described in Table 2.

Table 2 – Guard functions of immediate transitions - tier III and IV

Transition	**Guard Function**
IT_IT_5	$(\#IT\_SERV\_1\_UP > 0) AND$ $((\#IT\_NAS\_1\_UP > 0)\ OR (\#IT\_DISK\_ARRAY\_1\_UP > 0))$
IT_IT_6	$(\#IT\_SERV\_1\_UP = 0) OR$ $((\#IT\_NAS\_1\_UP = 0)\ AND (\#IT\_DISK\_ARRAY\_1\_UP = 0))$
IT_IT_7	$(\#IT\_SAN\_1\_UP = 0)$
IT_IT_8	$(\#IT\_SAN\_1\_UP > 0)$

Using these models, it is possible to evaluate the availability of cloud infrastructure, and calculate the MTTF and MTTR values that will be used to integrate with fog and edge devices.

3.2.2 Integrating cloud models with edge and fog models

Figure 14 shows our SPN model representing the whole e-health monitoring system. We consider an architecture with the following components: the sensor, the microcontroller (in this dissertation, the sensor and the microcontroller are represented in an edge device), fog device, and cloud data center. To represent these components, we utilize building blocks composed of two places (one to represent when the component is working (ON) and another to represent the failure (DOWN)), and two stochastic transitions (that represent (i) the failure and (ii) the repair of a specific component).

By way of illustration, in the cloud building block, the place *Cloud_ON* represents that the cloud provider is running, and the place *Cloud_OFF* that it has failed or unavailable. The transition *Cloud_Fail* represents the cloud failure event (MTTF value), while *Cloud_Repair* signals the time taken to repair the cloud infrastructure (MTTR value). In this dissertation, we consider that all failure and repair times are exponentially distributed (MATOS et al., 2015; MATOS et al., 2017). The other three components (sensors, microcontroller, and fog) follow the same operating logic.

Moreover, there are other three building blocks with immediate transitions (see the top of the Figure 14) that represent the system status in different scenarios. From left to right,

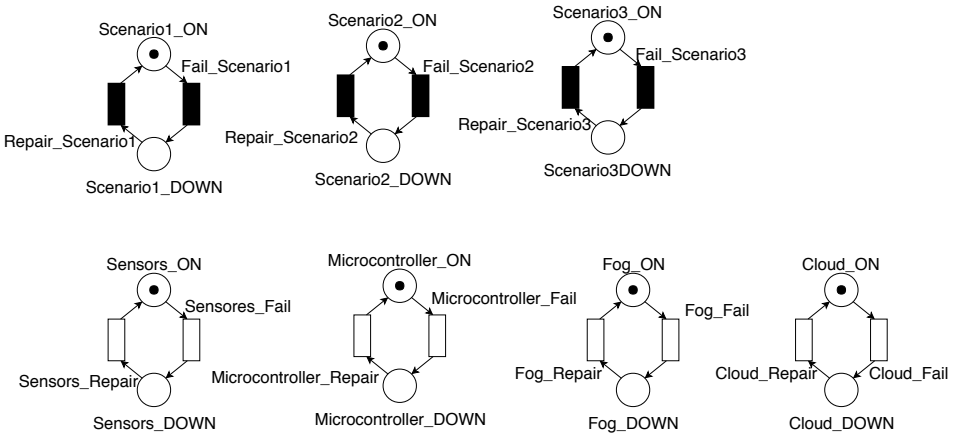


Figure 14 – E-health monitoring system model

those building blocks represent Scenarios 1, 2 and 3 respectively (described in subsection 3.1). Each scenario has a building block composed of ON and DOWN places and two immediate transitions. These places offer the same semantic meaning as the previous ones, as well as the transitions. The difference is that each transition is activated through a guard function instead of MTTF or MTTR values. These functions are presented in Table 3. For example, considering Scenario 1, where all components must be working (sensors, microcontrollers, fog devices, and cloud infrastructures), if one of these components fails, the transition *Fail_System_1* will fire, i.e., if there are no tokens in the respective places. Thus, the system becomes unavailable. On the other hand, when at least one of these components is working, i.e., there is one token in the respective places, the transition *Repair_System_1* will fire, making the system available again.

Table 3 – Guard functions of e-health system model

Transition	**Guard Function**
Fail_System_1	(#Sensors_ON=0)OR(#Microcontroller_ON=0)OR (#Fog_ON=0)OR(#Cloud_ON=0)
Repair_System_1	(#Sensors_ON=1)AND(#Microcontroller_ON=1)AND (#Fog_ON=1)AND(#Cloud_ON=1)
Fail_System_2	(#Sensors_ON=0)OR(#Microcontroller_ON=0)OR (#Cloud_ON=0)
Repair_System_2	(#Sensors_ON=1)AND(#Microcontroller_ON=1)AND (#Cloud_ON=1)
Fail_System_3	(#Sensors_ON=0)OR(#Microcontroller_ON=0)OR (#Fog_ON=0)
Repair_System_3	(#Sensors_ON=1)AND(#Microcontroller_ON=1)AND (#Fog_ON=1)

To provide more details about the fog device and cloud server that host an application, we modeled them as an RBD. To represent the fog device (Figure 15), we consider it is composed of hardware (HW), an operating system (OS), and the application (APP) that consumes the vital signs data from the patients. If any of these components fails, we consider that the fog device becomes unavailable and then the RBD is configured in a serial chain. The MTTF and MTTR values of cloud computing (that are assigned in transitions *Cloud_Fail* and *Cloud_Repair*) are extracted from cloud availability models showed in Subsection 3.2.1.

Figure 15 – Fog device model

The availability of each of the three scenarios is the probability of having a token in the place that represents the state ON of the respective building block (see Figure 14). To model the availability metric in our SPN, we utilize the following expression:

$$P\{scenario_x\_ON\} > 0 \tag{3.1}$$

where x is the number of the scenario (from 1 to 3), as described previously.

These models allow performing experiments to evaluate the availability of IoT applications that rely on cloud, fog, and edge infrastructures. In addition, it is possible to integrate with performance models in order to assess the impact of availability and performance of IoT applications.

3.3 PERFORMANCE MODEL

In this dissertation, we consider a selection of performance metrics based on (JAIN, 1990). To represent the clients requesting a service, we consider a queuing system $M/M/1/K$, meaning that the arrival process is a Poisson process with rate λ (M), the service time is independent and exponentially distributed with parameter µ (M), there is only a single server to process the requests (1), and the capacity of the system is limited (K). This queue configuration allows the evaluation of relevant aspects of the system, such as the impact of different arrival times and different queue capacities (RAHMATI et al., 2014); it is commonly used to represent cloud requests ((VILAPLANA et al., 2014; PHAM et al., 2015; AL-HAIDARI; SQALLI; SALAH, 2015; ADHIKARY et al., 2017; GOLDSZTAJN et al., 2018)).

In order to illustrate how these performance metrics were modeled using an SPN approach, consider the simple queue model shown in Figure 16. The transition $T1$ represents the arrival of requests while transition $T2$ represents the service time. We consider that both Arrival Time (AT) and Service Time (ST) are exponentially distributed (YANG et al., 2009; KHAZAEI; MISIC; MISIC, 2012). The Place $P1$ represents requests that are in service. The Place $P2$ represents when the system resource (e.g. a web server) is running, while $P3$ represents when a resource is in failure. Transitions $T\_failure$ and T_repair represent the failure and the repair of the resource, respectively. $T2$ only fires when there is a token in place $P2$, i.e., when the resource is running and this behavior is assured by guard function $\#P2 > 0$. The place $P4$ represents the total capacity K that the system can withstand. In other words, it is the number of requests that can be queued in system.

In this dissertation, we consider the following performance metrics[5]:

[5] The metric descriptions follow the Mercury tool syntax.

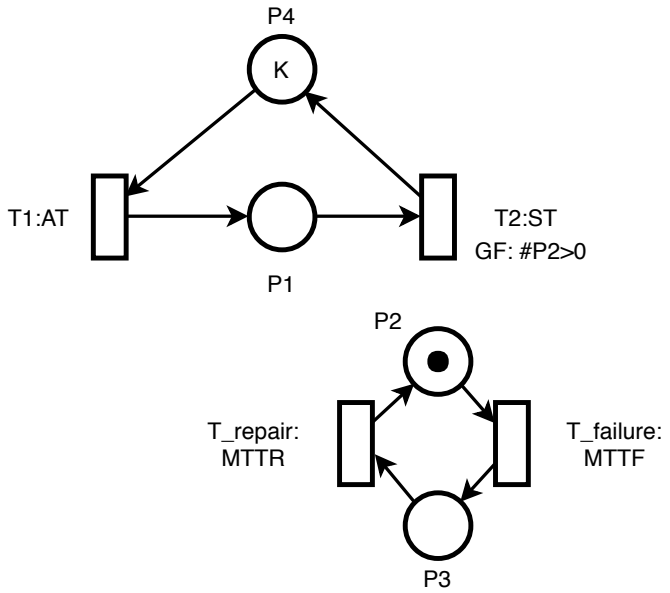


Figure 16 – SPN model to represent a simple queue

- **Throughput (TP)**: can be defined as the rate of requests that can be serviced by the system. Generally, the TP of a system increases as the load on the system increases. However, after a certain load level, the TP stops increasing, and, in most cases, even starts decreasing. In our system, the TP represents the number of requests the web applications (fog or cloud applications) can process. Taking into account the queue presented in Figure 16, TP can be calculated as the probability of having tokens in place $P1$ (requests in queue) and in place $P2$ (system working) multiplied by the service rate:

$$TP = P\{(\#P1 > 0)AND(\#P2 > 0)\} \times (1/ST) \tag{3.2}$$

- **Service Time ST**: is, in a simplified way, the interval between a user's request and the system response. In our system, we can define the interval between the Hypertext Transfer Protocol (HTTP) request from the microcontroller and the response time of the web application (hosted either in the fog or cloud). Service time can be calculated as the number of tokens expected in place $P1$ divided by TP:

$$ST = E\{\#P1\}/(P\{\#P1 > 0\} \times (1/ST)) \tag{3.3}$$

In this way and considering the proposed metrics to represent the e-health monitoring system performance, we propose the SPN model presented in Figure 17.

This model is composed of three queues with each one representing a different scenario. The Place *Requests_Arrival* indicates that there is data from the sensor to be sent to the web application (in the fog device or the cloud server). The stochastic transi-

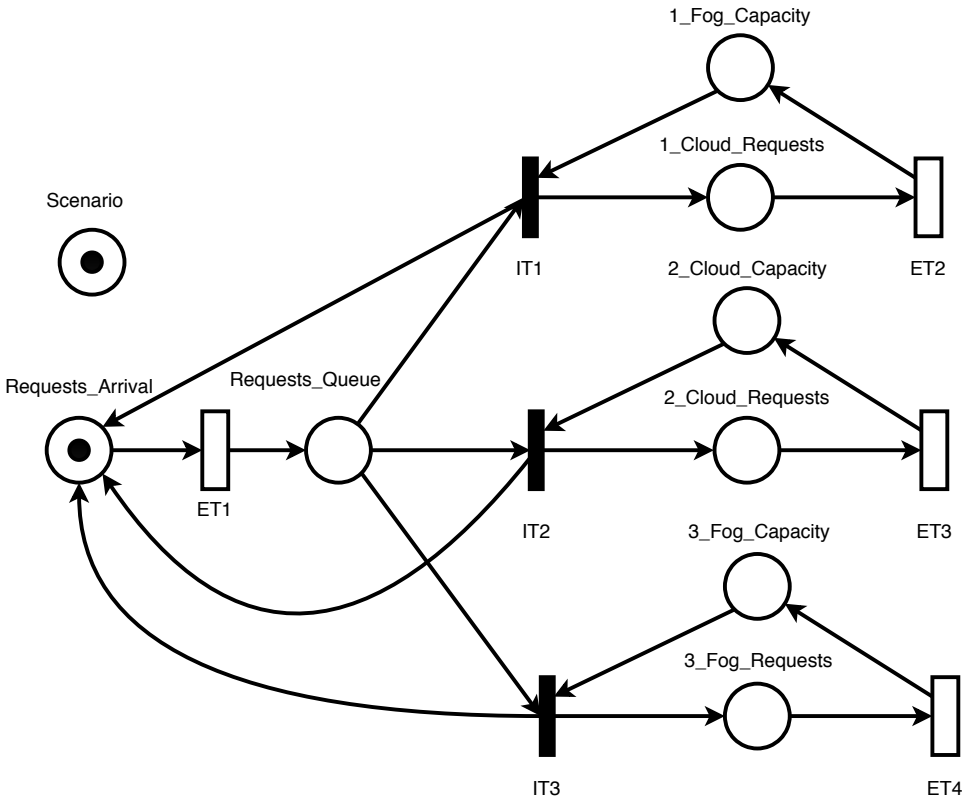


Figure 17 – Performance model of e-health monitoring system

tion $ET1$ indicates the arrival time of requests; it only fires when the sensor is working. Place *Requests_Queue* stores tokens that represent requests that will be sent to a web application. The immediate transitions $IT1$, $IT2$, and $IT3$ will fire in accordance with the number of tokens present in Place *Scenario* indicating the evaluated scenario.

Scenario 3 is an example where queue behavior is described in the model. The immediate transition $IT3$ will fire only when the microcontroller is working and there are three tokens in Place *Scenario* as in Scenario 3. The Place $3\_Fog\_Capacity$ represents the capacity of a fog device, i.e., how many requests a fog device can process simultaneously, through number of tokens in this place. The Place $3\_Fog\_Requests$ represents the number of requests being processed by a fog device. When $IT3$ fires, one token is consumed from Place $1\_Fog\_Capacity$ and one token is produced in Place $1\_Fog\_Requests$ and in Place *Requests_Arrival*, enabling new requests to arrive to the fog device. The stochastic transition $ET4$ represents the mean time for a fog application to process a request. When $ET4$ fires, one token is consumed from Place $3\_Fog\_Requests$ and one token is produced in Place $3\_Fog\_Capacity$ again (retrieving the capacity that was being consumed). Scenario 2 is modeled in a similar way but it is represented by Places $2\_Cloud\_Requests$ and $2\_Cloud\_Capacity$, and transitions $ET3$ and $IT2$.

Scenario 1 behaves similarly to the others, with only one difference: the immediate transition $IT1$ will fire when there is one token in Place *Scenario* and when the microcontroller and fog device are working, because these devices are responsible for sending

data to the cloud instance. We assume that this scenario is limited only by the capacity of the fog device once all requests to the cloud are sent by the fog device, and that the cloud has superior capacity than the fog device. So, the number of tokens in Place 1_*Fog_Capacity* represents the capacity of a fog device while Place 1_*Cloud_Requests* represents the number of requests being processed in the cloud instance in Scenario 1. The stochastic transition $ET2$ represents the time to process a request in Scenario 1.

All of this behavior is assured by guard functions presented in Table 4. Some of these guard functions connect the performance model with our availability model (Figure 14) in order to evaluate failure impact on system performance metrics. The equations to compute the performance metrics (TP and ST) are presented in Table 5 and follow the same logic presented previously.

Table 4 – Guard functions of performance model

Transition	**Guard function**
$ET1$	#Sensor_up>0
$ET2$	#Cloud_up>0
$ET3$	#Cloud_up>0
$ET4$	#Fog_up>0
$IT1$	(#scenario=1)AND(#Microcontroller_ON>0)AND(#Fog_ON>0)
$IT2$	(#scenario=2)AND(#Microcontroller_ON>0)
$IT3$	(#scenario=3)AND(#Microcontroller_ON>0)

Table 5 – Equations for performance metrics

Scenario	**Metric**	**Equation (Mercury tool sintaxe)**
1	throughput	P{(#1_Cloud_Requests>0)AND(#Cloud_ON>0)}*(1/ST_Fog_Cloud)
	service time	E{#1_Cloud_Requests}/(P{#1_Cloud_Requests>0}*(1/ST_Fog_Cloud))
2	throughput	P{(#2_Cloud_Requests>0)AND(#Cloud_ON>0)}*(1/ST_Cloud)
	service time	E{#2_Cloud_Requests}/(P{#2_Cloud_Requests>0}*(1/ST_Cloud))
3	throughput	P{(#3_Fog_Requests>0))AND(#Fog_ON>0)}*(1/ST_Fog)
	service time	E{#3_Fog_Requests}/(P{#3_Fog_Requests>0}*(1/Fog_Cloud))

3.4 PROTOTYPING A E-HEALTH MONITORING SYSTEM

Our main goal behind building and performing experiments in a prototype is to acquire real data to feed our analytical models. This prototype represents a simplified version of the architecture illustrated in Figure 5. With this prototype we measure the time to send data from the sensor to both the fog device and the cloud infrastructure.

3.4.1 Prototype infrastructure description

The prototype infrastructure is composed of two edge devices, two fog devices, and a cloud with four different geo-locations. Depending on the scenario, the amount of devices can vary. Figure 6 describes the scenarios considered in this dissertation. Table 6 describes the hardware specifications of the edge and fog devices, and the geo-locations of cloud instances.

Table 6 – Prototype infrastructure components

Device	**Type**	**Specification**
Edge device	Heart rate sensor	Operate from 3 V to 5 V
Edge device	Arduino UNO	Clock speed 16 MHz, SRAM 2 KB, Flash Memory 32 KB
Fog device	Raspberry Pi 3	Quad Core 1.2GHz CPU, 1GB RAM and 802.11n wireless
Fog device	Netbook	Intel Atom processor 1.6GHz, 2GB RAM and 802.11b/g/n wireless
Cloud	Elastic Compute Cloud (EC2)	Four different geographic locations: (a) Sao Paulo/Brazil, (b) California/USA, (c) London/England, and (d) Tokyo/Japan.

Considering all the components described in Table 6, we obtain different configurations for each scenario. For instance, Table 7 shows all possible combinations of component configurations related to Scenario 1. Scenario 1 has 16 combinations while Scenarios 2 and 3 both have four combinations.

Table 7 – Component configurations of Scenario 1

Scenario	**Configuration**	**Edge device**	**Edge device**	**Fog device**	**Fog device network**	**Cloud location**
1	1	Heart rate sensor	Arduino UNO	Raspberry Pi 3	Ethernet	Sao Paulo/Brazil
	2	Heart rate sensor	Arduino UNO	Raspberry Pi 3	Ethernet	California/USA
	3	Heart rate sensor	Arduino UNO	Raspberry Pi 3	Ethernet	London/England
	4	Heart rate sensor	Arduino UNO	Raspberry Pi 3	Ethernet	Tokyo/Japan
	5	Heart rate sensor	Arduino UNO	Raspberry Pi 3	IEEE 802.11	Sao Paulo/Brazil
	6	Heart rate sensor	Arduino UNO	Raspberry Pi 3	IEEE 802.11	California/USA
	7	Heart rate sensor	Arduino UNO	Raspberry Pi 3	IEEE 802.11	London/England
	8	Heart rate sensor	Arduino UNO	Raspberry Pi 3	IEEE 802.11	Tokyo/Japan
	9	Heart rate sensor	Arduino UNO	Netbook	Ethernet	Sao Paulo/Brazil
	10	Heart rate sensor	Arduino UNO	Netbook	Ethernet	California/USA
	11	Heart rate sensor	Arduino UNO	Netbook	Ethernet	London/England
	12	Heart rate sensor	Arduino UNO	Netbook	Ethernet	Tokyo/Japan
	13	Heart rate sensor	Arduino UNO	Netbook	IEEE 802.11	Sao Paulo/Brazil
	14	Heart rate sensor	Arduino UNO	Netbook	IEEE 802.11	California/USA
	15	Heart rate sensor	Arduino UNO	Netbook	IEEE 802.11	London/England
	16	Heart rate sensor	Arduino UNO	Netbook	IEEE 802.11	Tokyo/Japan

In Scenario 1, we use a heart rate sensor as the edge device [6]. This sensor reads the heart beats using an amplified optical sensor to estimate the heart beat per minute Beat

[6] https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino

per Minute (BPM), the signal, and the interval between beats (IBB) of the patient. This sensor is attached to the Arduino platform, UNO[7], that acts as a microcontroller.

To send the vital signs data generated from the sensor to equipment with better computational capacity, for example a fog or cloud device, we use an Ethernet shield [8] module to enable the Arduino to send data to higher layers. A software was developed that reads data from the heart rate sensor and periodically sends the data (through HTTP requests) to a web application hosted in the fog and cloud layers.

Currently, a number of specialized IoT communication protocols have been developed such as Message Queuing Telemetry Transport (MQTT) [9] and Constrained Application Protocol (CoAP) (SHELBY; HARTKE; BORMANN, 2014) in order to provide a lightweight communication protocol to support IoT device communication. However, it is still useful to implement a HTTP-based application, since HTTP is the most common protocol used for Internet communication, and according to (SHANG et al., 2016), IoT applications usually adopt HTTP as the messaging protocol in order to support Representational State Transfer (Rest) interfaces. In this way, we implemented our prototype following a Restful architecture using HTTP to send data from sensors to a fog device and to a cloud infrastructure.

To represent the fog device in our prototype, we use two different devices: *(a)* Raspberry Pi 3 [10], with Quad Core 1.2GHz CPU and 1GB RAM; and *(b)* Netbook CCE with Intel Atom processor 1.6GHz with 2GB RAM. Both devices were connected to the Arduino by using the same network (Ethernet and IEEE 802.11), also located in Recife, Brazil.

The public cloud environment used was the Elastic Compute Cloud (EC2) from Amazon Web Services (AWS)[11]. EC2 allows users to easily create, launch, stop, or terminate one or multiple instances as well as selecting the operating system and applications (CHEN et al., 2017a). Also, it is possible to select the geographic region in which the instance will be hosted. As such, in order to measure the impact of location on each instance, we created instances in four different geographic regions: *(a)* São Paulo/Brazil, *(b)* California/USA, *(c)* London/England, and *(d)* Tokyo/Japan.

3.4.2 Prototype applications description

On the fog and cloud computing side, we configured a web application that receives and processes data from the edge device. This web application was implemented using Python and Flask[12]. In addition, we use Apache as a container for both applications. Figure 18

[7] https://www.arduino.cc/
[8] https://www.arduino.cc/en/Guide/ArduinoEthernetShield
[9] http://mqtt.org/
[10] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[11] www.aws.amazon.com/ec2
[12] Flask is a microframework for building web applications using the Python language. See http://flask.pocoo.org/docs/0.12/

shows an example considering the Scenario 1. The Scenario 2 is similar, but it disregards the fog devices, while the Scenario 3 disregards the cloud infrastructure (see the Figure 6).

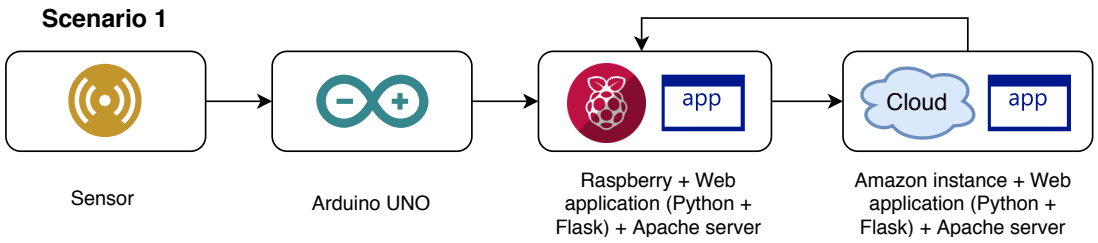


Figure 18 – Example of prototype configuration in scenario 1

We use the same web application in Scenarios 2 and 3, however the cloud and fog applications differ from Scenario 1. In Scenario 1, the fog application receives data from the Arduino and sends the data to the cloud application. After it receives a cloud response, the fog device returns a response to the Arduino. In Scenarios 2 and 3, the cloud and fog applications receive and process requests directly from the Arduino and send back a confirmation response to the Arduino platform.

4 AVAILABILITY ANALYSIS OF DATA CENTER MODELS

We separated the results of the stochastic models in two chapters. In this chapter, we present the results for the individual cloud models presented in the Section 3.2, some related works, and discuss the obtained results. In the next chapter, we present the results for the integrated models, that jointly consider the edge, fog, and cloud infrastructure.

4.1 CLOUD RESULTS

We performed stationary analysis using the Mercury tool[1] in order to calculate the availability of cloud models previously described. In this method, an underlying CTMC defined by the SPN model can be solved in order to obtain a analytic solution (STOJIC, 2017) and we use this method to solve all models proposed in this dissertation.

Regarding the RBD model of server (see Figure 9), the MTTF and MTTR values are presented in Table 8. We are considering a web application (APP), but we can easily use other type of applications here instead[2]. Based on this RBD model, we calculate the MTTF and MTTR of the server, and these values are used in our SPN (specifically in transitions IT_ET_9 and IT_ET_10, respectively), in order to estimate the overall data center availability. We consider evaluating only the most basic and fully-redundant tiers (tier I and IV) of the TIA-942 standard, in order to see the impact of redundancy in data center cloud availability. It is worth to mention that all stochastic transitions of the models proposed in this dissertation are exponentially distributed (ANDRADE et al., 2017).

Table 8 – RBD parameters obtained from (ARAUJO et al., 2014a)

Components	**MTTF (in hours)**	**MTTR (in hours)**
HW	8,760	1.667
OS	1,440	1
VM	2,880	0.17
APP	6,865.3	0.167

Table 9 presents all MTTF and MTTR values of the stochastic transitions we used in our SPN models, including MTTF and MTTR obtained from our RBD model.

Table 10 shows the availability level and the downtime of both Tier I and IV. Tier IV presents approximately 99.90% of availability, corresponding to around 8.58 hours of downtime in a year, while the Tier I is only 99.76%, meaning 20.96 hour of downtime.

[1] http://www.modcs.org/?page_id=1397

[2] We taken the MTTF and MTTR values of a web application from the literature, but as future works we plan develop our application and measure its parameters.

Table 9 – Parameters of stochastic transitions obtained from (GUIMARAES; MACIEL; JUNIOR, 2015), (YUE et al., 2016), and (SCHROEDER; GIBSON, 2007)

Transition	**Meaning**	**Value (in hours)**
IT_ET_1	Edge Router MTTF	796
IT_ET_2	Edge Router MTTR	1
IT_ET_3	Core Router MTTF	16243
IT_ET_4	Core Router MTTR	0.78
IT_ET_5	Aggregation Router MTTF	8247
IT_ET_6	Aggregation Router MTTR	0.63
IT_ET_7	Access Switch MTTF	13043.48
IT_ET_8	Access Switch MTTR	0.35
IT_ET_9	Server MTTF	768.35
IT_ET_10	Server MTTR	0.7533
IT_ET_11	NAS MTTF	1200000
IT_ET_12	NAS MTTR	12
IT_ET_13	SAN MTTF	255358
IT_ET_14	SAN MTTR	7.66
IT_ET_15	Disk Array MTTF	1200000
IT_ET_16	Disk Array MTTR	12

Table 10 – Availability data center evaluation

Tier	**Availability (in %)**	**Downtime (in hours/year)**
I	99.7606835	20.9641
IV	99.9020543	8.5800

The formula used to calculate the downtime, D, in hours/year, takes into account the availability, A, and is shown in Eq. 4.1.

$$D = (1 - A) \times 8760 \tag{4.1}$$

We also performed sensitivity analysis to verify which parameters affect more the overall data center availability.

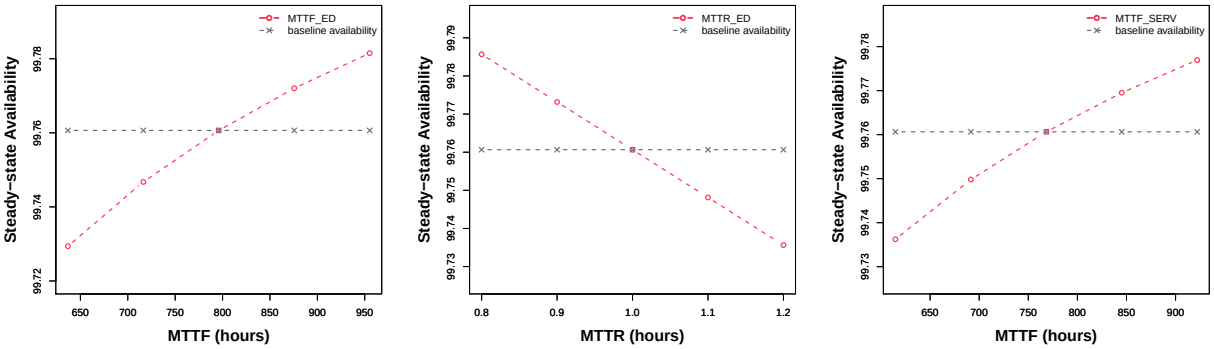
4.1.1 Sensitivity Analysis - Tier I

Table 11 shows the sensitivity result of our Tier I SPN containing the three higher and lower indices. Parameters with values equal to zero are not considered.

Results indicate that the edge router MTTF ($edge\_MTTF$) has the greatest impact in the data center availability in case of the Tier I. In other words, a variation in this value has a higher impact on the service availability. The second and third pa-

Table 11 – Sensitivity ranking of Tier I

Parameter	Sensitivity Index
$edge\_MTTF$	2.53×10^{-4}
$edge\_MTTR$	2.51×10^{-4}
$serv\_MTTF$	1.96×10^{-4}
$access\_MTTR$	4.02×10^{-6}
NAS_MTTF	2.02×10^{-6}
NAS_MTTR	1.98×10^{-6}



(a) (b) (c)

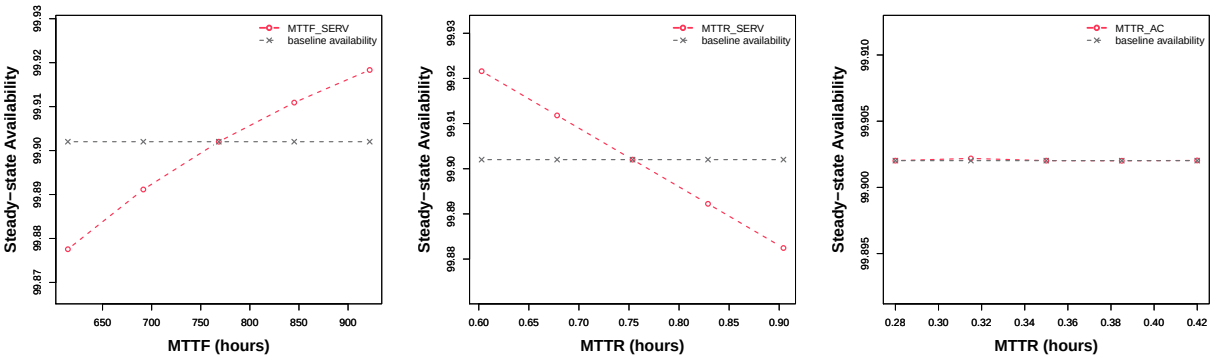
Figure 19 – Impact on Tier I availability varying (a) Edge Router MTTF, (b) Edge router MTTR and (c), Server MTTF.

rameters that have more impact are the edge router MTTR ($edge\_MTTR$) and server MTTF ($serv\_MTTF$), respectively. The metric with the smallest index was NAS MTTR (NAS_MTTR). When the value of this metric was changed from 12h to 14h (increase of 20%), a small availability variation (99.760649% to 99.760449%) was registered.

Figure 19(a) shows the Tier I data center availability when we varied the $edge\_MTTF$ value. A variation of 20% in the $edge\_MTTF$ results in an availability increase from 99.76065% to 99.78151%, that means a downtime decrease from 20.96706h to 19.139724h. Figure 19(b) shows that the availability decreasing from 99.76065% to 99.73563%, when $edge\_MTTR$ increases from 1h to 1.2h. And Figure 19(c) shows the impact of the increase in $serv\_MTTF$ parameters; which leads to the improvement of the overall availability.

4.1.2 Sensitivity Analysis - Tier IV

Table 12 presents the results of sensitivity analysis regarding Tier IV architecture with the three higher and lower indices. The result is different from Tier I because all architectural components are duplicated; has an influence on which components are most critical for availability. The server MTTF ($serv\_MTTF$) has the greatest impact on



(a) (b) (c)

Figure 20 – Impact on Tier IV availability varying (a) Server MTTF, (b) Server MTTR and (c), Access switch MTTR.

data center availability; and the second and third values that have the most impact are server MTTR ($serv\_MTTR$) and access switch MTTR ($access\_MTTR$), respectively. The metric with the smallest sensitivity analysis index was disk array MTTR ($disk\_array\_MTTR$). When disk array MTTR value varied in 20% resulted in an impact on availability only in the seventh decimal place.

Table 12 – Sensitivity ranking of Tier IV

Parameter	**Sensitivity Index**
$serv\_MTTF$	1.98×10^{-4}
$serv\_MTTR$	1.96×10^{-4}
$access\_MTTR$	1.76×10^{-6}
NAS_MTTR	7.20×10^{-11}
$disk\_array\_MTTF$	6.78×10^{-11}
$disk\_array\_MTTR$	3.17×10^{-11}

As presented in Figure 20(a), a variation of 20% in $serv\_MTTF$ resulted in an availability increase from 99.90202% to 99.91834%. On the other hand, an increase of 20% in $serv\_MTTR$ resulted in an availability increase from 99.90202% to 99.88245% (see Figure 20(b)). Figure 20(c) depicts the impact of the access switch MTTR ($access\_MTTR$) on a data center availability. The availability declines with the increase of this parameter.

4.2 RELATED WORKS

The authors in (ANDRADE; NOGUEIRA, 2018) propose an SPN model to evaluate several metrics about cloud data centers in the disaster recovery context. The scenario considered is composed of two data centers. The first one there are six VMs run a service that requires

high-performance like scientific computing. The second data center acts as failover site, and when the primary data center fails, the service is replicated for the second data center. There is a disaster monitor to check if the primary data center fails, and configure the load balance to redirect the requests for the second data center. Using the SPN model proposed, the authors carried out experiments evaluating performability metrics such as availability, downtime, virtual machines utilization, and average number of busy VMs. The authors also evaluate how the number of active VMs influences the service time: as the number of VMs increases the services times decreases because there are VMs to handle the requests.

In (MELO et al., 2015), authors propose an availability model of a Eucalyptus cloud environment that runs a video streaming application. To estimate availability, authors use RBD and Markov Chains. The RBD models the components of the Eucalyptus architecture, while the Markov Chain models the behavior of the streaming service. The results of sensitivity analysis show that the repair rate of the front-end module is the most important parameter with respect to the availability.

A model for evaluating availability of private clouds is proposed by (MATOS et al., 2016) where RBD and Markov Chains are used. The architecture is based on Eucalyptus, and employs warm-standby in the main components. RBD is used to model the dependency between components, while Markov Chains model the redundant behavior of cloud components. Authors evaluate three architectures with one, two and three clusters, achieving availability of 99.9938749%, 99.9969376% and 99.9969377%, respectively.

The authors in (CHEN et al., 2017b) present a CTMC model to assess the survivability of cloud services. Survivability is a interesting aspect that have direct impact on the availability services, once this metric is related to the recuperation of services after its fails. The model considers an architecture composed of two data centers. In the first one there are two servers: the first one has two active service running in virtual machines, and the last one is standby. The second data center has a server in standby too with a virtual machine. The both data centers have NAS as storage unit. Finally the two data centers are connected into a backup server. The CTMC model consider the migration of virtual machine in case of failure of the servers which the service is running with the purpose keep the service available. The authors evaluate the probability of the service is recovery at the different timestamps, which increases and remains constant after a time.

In (JAMMAL et al., 2016), the authors proposed an SPN model to evaluate the availability of cloud services. Their consider three main components in the model: virtual machines which run in server operating in a data center. The model also consider the arrival of requests, and the failures of the components may impacts the processing of this requests. The authors evaluate the percentage of requests processed in the data center varying the MTTF and MTTR values of the components and the processing time. The results showed that the decrease of the MTTR value results in a great impact on the

number of requests processed.

Our work differs from the literature because we consider a more detailed and scalable IT subsystem model (other works are focused only on the software level and the hardware level is not detailed), and we also take into account data center standard to create our models.

4.3 CONSIDERATIONS ABOUT CLOUD MODELS RESULTS

For the stationary analysis results, we noted that availability increased considerably, from 99.7607% for Tier I to 99.9021% for Tier IV. This increase represents a downtime decrease from 20.96h to 8.58h per year, which can greatly improve the application performance. Therefore, an application with high availability requirements (e.g. critical applications), needs to be run on architectures with redundant components that provides a highest availability level, such as the Tier IV.

About the sensitivity analysis performed in this chapter, the components with highest impact on availability were the edge router for Tier I, and the server for Tier IV. The changes in the critical fault point in these two scenarios, is related to the different level of components redundancy. In order to improve the architectures' availability, an investment can be made in these components, either by adding redundancy or purchasing new equipment with better reliability.

5 RESULTS OF INTEGRATED SCENARIO MODELS

This chapter shows the performance results of the integration scenario of edge, fog, and cloud computing components. Firstly, we will show the results obtained from the experimental prototype. These results will be used in the performance models. Next, we will show the availability results of the integrated infrastructure models, in addition to the results regarding sensitivity analysis. Finally, the performance results will be explained, related works will be presented, and considerations about results will be made.

5.1 PROTOTYPE MEASUREMENT METHODOLOGY AND RESULTS

In summary, we have now defined three different scenarios (Figure 6), and have two different fog devices, two different network connections, and four different cloud geolocations. Furthermore, for each scenario we have multiple configurations.

Figure 21 shows the steps used to setup the devices according to each scenario used in our experiments, while Figure 22 shows the steps used to perform the experiments. Note that the interval between requests was set up to two seconds, totaling 102 requests for each experiment.

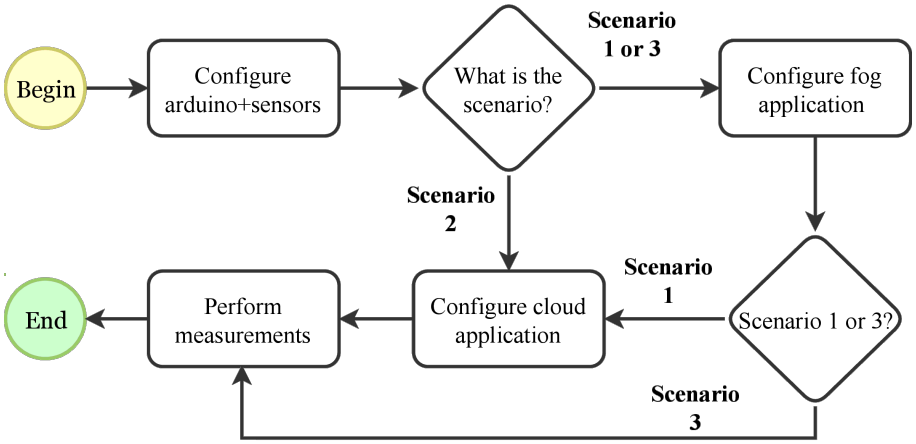


Figure 21 – Methodology for configuring devices according to each scenario

Table 13 depicts the average time (and the standard deviation) obtained from our measurements. In Scenario 1, one can see that the geographic location of the cloud instances impacts performance. For example, the Netbook connected to the São Paulo instance using the Ethernet option had a mean delay of 197.84 ms, while the mean service time for the Tokyo instance was 586.02 ms. In this case, there is an increase of 196.20% due to geographical distance. As expected, the connection type also impacted delay. For example, with the Netbook connected to the cloud instance located in São Paulo, the delay increases 31.83% when we changed the Ethernet connection to IEEE 802.11 (corresponding

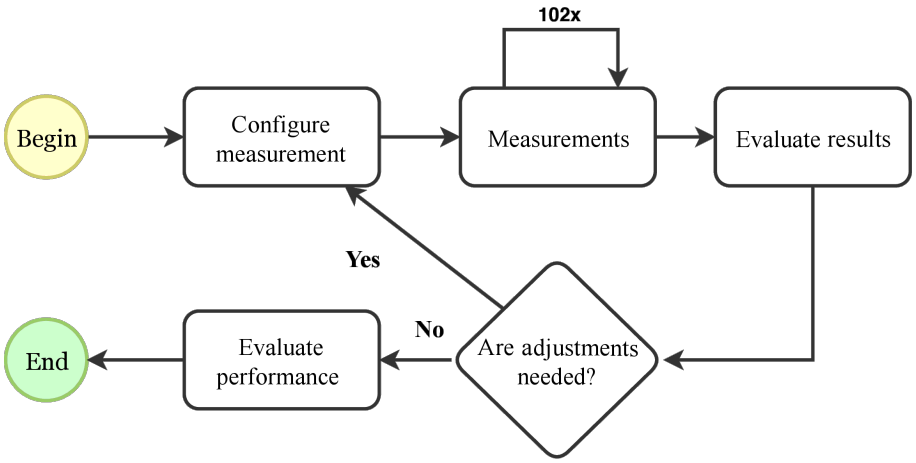


Figure 22 – Methodology for performing measurements with the prototype

to a change from 197.84 ms to 260.81 ms respectively). However, when the Raspberry Pi was used as a fog device, the impact of the network connection diminished. Delay from the Raspberry Pi using the Ethernet to a cloud instance located in São Paulo was 207.15 ms, while using IEEE 802.11 was 215.61 ms, an increase of only 3.92%. The largest impacts recorded related to the Netbook can be explained by the network card.

In Scenario 2, the same geographic impact is noted. However, once there is a direct connection between the microcontroller (Arduino) and the cloud instances, the delay is lower than in Scenario 1 since there is an intermediary fog node. As expected, the greater the distance from the microcontroller to the cloud instance, the longer the delay. From an instance located in São Paulo to an instance located in Tokyo, the delay increased 215.45% from 179.76 ms to 567.06 ms.

Scenario 3 behaves similarly to Scenario 1 regarding network connection impact. The mean delay from the microcontroller to the Netbook through an Ethernet connection was 66.14 ms, while through IEEE 802.11 was, on average, 76.37 ms, experiencing an increase of 15.46%. The Raspberry Pi mean delay increase was similar, going from 67.77 ms to 75.50 ms (increase of 11.40%) using the Ethernet and IEEE 802.11 network connection respectively.

In general, it is possible to note that the Netbook has a superior performance than the Raspberry Pi. In both Scenarios 1 and 3, considering both types of network connection, the service time for the Netbook was lower than that of the Raspberry Pi. This is due to the fact that the computational capacity of the Netbook is superior to that of the Raspberry Pi. In addition, IEEE 802.11 had a high standard deviation average service time when compared to the Ethernet connection. This can be explained by the dynamic interference with physical objects, high packet loss, and the overhead of the collision avoidance mechanism present in wireless connections (SAGARI; SESKAR; RAYCHAUDHURI, 2015).

Table 13 – Results from prototype experiments

Scenario configuration	**Average time (in ms)**	**Standard deviation**
Scenario 1 : Netbook (Ethernet) → São Paulo	197.84	4.6487
Scenario 1 : Netbook (Ethernet) → California	432.04	10.4320
Scenario 1 : Netbook (Ethernet) → London	455.98	98.8941
Scenario 1 : Netbook (Ethernet) → Tokyo	586.02	13.39002
Scenario 1 : Netbook (IEEE 802.11) → São Paulo	260.81	194.4324
Scenario 1 : Netbook (IEEE 802.11) → California	547.54	316.4965
Scenario 1 : Netbook (IEEE 802.11) → London	574.57	269.2531
Scenario 1 : Netbook (IEEE 802.11) → Tokyo	667.97	332.1957
Scenario 1 : Raspberry Pi (Ethernet) → São Paulo	207.15	105.0747
Scenario 1 : Raspberry Pi (Ethernet) → California	440.33	70.7183
Scenario 1 : Raspberry Pi (Ethernet) → London	446.88	6.5417
Scenario 1 : Raspberry Pi (Ethernet) → Tokyo	588.56	12.4967
Scenario 1 : Raspberry Pi (IEEE 802.11) → São Paulo	215.61	24.5928
Scenario 1 : Raspberry Pi (IEEE 802.11) → California	456.57	17.9215
Scenario 1 : Raspberry Pi (IEEE 802.11) → London	471.14	17.5994
Scenario 1 : Raspberry Pi (IEEE 802.11) → Tokyo	611.72	20.2465
Scenario 2 : São Paulo	179.76	11.9883
Scenario 2 : California	414.38	14.9591
Scenario 2 : London	426.85	8.6601
Scenario 2 : Tokyo	567.06	15.1969
Scenario 3 : Netbook (Ethernet)	66.14	0.8167
Scenario 3 : Netbook (IEEE 802.11)	76.37	15.7349
Scenario 3 : Raspberry Pi (Ethernet)	67.77	1.4624
Scenario 3 : Raspberry Pi (IEEE 802.11)	75.5	4.0961

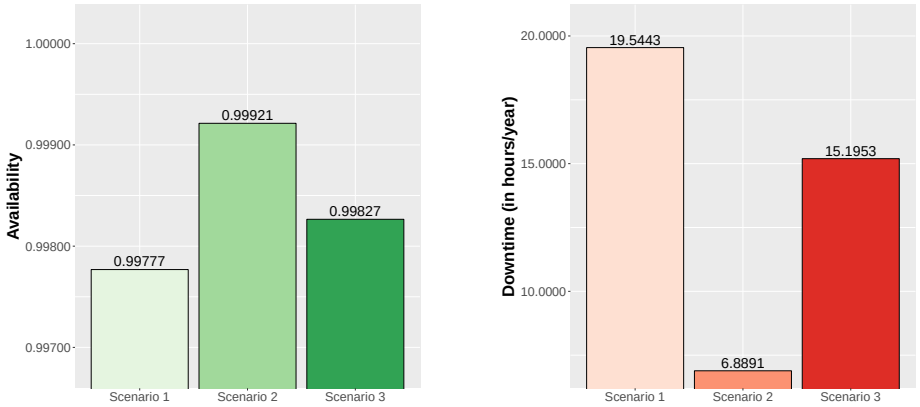
5.2 AVAILABILITY RESULTS

To analyse our models, we aligned each component's MTTF and MTTR values in line with existant literature. The MTTF and MTTR values of fog device hardware (*HW_Fog*) were estimated through an average of hardware values found in (ARAUJO et al., 2014a; MATOS et al., 2017; SILVA et al., 2013; TANG et al., 2004; KIM; MACHIDA; TRIVEDI, 2009), because the values for the specific hardware used were not available. All the values we used to set our models are described in Table 14. The cloud MTTF and MTTR values are obtained from the cloud models previously described in Subsection 3.2.1. This time we used only the Tier IV of cloud data center because this configuration achieves the highest availability value.

The availability and downtime results for each scenario are presented in Figure 23. Scenario 2 presents the best availability level (0.9992%) in comparison with Scenarios 1 (0.9978%) and 3 (0.9983%). This corresponds to Scenario 2 having 7.008 hours/year of downtime, while Scenario 3 experiences 14.892 hours/year, and Scenario 1 suffers 19.272 hours/year. Scenario 1 presents the lowest availability because all the components are in

Table 14 – MTTF and MTTR of components in hours (values obtained from (ARAUJO et al., 2014a), (SILVA et al., 2013),(TANG et al., 2004), (KIM; MACHIDA; TRIVEDI, 2009) (NOVACEK,), (BALC et al., 2017))

Component	**MTTF (h)**	**MTTR (h)**
HW_Cloud	1177.32	0.59
HW_Fog	4765.79	3.47
OS	1440	1
VM	2880	0.17
Microcontroller	44957	5
Sensor	28011	5



(a) (b)

Figure 23 – (a) Availability levels and (b) downtime regarding the e-health monitoring system considering the three scenarios proposed.

series; all components are essential to e-health monitoring system operation.

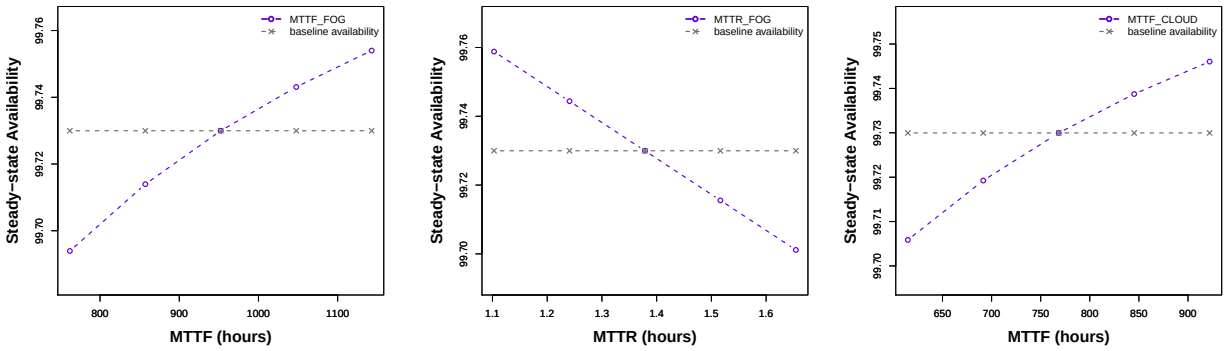
5.3 SENSITIVITY ANALYSIS

For the sensitivity analysis, we followed a similar methodology presented in (MATOS et al., 2015). We used the MTTF and MTTR values from our system's components as parameters, varying them in ten values within a range defined by maximum and minimum values (10% plus and minus the default value). Table 15 shows the top three components (and their respective sensitivity index) that most impact system availability.

Figure 24 depicts the availability variation of those three parameters. As expected, when we increase the MTTF value (see Figures 24.a and 24.c), the availability also increases, but in this case, the fog device MTTF has more impact than the cloud MTTF. An increase of 20% in fog device MTTF results in a reduction of 2.1051 hours in annual downtime, while the same increase in cloud MTTF results in a smaller decrease in downtime, i.e., only 1.4088 hours.

Table 15 – Indexes of the three paramaters that affect more the metric of each scenario

Scenario 1		**Scenario 2**		**Scenario 3**	
Parameter	**Index**	**Parameter**	**Index**	**Parameter**	**Index**
MTTF_Fog	2.64×10^{-4}	MTTF_Cloud	1.95×10^{-4}	MTTF_Fog	2.65×10^{-4}
MTTR_Fog	2.57×10^{-4}	MTTR_Cloud	1.93×10^{-4}	MTTR_Fog	2.57×10^{-4}
MTTF_Cloud	1.95×10^{-4}	MTTR_Sensors	3.57×10^{-5}	MTTR_Sensors	3.57×10^{-5}

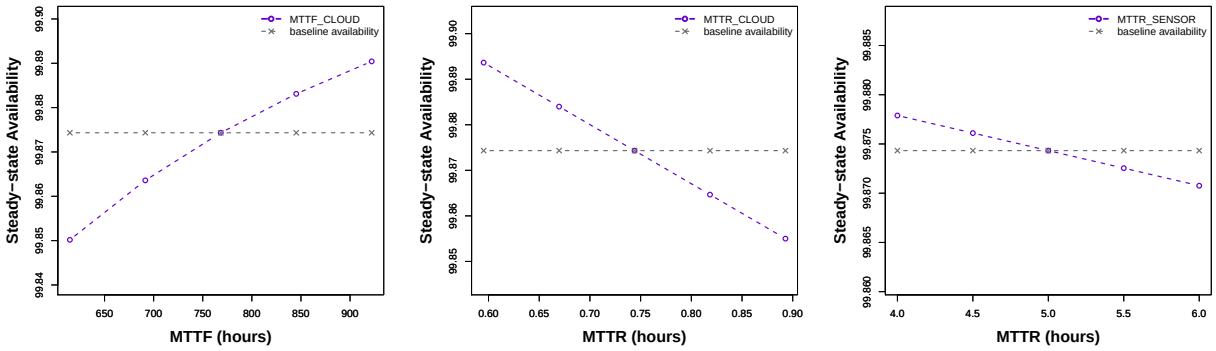


(a) (b) (c)

Figure 24 – Availability results for scenario 1 varying (a) the fog device MTTF, (b) the fog device MTTR and (c), the cloud MTTF.

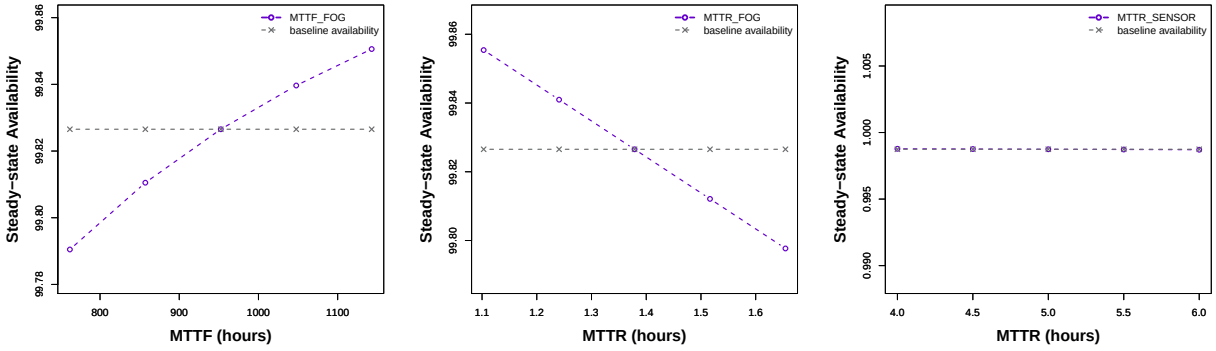
Scenarios 2 and 3 (Figures 25 and 26, respectively) behave similarly due to their relies on only one application instance; scenario 2 considers only the cloud while scenario 3 only considers fog presence. In scenario 2, the three parameters that have the greatest impact on system availability are the MTTF and MTTR of Cloud and the MTTR of the sensors. A variation in MTTF of cloud results in a reduction in annual downtime of 1.4108 hours, while the same variation in MTTR of sensors results in a minor reduction in annual downtime, 0.3123 hours to be precise. In scenario 3 the MTTF and MTTR of fog are those with greater impact on system availability. An increase of 20% in the MTTF of the fog devices results in a decrease of 2.1071 hours, while the same increase in the sensor's MTTR results in a decrease of 0.0312 hours.

Downtime in e-health systems may vary between few minutes and 16 hours (WANG et al., 2016), but unplanned downtime greater than eight hours are more commons (SITTIG; GONZALEZ; SINGH, 2014). Several incidents happens due to these downtime, e.g., "*a hospital-wide system breakdown delayed post-surgery treatment leading to a permanent musculoskeletal disability. In another case, a patient died when a network problem delayed transmission of images for diagnosis*" (WANG et al., 2016). Thus, any reduction in the downtime is important to avoid complications on the patients health.



(a) (b) (c)

Figure 25 – Availability results of scenario 2 varying (a) the cloud MTTF, (b) the cloud MTTR and (c), the sensor MTTR.



(a) (b) (c)

Figure 26 – Availability results of scenario 3 varying (a) the fog MTTF, (b) the fog MTTR and (c), the sensor MTTR.

5.4 PERFORMANCE RESULTS

We perform stationary analysis on our performance model. The service time used in the analysis was set up with delays measured through our prototype experiments (Table 13), while the time of requests arrival was set to two seconds. We use two seconds to ensure the stability of the queue system, the time between arrival must be greater than the time service (BOLCH et al., 2006), thus we used a time greater than all services times measured using the prototype.

Table 16 presents the performance results obtained from the analysis of our model. One can note that changing the geographic location of the cloud instance impacts the performance metrics evaluated. For example, let's consider the Netbook sending data using the Ethernet. To send data to the cloud instance located in São Paulo, the throughput was 15,738,614.69 requests/year. To send to the instance located in Tokyo was 15,737,248.83

requests/year. Under this same configuration, the service time increases from 699.43 ms to 1,352.76 ms.

Table 16 – Performance results of scenario 1

Fog device → Network connection → Cloud geo-location	**Throughput (requests/year)**	**Service Time (ms)**
Netbook → Ethernet → São Paulo	15,738,614.69	699.43
Netbook → Ethernet → California	15,737,730.46	1,057.48
Netbook → Ethernet → London	15,737,656.95	1,099.75
Netbook → Ethernet → Tokyo	15,737,248.83	1,352.76
Netbook → IEEE 802.11 → São Paulo	15,738,343.99	787.10
Netbook → IEEE 802.11 → California	15,737,365.47	1,273.35
Netbook → IEEE 802.11 → London	15,737,283.05	1,328.70
Netbook → IEEE 802.11 → Tokyo	15,737,003.76	1,536.62
Raspberry Pi → Ethernet → São Paulo	15,738,571.75	712.02
Raspberry Pi → Ethernet → California	15,737,708.86	1,071.99
Raspberry Pi → Ethernet → London	15,737,681.55	1,083.50
Raspberry Pi → Ethernet → Tokyo	15,737,240.85	1,358.17
Raspberry Pi → IEEE 802.11→ São Paulo	15,738,524.82	723.57
Raspberry Pi → IEEE 802.11 → California	15,737,651.90	1,100.75
Raspberry Pi → IEEE 802.11 → London	15,737,605.94	1,127.11
Raspberry Pi → IEEE 802.11 → Tokyo	15,737,172.70	1,408.16

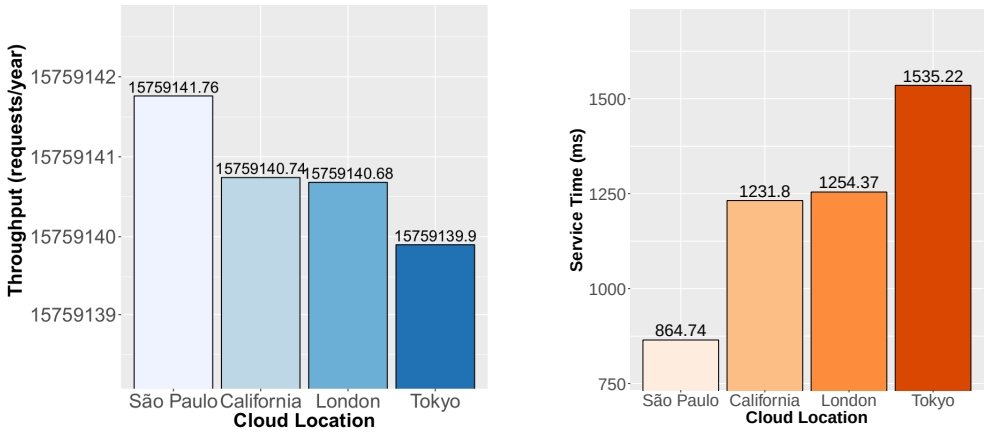
The change of connection type had a low impact on the metrics evaluated. For instance, for messages sent from the Netbook to the instance located in São Paulo, the throughput values using the Ethernet and IEEE 802.11 were 15,738,614.69 requests/year and 15,738,343.99 requests/year respectively; the service times were 699.43 ms and 787.10 ms respectively. In general and as expected, the Netbook had a superior performance than the Raspberry Pi due to its superior hardware configuration.

Figure 27 and Table 17 present the results of scenario 2. As in scenario 1, the geographic locations of cloud instances also impacted the performance metrics. The throughput for the instance located in São Paulo was 15,759,141.76 requests/year and decreased as the distance of the instance location increased, reaching 15,759,139.90 requests/year for the instance located in Tokyo. The service time was also impacted, from 864.74 ms for the instance located in São Paulo and 1,535.22 ms for instance located in Tokyo.

Table 17 – Performance results of scenario 2

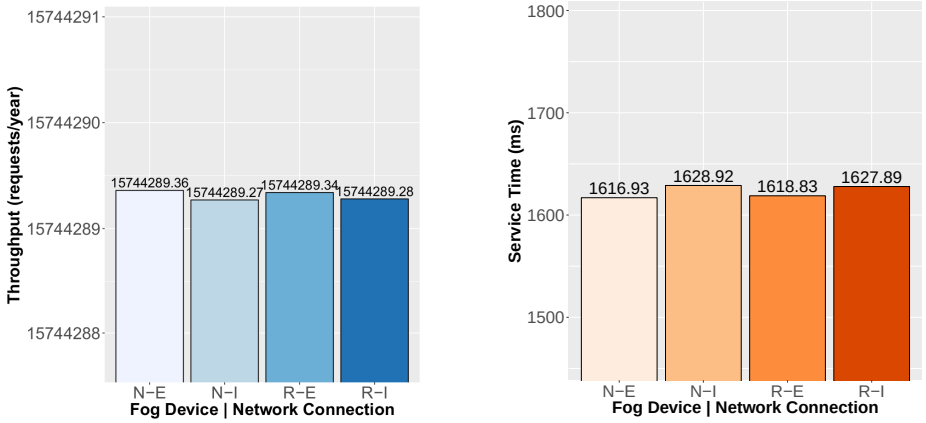
Cloud geo-location	**Throughput (requests/year)**	**Service Time (ms)**
São Paulo	15,759,141.76	864.74
California	15,759,140.74	1,231.80
London	15,759,140.68	1,254.37
Tokyo	15,759,139.90	1,535.22

Finally, Figure 28 and Table 18 present the performance results for scenario 3. In this scenario, a significant impact was identified related to the type of network connection. For



(a) (b)

Figure 27 – Scenario 2 performance results: (a) throughput, and (b) service time.



(a) (b)

Figure 28 – Scenario 3 performance results: (a) throughput, and (b) service time.

example, the throughput of the Netbook using the Ethernet was 15,744,289.36 requests/year, while through IEEE 802.11 was 15,744,289.27 requests/year. A similar impact was identified with the service time, i.e., 1,616.93 ms and 1,628.92 ms. The Netbook and Raspberry Pi display similar behaviors as they both experienced a close delay in prototype experiments.

Table 18 – Performance results of scenario 3

Fog device → Network connection	**Throughput (requests/year)**	**Service Time (ms)**
Netbook → Ethernet (N-E)	15,744,289.36	1,616.93
Netbook → IEEE 802.11 (N-I)	15,744,289.27	1,628.92
Raspberry Pi → Ethernet (R-E)	15,744,289.34	1,618.83
Raspberry Pi → IEEE 802.11 (R-I)	15,744,289.28	1,627.89

The processing capacity of the fog device (place 3_$Fog_Capacity$) and cloud server

(place 3_*Cloud_Capacity*) was limited in our models (600 and 1000, respectively) to avoid state explosion when solving the model analytically. Nonetheless, we noted that increasing the capacity to process simultaneous requests substantially neither affects the throughput nor the service time metrics. For values greater than 600 and 1000, the impact on analyzed metrics was almost insignificant.

For instance in scenario 2, when we set fog and cloud capacity as 600 and 1000 respectively, the Markov chain was composed of 24,032 states and the result was 1799.3254 requests/hour and 1,269.2155 ms for throughput and service time respectively. When we use 1,200 and 2,000, the Markov chain was composed of 48,032 states resulting in 1799.4367 requests/hour and 1,412.8086 ms for throughput and service time respectively).

5.5 RELATED WORKS

Some works have proposed solutions to deal with IoT applications integrated with fog and cloud computing. For instance, an analytic model is used in (LI et al., 2017) to decide where to process the data obtained from the IoT devices considering renewable energy consumption and Quality of Service (QoS) of the application. To validate their model, authors presented a video streaming analysis application, where cars in a road have cameras that capture data from the road and send this to the edge/cloud and one of those returns to the driver the conditions of the road.

In (SOUZA et al., 2016), authors propose a resource allocation strategy in the IoT context taking into account the offered services of fog and cloud, the energy consumption balance between them, and delay; and they modeled a strategy to analyze those parameters.

Further, a number of studies presented models to represent applications in health use case scenarios. Authors in (ZENG; KOUTNY; WATSON, 2015) proposed a model to represent and evaluate the security of information flow in IoT systems integrated with cloud, using a medical application as an example. They analyzed how the availability of the service providers affects the security of the information flow.

Authors in (ZENG; KOUTNY; WATSON, 2015) use a medical application as a case study for their proposed model that analyzes the security of the information flow in IoT systems integrated with cloud infrastructures. In (COLOM et al., 2017), authors propose a framework that enables multiple applications to share IoT computational devices for health monitoring. The use case scenario in (LOMOTEY; PRY; SRIRAMOJU, 2017) is one with a wearable IoT architecture for health care systems. In (ARAUJO et al., 2014b), authors proposed stochastic models to represent a health service relying on mobile cloud computing infrastructure (i.e., cloud infrastructure, wireless communication and mobile device). They made experiments considering scenarios with different wireless communication channels (Wi-Fi and 4G), different battery discharge rates, and different timeouts.

The work presented in this dissertation differs from the literature because we considered an integrated system (with IoT, fog and cloud) and modeled it to evaluate the availability and performance of an e-health monitoring system and analyzing it under three different scenarios. We provided a sensitivity analysis to understand how different values of MTTF and MTTR impact the availability of the whole system, and we also carried out performance analysis to obtain real data to use as input to our models.

5.6 CONSIDERATIONS ABOUT INTEGRATED RESULTS

From the stationary analysis, we evaluated availability in each scenario. We noted that the best availability level (0.9987%) was achieved in scenario 2 (where only the cloud application is considered) and the worst one (0.9973%) was scenario 1 (using both fog and cloud). It is an interesting result since in scenario 1 we have more computational layers and consequently more devices can fail (and thereby decreasing the availability of the whole system). Thus, despite the extension of fog node capability by the cloud, this integrated scenario is more complex and as a result has more points of failures despite the hypothesized performance benefits. The Scenario 2 presents the highest availability level because the cloud data center environment presents a higher availability level than the fog infrastructure.

Regarding the sensitivity analysis results, there is greater variability in the components that impact the availability of e-health monitoring system in the scenarios examined. In Scenario 1, the most critical component is the fog device whereas in Scenarios 2 and 3, the fog device and cloud infrastructure impact the system respectively. An increase of 20% in MTTF value of fog device (Scenario 1) results in a reduction of 2.1051 hours in annual downtime. An increase of 20% in MTTF of cloud infrastructure results in reduction of 1.4108 hours of annual downtime (Scenario 2). And in Scenario 3, an increase of 20% in MTTF of fog devices results in reduction of 2.1071 hours of annual downtime. Depending on the configuration chosen for the e-health monitoring system, investments can be made to increase availability, or by adding more redundant equipment provide greater reliability.

Within the prototype experiments, we noted an increase of delay as the distance from microcontroller/fog devices to cloud instance increases both in Scenario 1 and 2, as expected. The delay in Scenario 1 was slightly higher than Scenario 2 once the fog devices are added. However, the addition of fog devices (Scenario 1) enables the pre-analysis of data collected by sensors before sending to the cloud. Thus, simple decisions, such as calling an ambulance, can be made quicker.

We also observed that the delay to send data from microcontroller to fog devices is significantly lower than when sending data to the cloud. Considering the closer proximity of the the cloud instance to the microcontroller (São Paulo), the mean delay was 179.76 ms, while the higher mean delay to send to the cloud was 76.37 ms (Netbook with IEEE 802.11). For delay-sensitive systems (e.g. e-health monitoring systems, augmented reality,

real-time video analytics, and content delivery (YI; LI; LI, 2015)), fog devices can greatly reduce the delay and increase the performance of these systems. However, one should not disregard the limited computing capacity of these devices.

These experiment results were used to feed the performance model. For Scenarios 1 and 2, as the geographic distance of the cloud instance increases, the throughput decreases (see Tables 16 and 17). Once the time to process a request increases, fewer requests will be processed because they will remain in the queue for longer (see Figure 16). In addition, service time is directly impacted by the geographic location of an instance i.e. the longer it takes to send the request, the longer it will take to process the request and to receive a response. For an e-health monitoring application, where some decisions need to be made quickly, hosting the cloud instance in a remote geographic region may have a significant impact on service time and associated QoS levels.

Scenario 1 had the lowest availability level and the worst performance results. This can be explained as follows. The e-health system is considered operational only when all components of the scenario are working and this reduces the availability of the system as a whole. Relatively poor performance results are due to the high delay in sending data from the microcontroller to the cloud passing through a fog device. Notwithstanding this, this scenario can take advantage of the computing capacity and virtually unlimited data storage that cloud computing offers.

Scenario 2 presented the best availability level and better performance results than others. The improvement in performance is due to the decrease in delay because the data is sent directly from the microcontroller to the cloud (without fog devices). Notwithstanding this, the delay in Scenario 2 can compromise systems that are delay-sensitive.

Scenario 3 had lower availability results than Scenario 2, because the availability of fog devices is lower than cloud infrastructure. Similar to Scenario 1, the availability has an impact in throughput, decreasing the number of requests processed when compared to Scenario 2. The service time is lower than other scenarios, since delay to send data to fog devices is lower. However, fog devices have lower computational capacity than cloud devices, so this scenario may not be the most appropriate for systems that handle large amounts of data.

6 CONCLUSION AND FUTURE WORKS

Fog and cloud computing address a number of problems encountered in IoT however they also increase management complexity. Despite fog and cloud computing offering greater availability and resilience, they can also be viewed as vulnerabilities or potential points of failure. As such, in addition to edge device failure, attention must be paid to fog node and cloud infrastructure failures. While cloud and fog integration is relatively well known and shares common technologies, the integration/extension with IoT is a non-trivial task, mostly due to massive device heterogeneity and service requirements.

Some applications have high criticality and any downtime can lead in extreme cases to loss of life, as in the case of an e-health monitoring system, the focus of this dissertation. The data collected by these applications are very critical, once that are related to vital conditions of patients. Therefore, any downtime of application (that have different roots, such as hardware or software failures) may compromise the operation of the application.

In this dissertation, we proposed a set of stochastic models to evaluate IoT applications that rely on edge, fog, and cloud infrastructures. We consider the e-health scenario due him sensitivity to availability. This service was evaluated regarding its availability and performance. We proposed stochastic models by using SPN and RBD approaches. To use realistic data as an input to our models, we also developed and implemented a prototype with different types of fog device, network connection, and cloud instance geographic location configurations. Firstly we propose a set of models to evaluate the cloud data center infrastructure where the applications be hosted. After we propose models that integrate edge, fog, and cloud infrastructures to perform analyses about availability. Finally we propose models to asses the performance applications.

Cloud results showed that availability increases from Tier I to Tier IV. With sensitivity analysis were evaluated the components that most impact the availability, and how a variation in parameters of components impact the availability of data center. The Tier I model achieved approximately 99.76% of availability, while Tier IV achieved 99.90%. Sensitivity analysis showed that the component that most impacts on the availability was the edge router for Tier I architecture, and servers for Tier IV. Thus, to improve the availability, one can make an investment in these components, whether buying from more reliable equipment or adding redundancy. Experiments showed a great variation of data center availability when the values of these metrics were changed.

From integrated results, it is clear that there is a trade-off between performance and service time. In this way, it is necessary to prioritize the consideration of the application requirements before deciding on the best architecture. This is illustrated by the results from the various scenarios. For example, the scenario that relies only on cloud infrastructure (Scenario 2) presents the best availability level since cloud providers can offer

a better service as measured by reliability than fog devices. On the other hand, Scenario 1, which relies on fog devices and cloud infrastructure, may be more appropriate to host e-health monitoring systems due to the technological limitations of end-devices. Other configurations may be more appropriate depending on the use scenario e.g. big data services.

The main contributions of this dissertation are:

- **stochastic models** to evaluate cloud data center infrastructures based on TIA-942 standard. It is possible analyze these models to identify most critical components through sensitivity analysis;
- **stochastic models** to understand how failures in edge devices, fog devices, and/or cloud infrastructure impacts e-health monitoring **system availability**. These models will also be used to perform **sensitivity analysis** to understand which components have a significant impact on the e-health monitoring system availability;
- a **prototype** to conduct performance evaluations in order to feed the stochastic models with real data. This prototype is composed of two different configurations of fog devices, two different network connections to access the cloud instance, and four different geo-locations of cloud instance in order to characterize the heterogeneity of I2C-based systems; and
- **stochastic performance models** integrated with the availability models, and real data outputted from the prototype in order to understand how different capacity of fog devices and also different geo-location of cloud instances impact on performance metrics, such as throughput and service time.

6.1 DIFFICULTIES FOUND

Several difficulties were found in the realization of this study. It is difficult to obtain data related equipment specification (more precisely MTTF and MTTR values). These data are very scarce in the literature, mainly for hardware devices, for example the MTTF and MTTR of fog devices. Another difficulty is related to scaling our prototype. We use instances in Amazon AWS, and, as any cloud payment model, you pay as you use. So, the more instances we use, the more expensive our prototype will be. In addition, buying the devices used in prototype on a large scale may be impractical. Another difficulty is related to availability model validation. The validation of such models could be done by creating a prototype and exposing this prototype and model to the same considerations in order to verify if they behave similarly.

6.2 LIMITATIONS OF WORK

This dissertation has some limitations regarding the solutions presented. In cloud data center models, we disregard some components that we consider that not affect the availability, such as the large frame processing, with the purpose of simplifying our models. Another limiting aspect is about the modeled servers. We assume that they are similar. In others words, we consider that all servers present in cloud infrastructure have the same hardware and software configurations. This aspect can be observed too in our integrated model. We consider that all fog nodes and edge devices are similar. It is important to mention that is possible to consider heterogeneous devices in our models, but they will become more complex, once it is necessary to include more places and transitions in our SPN models to represent them. Consequently, we made that assumption in order to simplify our models. A third limitation has to do with our prototype. We consider a web application that may be hosted in cloud or fog devices to process the data collected by sensors (e.g. data mining, neural networks, machine learning algorithms). However, building the actual application is considered outside of scope of this dissertation. We consider only the web application that receive the data. So, the service time for this application is only the time to receive and respond to a request.

6.3 PUBLICATIONS

The Table 19 presents the scientific papers produced in scope of this dissertations, including papers published and submitted. The Table 20 shows others publications produced during the dissertation production.

Table 19 – Scientific papers produced

#	Reference	Type	Status	Qualis
1	**Santos, G. L.**, Endo, P. T., Gonçalves, G., Rosendo, D., Gomes, D., Kelner, J., ... & Mahloo, M. (2017, July). Analyzing the IT subsystem failure impact on availability of cloud services. In Computers and Communications (ISCC), 2017 IEEE Symposium on (pp. 717-723).	Conference	Published	A2
2	Rosendo, D., **Leoni, G.**, Gomes, D., Moreira, A., Gonçalves, G., Endo, P., ... & Mahloo, M. (2018, January). How to Improve Cloud Services Availability? Investigating the Impact of Power and It Subsystems Failures. In Proceedings of the 51st Hawaii International Conference on System Sciences.	Conference	Published	A1
3	Tigre, M.F.F.L.S, **Santos, G. L.**, Lynn, T. Sadok, D., Kelner, J., & Endo, P. T. (2018, June). Modeling the availability of an e-health system integrated with edge, fog and cloud infrastructures. In Computers and Communications (ISCC), 2018 IEEE Symposium on (pp. 717-723).	Conference	Published	A2
4	Endo, P. T., **Santos, G. L.**, Rosendo, D., Gomes, D. M., Moreira, A., Kelner, J., ... & Mahloo, M. (2017). Minimizing and Managing Cloud Failures. Computer, 50(11), 86-90.	Journal	Published	A1
5	**Santos, G.L.**, Endo, P. T., Tigre, M.F.F.L.S, Ferreira, L., Sadok, D., Kelner, J., Lynn, T. (2018). Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures. Journal of Cloud Computing 7.1 (2018): 16.	Journal	Published	B2
6	Endo, P. T., Gonçalves, G. E., Rosendo, D., Gomes, D., **Santos, G. L.**, Moreira, A. L. C., ... & Mahloo, M. (2017). Highly Available Clouds: System Modeling, Evaluations, and Open Challenges. In Research Advances in Cloud Computing (pp. 21-53). Springer, Singapore.	Chapter book	Published	-
7	**Santos, G. L.**, Ferreira, M., Ferreira, L., Kelner, J., Sadok, D., Albuquerque, E., Lynn & T., Endo, P. T (2018). Integrating IoT + fog + cloud infrastructures: System modeling and research challenges. In : Fog and Edge Computing: Principles and Paradigms. Springer.	Chapter book	Published	-

Table 20 – Other related publications

#	Reference	Type	Status	Qualis
1	Rocha, É., Endo, P. T., **Leoni, G.**, Braga, J., & Lynn, T. (2017, October). Analyzing the impact of power infrastructure failures on cloud application availability. In Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on (pp. 1746-1751). IEEE.	Conference	Published	A2
2	**Santos, G.L.**, Gomes, D., Sadok, D., Kelner, J., Rocha, E., & Endo, P. T. (2018). How Do Checkpoint Mechanisms and Power Infrastructure Failures Impact on Cloud Applications? International Journal of Grid and Utility Computing (IJGUC).	Journal	Accepted	B2
3	Rocha, E. S. **Santos, G.L.**, & Endo, P. T. (2018). Analyzing the impact of power subsystem failures and checkpoint mechanisms on availability of cloud applications. Revista do IEEE América Latina.	Journal	Under Review	B4
4	**Santos, G.L.**, Gomes, D., Sadok D., Kelner, J., Silva, J. A., Endo, P. T. & Lynn T. (2018). The Internet of Medical Things: Optimizing e-health system availability based on edge, fog and cloud infrastructures. International Journal of Computational Science and Engineering.	Journal	Under Review	B1
5	**Santos, G.L.**, Gomes, D., Kelner, J., Sadok D., Silva, J. A., Endo, P. T. & Lynn T. (2018). Maximizing System Availability with BudgetConstraints in the Internet of Medical ThingsThrough Nature-Inspired Approaches. IEEE Internet of things journal.	Journal	Under Review	B2

6.4 FUTURE WORKS

As future works, we plan to analyze the impact of redundancy and understand how we can improve the availability of the system by adding more cloud servers or fog devices. We also plan increase the complexity of model to consider more elements, such as different sensors, fog devices, and the connectivity between the devices. We can also increases the

complexity of the cloud data center models, considering the power and cooling subsystems, and how they impact the availability of the cloud service hosted.

We also plan to evolve the prototype system with different types of smart-end devices, fog devices, and analytical techniques, including the use of machine learning to treat the data collected by edge devices, for example deep learning. With the prototype, new experiments can be performed, considering the complete service time to evaluate and store the patient data. We can also evaluate different communication protocols, such s MQTT and CoAP, and different technologies, such as Bluetooth Low Energy, ZigBee, Lora, and so on. We also plan evaluate the cost of the prototype as we increase the number of components with the intention of verifying the feasibility of implanting the system, since it would be interesting a solution of low cost.

Finally, we plan use optimization algorithms to maximize the e-health system availability taking into account some constraints such as a limited budget and the cost of components.

REFERENCES

ADHIKARY, T.; DAS, A. K.; RAZZAQUE, M. A.; ALRUBAIAN, M.; HASSAN, M. M.; ALAMRI, A. Quality of service aware cloud resource provisioning for social multimedia services and applications. *Multimedia Tools and Applications*, Springer, v. 76, n. 12, p. 14485–14509, 2017.

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, IEEE, v. 17, n. 4, p. 2347–2376, 2015.

AL-HAIDARI, F.; SQALLI, M.; SALAH, K. Evaluation of the impact of edos attacks against cloud computing services. *Arabian Journal for Science and Engineering*, Springer, v. 40, n. 3, p. 773–785, 2015.

AMENDOLA, S.; LODATO, R.; MANZARI, S.; OCCHIUZZI, C.; MARROCCO, G. Rfid technology for iot-based personal healthcare in smart spaces. *IEEE Internet of things journal*, IEEE, v. 1, n. 2, p. 144–152, 2014.

ANDRADE, E.; NOGUEIRA, B. Performability evaluation of a cloud-based disaster recovery solution for it environments. *Journal of Grid Computing*, Springer, p. 1–19, 2018.

ANDRADE, E.; NOGUEIRA, B.; MATOS, R.; CALLOU, G.; MACIEL, P. Availability modeling and analysis of a disaster-recovery-as-a-service solution. *Computing*, Springer, p. 1–26, 2017.

ARAUJO, J.; MACIEL, P.; TORQUATO, M.; CALLOU, G.; ANDRADE, E. Availability evaluation of digital library cloud services. In: IEEE. *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*. [S.l.], 2014. p. 666–671.

ARAUJO, J.; SILVA, B.; OLIVEIRA, D.; MACIEL, P. Dependability evaluation of a mhealth system using a mobile cloud infrastructure. In: IEEE. *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. [S.l.], 2014. p. 1348–1353.

ARDAGNA, D.; CASALE, G.; CIAVOTTA, M.; PÉREZ, J. F.; WANG, W. Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, v. 5, n. 1, p. 11, 2014.

ASSOCIATION, T. I. et al. Tia-942 data center standards overview. *White Paper*, 2006.

AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B. et al. *Fundamental concepts of dependability*. [S.l.]: University of Newcastle upon Tyne, Computing Science, 2001.

BAKER, T.; MACKAY, M.; RANDLES, M.; TALEB-BENDIAB, A. Intention-oriented programming support for runtime adaptive autonomic cloud-based applications. *Computers & Electrical Engineering*, Elsevier, v. 39, n. 7, p. 2400–2412, 2013.

BALC, C.; CRETU, A.; MUNTEANU, R.; IUDEAN, D.; BALAN, H.; KARAISAS, P. Reliability modeling for an automatic level control system. In: IEEE. *Optimization of Electrical and Electronic Equipment (OPTIM) & 2017 Intl Aegean Conference on Electrical Machines and Power Electronics (ACEMP), 2017 International Conference on*. [S.l.], 2017. p. 995–1000.

BALSAMO, S.; MARIN, A.; STOJIC, I. Deriving the performance indices in product-form stochastic petri nets: open problems and simulation. In: *EUROSIS European Simulation Multiconference*. [S.l.: s.n.], 2015.

BARROSO, L. A.; CLIDARAS, J.; HÖLZLE, U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, Morgan & Claypool Publishers, v. 8, n. 3, p. 1–154, 2013.

BAUSE, F.; KRITZINGER, P. Stochastic petri nets. *Verlag Vieweg, Wiesbaden*, v. 26, 1996.

BOLCH, G.; GREINER, S.; MEER, H. D.; TRIVEDI, K. S. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. [S.l.]: John Wiley & Sons, 2006.

BONOMI, F.; MILITO, R.; NATARAJAN, P.; ZHU, J. Fog computing: A platform for internet of things and analytics. In: *Big Data and Internet of Things: A Roadmap for Smart Environments*. [S.l.]: Springer, 2014. p. 169–186.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: ACM. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.], 2012. p. 13–16.

BORGIA, E. The internet of things vision: Key features, applications and open issues. *Computer Communications*, Elsevier, v. 54, p. 1–31, 2014.

BORTHAKUR, D.; DUBEY, H.; CONSTANT, N.; MAHLER, L.; MANKODIYA, K. Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things. *arXiv preprint arXiv:1712.09347*, 2017.

BOTTA, A.; DONATO, W. D.; PERSICO, V.; PESCAPÉ, A. On the integration of cloud computing and internet of things. In: IEEE. *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. [S.l.], 2014. p. 23–30.

BOTTA, A.; DONATO, W. D.; PERSICO, V.; PESCAPÉ, A. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, Elsevier, v. 56, p. 684–700, 2016.

BUYYA, R.; DASTJERDI, A. V. *Internet of Things: Principles and paradigms*. [S.l.]: Elsevier, 2016.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J.; BRANDIC, I. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, Elsevier, v. 25, n. 6, p. 599–616, 2009.

CHEN, X.; HUANG, X.; JIAO, C.; FLANNER, M. G.; RAEKER, T.; PALEN, B. Running climate model on a commercial cloud computing environment: A case study using community earth system model (cesm) on amazon aws. *Computers & Geosciences*, Elsevier, v. 98, p. 21–25, 2017.

CHEN, Z.; CHANG, X.; HAN, Z.; LI, L. Survivability modeling and analysis of cloud service in distributed data centers. *The Computer Journal*, Oxford University Press, v. 61, n. 9, p. 1296–1305, 2017.

CHIUCHISAN, I.; COSTIN, H.-N.; GEMAN, O. Adopting the internet of things technologies in health care systems. In: IEEE. *Electrical and Power Engineering (EPE), 2014 International Conference and Exposition on*. [S.l.], 2014. p. 532–535.

COLOM, J. F.; MORA, H.; GIL, D.; SIGNES-PONT, M. T. Collaborative building of behavioural models based on internet of things. *Computers & Electrical Engineering*, Elsevier, v. 58, p. 385–396, 2017.

DANTAS, J.; MATOS, R.; ARAUJO, J.; MACIEL, P. An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. [S.l.], 2012. p. 1664–1669.

DANTAS, J.; MATOS, R.; ARAUJO, J.; MACIEL, P. Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. *Computing*, Springer, v. 97, n. 11, p. 1121–1140, 2015.

DESIGN, D. C. Implementation best practices. *ANSI/BICSI*, p. 002–2011, 2011.

ENDO, P. T.; GONÇALVES, G. E.; ROSENDO, D.; GOMES, D.; SANTOS, G. L.; MOREIRA, A. L. C.; KELNER, J.; SADOK, D.; MAHLOO, M. Highly available clouds: System modeling, evaluations, and open challenges. In: *Research Advances in Cloud Computing*. [S.l.]: Springer, 2017. p. 21–53.

EVANS, D. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, v. 1, n. 2011, p. 1–11, 2011.

FRANCO, J. M.; BARBOSA, R.; ZENHA-RELA, M. Availability evaluation of software architectures through formal methods. In: IEEE. *Quality of Information and Communications Technology (QUATIC), 2014 9th International Conference on the*. [S.l.], 2014. p. 282–287.

GOLDSZTAJN, D.; FERRAGUT, A.; PAGANINI, F.; JONCKHEERE, M. Controlling the number of active instances in a cloud environment. *ACM SIGMETRICS Performance Evaluation Review*, ACM, v. 45, n. 2, p. 15–20, 2018.

GRANATO, L. *Excelsior! As frases mais inspiradoras de Stan Lee para sua carreira*. 2018. Https://exame.abril.com.br/carreira/excelsior-as-frases-mais-inspiradoras-de-stan-lee-para-sua-carreira/. Accessed in: 25 jun 2019.

GUIMARAES, A. P.; MACIEL, P.; JUNIOR, R. M. Design of it infrastructures of data centers: An approach based on business and technical metrics. *Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, John Wiley & Sons, p. 265, 2015.

HAMBY, D. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment*, Springer, v. 32, n. 2, p. 135–154, 1994.

HO, G.; LEUNG, D.; MISHRA, P.; HOSSEINI, A.; SONG, D.; WAGNER, D. Smart locks: Lessons for securing commodity internet of things devices. In: ACM. *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. [S.l.], 2016. p. 461–472.

HØYLAND, A.; RAUSAND, M. *System reliability theory: models and statistical methods*. [S.l.]: John Wiley & Sons, 2009.

IORGA, M.; FELDMAN, L.; BARTON, R.; MARTIN, M. J.; GOREN, N. S.; MAHMOUDI, C. *Fog computing conceptual model*. [S.l.], 2018.

ISLAM, S. R.; KWAK, D.; KABIR, M. H.; HOSSAIN, M.; KWAK, K.-S. The internet of things for health care: a comprehensive survey. *IEEE Access*, IEEE, v. 3, p. 678–708, 2015.

JACOBSSON, A.; BOLDT, M.; CARLSSON, B. A risk analysis of a smart home automation system. *Future Generation Computer Systems*, Elsevier, v. 56, p. 719–733, 2016.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1990.

JAMMAL, M.; KANSO, A.; HEIDARI, P.; SHAMI, A. A formal model for the availability analysis of cloud deployed multi-tiered applications. In: IEEE. *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. [S.l.], 2016. p. 82–87.

KHAZAEI, H.; MISIC, J.; MISIC, V. B. Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems. *IEEE Transactions on parallel and distributed systems*, IEEE, v. 23, n. 5, p. 936–943, 2012.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. S. Availability modeling and analysis of a virtualized system. In: IEEE. *Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on*. [S.l.], 2009. p. 365–371.

KOROLYUK, V. S.; KOROLYUK, V. V. *Stochastic models of systems*. [S.l.]: Springer Science & Business Media, 2012.

KUNTZ, G. W. M. Symbolic semantics and verification of stochastic process algebras. 2006.

LI, H.; OTA, K.; DONG, M. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, IEEE, v. 32, n. 1, p. 96–101, 2018.

LI, S.; XU, L. D.; ZHAO, S. The internet of things: a survey. *Information Systems Frontiers*, Springer, v. 17, n. 2, p. 243–259, 2015.

LI, Y.; ORGERIE, A.-C.; RODERO, I.; PARASHAR, M.; MENAUD, J.-M. Leveraging renewable energy in edge clouds for data stream analysis in iot. In: IEEE. *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*. [S.l.], 2017. p. 186–195.

LIMAYE, A.; ADEGBIJA, T. Hermit: A benchmark suite for the internet of medical things. *IEEE Internet of Things Journal*, IEEE, v. 5, n. 5, p. 4212–4222, 2018.

LINDSTRÖM, J.; LÖFSTRAND, M.; REED, S.; ALZGHOUL, A. Use of cloud services in functional products: availability implications. *Procedia CIRP*, Elsevier, v. 16, p. 368–372, 2014.

LOMOTEY, R. K.; PRY, J.; SRIRAMOJU, S. Wearable iot data stream traceability in a distributed health information system. *Pervasive and Mobile Computing*, Elsevier, v. 40, p. 692–707, 2017.

LUAN, T. H.; GAO, L.; LI, Z.; XIANG, Y.; WEI, G.; SUN, L. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

M. FELDMAN R., B. R. I. et al. The nist definition of fog computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2017.

MAO, Y.; ZHANG, J.; LETAIEF, K. B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 34, n. 12, p. 3590–3605, 2016.

MARINESCU, D. C. *Cloud computing: theory and practice*. [S.l.]: Morgan Kaufmann, 2017.

MARSAN, M. A. Stochastic petri nets: an elementary introduction. In: SPRINGER. *European Workshop on Applications and Theory in Petri Nets*. [S.l.], 1988. p. 1–29.

MATOS, R.; ARAUJO, J.; OLIVEIRA, D.; MACIEL, P.; TRIVEDI, K. Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory*, Elsevier, v. 50, p. 151–164, 2015.

MATOS, R.; DANTAS, J.; ARAUJO, J.; TRIVEDI, K. S.; MACIEL, P. Redundant eucalyptus private clouds: Availability modeling and sensitivity analysis. *Journal of Grid Computing*, Springer, p. 1–22, 2016.

MATOS, R.; DANTAS, J.; ARAUJO, J.; TRIVEDI, K. S.; MACIEL, P. Redundant eucalyptus private clouds: Availability modeling and sensitivity analysis. *Journal of Grid Computing*, Springer, v. 15, n. 1, p. 1–22, 2017.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

MELO, C.; MATOS, R.; DANTAS, J.; MACIEL, P. Capacity-oriented availability model for resources estimation on private cloud infrastructure. In: IEEE. *Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on*. [S.l.], 2017. p. 255–260.

MELO, R.; BEZERRA, M. C.; DANTAS, J.; MATOS, R.; MELO, I.; MACIEL, P. Video on demand hosted in private cloud: Availability modeling and sensitivity analysis. In: IEEE. *Dependable Systems and Networks Workshops (DSN-W), IEEE International Conference on*. [S.l.], 2015. p. 12–18.

MOHSIN, M.; ANWAR, Z.; ZAMAN, F.; AL-SHAER, E. Iotchecker: a data-driven framework for security analytics of internet of things configurations. *Computers & Security*, Elsevier, 2017.

NOVACEK, G. *Tips for Predicting Product Reliability*. <http://circuitcellar.com/cc-blog/tips-for-predicting-product-reliability/>. Accessed: 2018-01-05.

PHAM, T. N.; TSAI, M.-F.; NGUYEN, D. B.; DOW, C.-R.; DENG, D.-J. A cloud-based smart-parking system based on internet-of-things technologies. *IEEE Access*, IEEE, v. 3, p. 1581–1591, 2015.

PIANOSI, F.; BEVEN, K.; FREER, J.; HALL, J. W.; ROUGIER, J.; STEPHENSON, D. B.; WAGENER, T. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environmental Modelling & Software*, Elsevier, v. 79, p. 214–232, 2016.

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. *Future Generation Computer Systems*, Elsevier, v. 78, p. 641–658, 2018.

RAHMATI, S. H. A.; AHMADI, A.; SHARIFI, M.; CHAMBARI, A. A multi-objective model for facility location–allocation problem with immobile servers within queuing framework. *Computers & Industrial Engineering*, Elsevier, v. 74, p. 1–10, 2014.

RAMIREZ, A. R. G.; GONZÁLEZ-CARRASCO, I.; JASPER, G. H.; LOPEZ, A. L.; LOPEZ-CUADRADO, J. L.; GARCÍA-CRESPO, A. Towards human smart cities: Internet of things for sensory impaired individuals. *Computing*, Springer, v. 99, n. 1, p. 107–126, 2017.

SAGARI, S.; SESKAR, I.; RAYCHAUDHURI, D. Modeling the coexistence of lte and wifi heterogeneous networks in dense deployment scenarios. In: IEEE. *Communication Workshop (ICCW), 2015 IEEE International Conference on*. [S.l.], 2015. p. 2301–2306.

SARDELLITTI, S.; SCUTARI, G.; BARBAROSSA, S. Joint optimization of radio and computational resources for multicell mobile-edge computing. *IEEE Transactions on Signal and Information Processing over Networks*, IEEE, v. 1, n. 2, p. 89–103, 2015.

SCHROEDER, B.; GIBSON, G. A. Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you? In: *FAST*. [S.l.: s.n.], 2007. v. 7, p. 1–16.

SHANG, W.; YU, Y.; DROMS, R.; ZHANG, L. Challenges in iot networking via tcp/ip architecture. *Technical Report NDN-0038. NDN Project*, 2016.

SHELBY, Z.; HARTKE, K.; BORMANN, C. The constrained application protocol (coap). 2014.

SHI, W.; CAO, J.; ZHANG, Q.; LI, Y.; XU, L. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, IEEE, v. 3, n. 5, p. 637–646, 2016.

SHI, W.; DUSTDAR, S. The promise of edge computing. *Computer*, IEEE, v. 49, n. 5, p. 78–81, 2016.

SIEWIOREK, D.; SWARZ, R. *Reliable computer systems: design and evaluatuion*. [S.l.]: Digital Press, 2017.

SILVA, B.; MACIEL, P.; TAVARES, E.; ZIMMERMANN, A. Dependability models for designing disaster tolerant cloud computing systems. In: IEEE. *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*. [S.l.], 2013. p. 1–6.

SITTIG, D. F.; GONZALEZ, D.; SINGH, H. Contingency planning for electronic health record-based care continuity: a survey of recommended practices. *International journal of medical informatics*, Elsevier, v. 83, n. 11, p. 797–804, 2014.

SOUZA, V. B.; MASIP-BRUIN, X.; MARIN-TORDERA, E.; RAMÍREZ, W.; SANCHEZ, S. Towards distributed service allocation in fog-to-cloud (f2c) scenarios. In: IEEE. *Global Communications Conference (GLOBECOM), 2016 IEEE*. [S.l.], 2016. p. 1–6.

STOJIC, I. Algorithms for stationary analysis of stochastic petri nets. Università Ca'Foscari Venezia, 2017.

TANG, D.; KUMAR, D.; DUVUR, S.; TORBJORNSEN, O. Availability measurement and modeling for an application server. In: IEEE. *Dependable Systems and Networks, 2004 International Conference on*. [S.l.], 2004. p. 669–678.

TOOSI, A. N.; CALHEIROS, R. N.; BUYYA, R. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, ACM, v. 47, n. 1, p. 7, 2014.

TRIVEDI, K. S. *Probability & statistics with reliability, queuing and computer science applications*. [S.l.]: John Wiley & Sons, 2008.

TRIVEDI, K. S.; SATHAYE, A.; RAMANI, S. Availability modeling in practice. *Google Scholar*, 2005.

TSAI, P.-H.; HONG, H.-J.; CHENG, A.-C.; HSU, C.-H. Distributed analytics in fog computing platforms using tensorflow and kubernetes. In: IEEE. *Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific*. [S.l.], 2017. p. 145–150.

VAQUERO, L. M.; RODERO-MERINO, L. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, ACM, v. 44, n. 5, p. 27–32, 2014.

VERMA, A. K.; SRIVIDYA, A.; KARANKI, D. R. *Reliability and safety engineering*. [S.l.]: Springer, 2010.

VILAPLANA, J.; SOLSONA, F.; TEIXIDÓ, I.; MATEO, J.; ABELLA, F.; RIUS, J. A queuing theory model for cloud computing. *The Journal of Supercomputing*, Springer, v. 69, n. 1, p. 492–507, 2014.

VU-BAC, N.; LAHMER, T.; ZHUANG, X.; NGUYEN-THOI, T.; RABCZUK, T. A software framework for probabilistic sensitivity analysis for computationally expensive models. *Advances in Engineering Software*, Elsevier, v. 100, p. 19–31, 2016.

WANG, Y.; COIERA, E.; GALLEGO, B.; CONCHA, O. P.; ONG, M.-S.; TSAFNAT, G.; ROFFE, D.; JONES, G.; MAGRABI, F. Measuring the effects of computer downtime on hospital pathology processes. *Journal of biomedical informatics*, Elsevier, v. 59, p. 308–315, 2016.

WORTMANN, F.; FLÜCHTER, K. Internet of things. *Business & Information Systems Engineering*, Springer, v. 57, n. 3, p. 221–224, 2015.

XIE, N.; DUAN, M.; CHINNAM, R. B.; LI, A.; XUE, W. An energy modeling and evaluation approach for machine tools using generalized stochastic petri nets. *Journal of Cleaner Production*, Elsevier, v. 113, p. 523–531, 2016.

YANG, B.; TAN, F.; DAI, Y.-S.; GUO, S. Performance evaluation of cloud service considering fault recovery. In: SPRINGER. *IEEE International Conference on Cloud Computing*. [S.l.], 2009. p. 571–576.

YANG, R.; LI, B.; CHENG, C. A petri net-based approach to service composition and monitoring in the iot. In: IEEE. *Services Computing Conference (APSCC), 2014 Asia-Pacific*. [S.l.], 2014. p. 16–22.

YI, S.; HAO, Z.; QIN, Z.; LI, Q. Fog computing: Platform and applications. In: IEEE. *Hot Topics in Web Systems and Technologies (HotWeb), 2015 Third IEEE Workshop on*. [S.l.], 2015. p. 73–78.

YI, S.; LI, C.; LI, Q. A survey of fog computing: concepts, applications and issues. In: ACM. *Proceedings of the 2015 Workshop on Mobile Big Data*. [S.l.], 2015. p. 37–42.

YI, S.; QIN, Z.; LI, Q. Security and privacy issues of fog computing: A survey. In: SPRINGER. *International Conference on Wireless Algorithms, Systems, and Applications*. [S.l.], 2015. p. 685–695.

YOU, C.; HUANG, K.; CHAE, H.; KIM, B.-H. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, IEEE, v. 16, n. 3, p. 1397–1411, 2017.

YUE, Y.; HE, B.; TIAN, L.; JIANG, H.; WANG, F.; FENG, D. Rotated logging storage architectures for data centers: Models and optimizations. *IEEE Transactions on Computers*, IEEE, v. 65, n. 1, p. 203–215, 2016.

ZANELLA, A.; BUI, N.; CASTELLANI, A.; VANGELISTA, L.; ZORZI, M. Internet of things for smart cities. *IEEE Internet of Things journal*, IEEE, v. 1, n. 1, p. 22–32, 2014.

ZENG, W.; KOUTNY, M.; WATSON, P. Opacity in internet of things with cloud computing (short paper). In: IEEE. *Service-Oriented Computing and Applications (SOCA), 2015 IEEE 8th International Conference on*. [S.l.], 2015. p. 201–207.

ZUBEREK, W. M. Timed petri nets definitions, properties, and applications. *Microelectronics Reliability*, Elsevier, v. 31, n. 4, p. 627–644, 1991.