

Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Avaliação de Performabilidade de uma Arquitetura de Sistema de Monitoramento de Pacientes em Hospitais Inteligentes com Fontes de Dados Interna e Externa

Francisco Marcos da Silva Rocha

Número de Ordem PPGCC: M001

Teresina-PI, Agosto de 2024

Francisco Marcos da Silva Rocha

Avaliação de Performabilidade de uma Arquitetura de Sistema de Monitoramento de Pacientes em Hospitais Inteligentes com Fontes de Dados Interna e Externa

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Redes de Computadores), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Prof. Dr. Francisco Airton Pereira da Silva

Teresina-PI

Agosto de 2024

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Sistema de Bibliotecas UFPI - SIBi/UFPI
Biblioteca Setorial do CCN

R672a Rocha, Francisco Marcos da Silva.
Avaliação de performabilidade de uma arquitetuta de sistema de monitoramento de pacientes em hospitais inteligentes com fontes de dados interna e externa / Francisco Marcos da Silva Rocha. -- 2024.
67 f.

Dissertação (Mestrado) - Universidade Federal do Piauí. Centro de Ciências da Natureza. Programa de Pós-Graduação em Ciências da Computação, Teresina, 2024.
"Orientador: Prof. Dr. Francisco Airton Pereira da Silva".

1. Dispositivos Inteligentes. 2. Hospitais inteligentes. 3. Internet das coisas (IoT). 4. Redes de Petri Estocrásticas. I. Silva, Francisco Airton Pereira da. II. Titulo.

CDD 621.3

Bibliotecária: Caryne Maria da Silva Gomes - CRB3/1461

Francisco Marcos da Silva Rocha

Avaliação de Performabilidade de uma Arquitetura de Sistema de Monitoramento de Pacientes em Hospitais Inteligentes com Fontes de Dados Interna e Externa

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Redes de Computadores), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Documento assinado digitalmente
gov.br
FRANCISCO AIRTON PEREIRA DA SILVA
Data: 18/09/2024 20:55:10-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Francisco Airton Pereira da Silva
Orientador

Documento assinado digitalmente
gov.br
GLAUBER DIAS GONCALVES
Data: 19/09/2024 09:03:17-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Glauber Dias Gonçalves
Membro da Banca

Documento assinado digitalmente
gov.br
BRUNO COSTA E SILVA NOGUEIRA
Data: 18/09/2024 13:54:42-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Bruno Costa e Silva Nogueira
Membro da Banca

Documento assinado digitalmente
gov.br
MARCOS FAGUNDES CAETANO
Data: 18/09/2024 11:13:08-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Marcos Fagundes Caetano
Membro da Banca

Teresina-PI
Agosto de 2024

Agradecimentos

Tenho dito algumas vezes que esse Mestrado tem me perseguido por muito tempo, e sempre refutei aceitá-lo por medo do desafio. No entanto, pela riqueza do processo e por tudo o que está envolvido, considero esse grau uma importante realização pessoal, que, se fosse necessário, faria novamente.

O desafio foi aceito, e Deus me capacitou mais uma vez, como tem me capacitado sempre. Agradeço primeiramente a Ele, por ser minha fonte de motivação e energia.

Agradeço à minha família, meus filhos Theo e Matteo, e em especial à minha esposa Naiane que por muitas vezes teve que suportar o processo árduo junto comigo.

Aos meus pais Antônio e Nazaré, expresso minha gratidão por sempre terem depositado em mim a dedicação ao esforço, ao mérito e ao estudo.

Agradeço também ao meu orientador pela paciência, pelos conselhos e pela ajuda durante esse período.

Por fim, agradeço a todos que sempre se sentiram felizes por minhas conquistas e que certamente estão felizes com mais essa. Espero, de alguma forma, contribuir e inspirar vocês.

Obrigado!

Resumo

A variedade de tipos de sensores para monitoramento de saúde ajuda na tomada de decisão por profissionais da área. Para determinados quadros clínicos é interessante o monitoramento da saúde do paciente pós alta. Dispositivos vestíveis de monitoramento de saúde (como *smartwatches*) podem ser utilizados para enviar dados para o hospital. No entanto, para o monitoramento de um grande número de pacientes (externos e internos), é necessária uma infraestrutura computacional resiliente e de alto desempenho. Tais características exigem equipamentos de alto custo monetário. Para ajudar a planejar tal infraestrutura, esta dissertação apresenta um modelo SPN (*Stochastic Petri Net*) para avaliação de desempenho e performabilidade de uma arquitetura de sistema hospitalar multicamadas (*edge-fog-cloud*). O modelo permite avaliar o tempo médio de resposta (MRT), nível de utilização de recursos (U) e probabilidade de perda de dados (DP). Uma característica bem específica é considerar duas fontes de dados (interna e externa). As métricas de desempenho são avaliadas em dois cenários, baseados na variação do número de contêineres para processamento simultâneo na *cloud* e na *fog*. No cenário de variação na *cloud* (cenário 01) observou-se um congestionamento das requisições quando a utilização nessa camada chegou aos 100%, o que afetou negativamente o desempenho do sistema. Neste cenário, a configuração de 25 contêineres na nuvem se destacou por entregar dados mais equilibrados sobre o MRT, dando ao administrador do sistema maior controle sobre a alocação de recursos quando o hospital pode ter uma grande demanda por pacientes. Já no cenário de variação da capacidade da *fog* (cenário 02), a configuração de 25 entregou um menor tempo de resposta e taxa de queda em comparação ao cenário 01, bem como uma menor taxa de utilização dos componentes. Variando a capacidade da *fog*, as métricas apresentaram um melhor desempenho com utilização dos componentes e tempos de resposta menores do que o cenário anterior. O modelo é utilizado também para avaliar a performabilidade da métrica MRT sobre a variação dos parâmetros de maior influência na disponibilidade (MTTF do Contêiner) e desempenho (capacidade de processamento da *cloud*).

Palavras-chaves: Internet das Coisas (IoT), avaliação de desempenho, performabilidade, e-health, hospital inteligente, Redes de Petri Estocásticas.

Abstract

The variety of health monitoring sensor types aids healthcare professionals in decision-making. For certain clinical conditions, monitoring a patient's health post-discharge is of interest. Wearable health monitoring devices (such as smartwatches) can be used to transmit data to the hospital. However, for monitoring a large number of patients (both external and internal), a resilient and high-performance computational infrastructure is required. Such characteristics demand high-cost equipment. To assist in planning such infrastructure, this dissertation presents a Stochastic Petri Net (SPN) model for performance and performability evaluation of a multi-layered hospital system architecture (edge-fog-cloud). The model allows for the assessment of average response time (MRT), resource utilization level (U), and data loss probability (DP). A highly specific feature is considering two data sources (internal and external). Performance metrics are evaluated in two scenarios based on the variation in the number of containers for simultaneous processing in the cloud and the fog. In the cloud variation scenario (scenario 01), a congestion of requests was observed when the utilization in this layer reached 100%, which negatively affected the system performance. In this scenario, the configuration of 25 containers in the cloud stood out for delivering more balanced data on the MRT, giving the system administrator greater control over the allocation of resources when the hospital may have a high demand for patients. In the fog capacity variation scenario (scenario 02), the 25 configuration delivered a lower response time and drop rate compared to scenario 01, as well as a lower component utilization rate. When the fog capacity varied, the metrics showed better performance with component utilization and shorter response times than in the previous scenario. The model is also used to evaluate the performability of the MRT metric on the variation of the parameters that most influence availability (Container MTTF) and performance (cloud processing capacity).

Keywords: Internet of Things (IoT), Performance Evaluation, Performability, e-health, smart hospital, Stochastic Petri Net.

Lista de ilustrações

Figura 1 – Principais componentes em uma arquitetura de contêiner. 7
Figura 2 – Exemplo de modelo SPN. 9
Figura 3 – Exemplo de implementação SPN com limitação de processamento simultâneo. 9
Figura 4 – Exemplo de SPN com fila e processamento. 9
Figura 5 – Exemplo de gráfico de Pareto com efeitos padronizados. 11
Figura 6 – Gráficos de interações sinérgica e antagônica. 11
Figura 7 – Metodologia utilizada na Dissertação 24
Figura 8 – Arquitetura de um sistema de *smart hospital* com duas fontes de dados. 28
Figura 9 – Modelo base em SPN. 29
Figura 10 – Componentes de fontes de dados do modelo SPN. 30
Figura 11 – Roteador, antena e máquina de edge do modelo. 31
Figura 12 – Máquinas de fog e cloud do modelo. 32
Figura 13 – Modelagem de um contêiner utilizada para obtenção de MTTR_CO e MTTF_CO. 33
Figura 14 – Análise de desempenho do cenário 1. 40
Figura 15 – Análise de desempenho do cenário 2. 42
Figura 16 – Fatores, interações e seus respectivos efeitos sobre o MRT. 47
Figura 17 – Gráficos de interação entre os fatores. 49
Figura 18 – Análise de performabilidade do MRT em relação a variação do MTTF_CO e C_C. 50
Figura 19 – Variação do ângulo de visão do gráfico da Figura 18. 50
Figura 20 – Diagrama de validação. 53
Figura 21 – Modelo SPN da validação. 53
Figura 22 – Diagrama de Classes do Pasid Validator. 54
Figura 23 – Comunicação socket no cenário de 1 Service para cada Load Balancer. 55
Figura 24 – Diagrama de sequência do experimento. 56
Figura 25 – MRTs obtidos do experimento e modelo em relação à variação de quantidade de Services. 57

Lista de tabelas

Tabela 1 – Equações para cálculo de intervalo de confiança 15
Tabela 2 – Equações para cálculo do tamanho da amostra 17
Tabela 3 – Trabalhos relacionados. 22
Tabela 4 – Descrição das principais transições do modelo base. 33
Tabela 5 – Equações utilizadas nas métricas. 35
Tabela 6 – Configurações dos cenários analisados. 37
Tabela 7 – Definição da quantidade de *tokens*. 37
Tabela 8 – MTTR e MTTF dos componentes e análise de sensibilidade. 44
Tabela 9 – Tabela de fatores e níveis do DoE. 45
Tabela 10 – Tabela de combinações entre os fatores. 46
Tabela 11 – Tabela dos parâmetros utilizados na análise de performabilidade. . . . 48
Tabela 12 – Parâmetros de Transições Temporárias e Capacidades de Fila e Processamento utilizados no modelo do experimento 54
Tabela 13 – Dados Estatísticos do T-Test 57

Lista de abreviaturas e siglas

DoE	*Design os Experiments*
DP	*Drop Probability*
IoT	*Internet of Things*
IoHT	*Internet of Health Things*
JVM	*Java Virtual Machine*
LPT	*Last Processing Time*
mHealth	*Mobile Health*
MRT	*Mean Response Time*
MTTF	*Mean Time To Failure*
MTTR	*Mean Time To Repair*
NTP	*Network Time Protocol*
SPN	*Stochastic Petri Net*
VANET	*Vehicular Ad Hoc Network*

Sumário

1 INTRODUÇÃO . . . 3
1.1 Objetivos . . . 5
1.2 Estrutura do Trabalho . . . 5

2 REFERENCIAL TEÓRICO . . . 7
2.1 Contêineres de Processamento . . . 7
2.2 Redes de Petri Estocásticas . . . 8
2.3 Análise de Sensibilidade e DoE . . . 10
2.4 Performabilidade . . . 12
2.5 Técnicas de Validação de Modelos . . . 13
2.5.1 Medições de Sistemas Reais . . . 13
2.5.2 Resultados Teóricos . . . 14
2.5.3 Técnicas Estatísticas para Validação . . . 14
2.5.4 Intervalo de Confiança para a Média . . . 15
2.5.5 Comparação de Duas Alternativas . . . 16
2.5.6 Determinação do Tamanho da Amostra . . . 17

3 TRABALHOS RELACIONADOS . . . 19
3.1 Trabalhos com o Contexto Específico em *Smart Hospital* . . . 19
3.2 Trabalhos com o Contexto Geral da *Healthcare* . . . 20
3.3 Discussão Sobre as Contribuições . . . 20

4 METODOLOGIA . . . 23

5 ARQUITETURA E MODELO SPN . . . 27
5.1 Arquitetura Computacional de um *Smart Hospital* . . . 27
5.2 Modelo SPN . . . 29
5.2.1 Métricas . . . 34

6 RESULTADOS E DISCUSSÃO . . . 37
6.1 Análise de Desempenho . . . 37
6.1.1 Cenário 1 - Variação da capacidade da Cloud . . . 38
6.1.2 Cenário 2 - Variação da capacidade da Fog . . . 40
6.2 Análise de Sensibilidade e Performabilidade . . . 43
6.2.1 Análise de Sensibilidade com Percentage Difference . . . 43
6.2.2 Análise de Sensibilidade com DoE . . . 44
6.2.3 Análise de Performabilidade da Métrica MRT . . . 48

7 VALIDAÇÃO 51
7.1 Processo de Validação 51
7.2 Modelo, Ambiente do Experimento e Ferramenta de Validação . . . 52
7.3 Resultados 56

8 CONCLUSÃO 59

9 PUBLICAÇÕES 61

REFERÊNCIAS 63

1 Introdução

A Internet das Coisas (IoT) é um dos grandes avanços da tecnologia moderna. Os dispositivos IoT são intercomunicáveis e podem extrair informações sobre diferentes ambientes. Os dispositivos IoT são usados, por exemplo, para monitorar pacientes em *smart hospitals* (USLU; OKAY; DURSUN, 2020), para monitorar casas inteligentes (HUANG et al., 2020) ou para detectar acidentes em redes VANETs (redes ad-hoc veiculares) (KAUR; MALHOTRA; KAUR, 2020). O grupo *Statista* [1] prevê que os dispositivos IoT instalados devem chegar a 21,5 bilhões de unidades em todo o mundo até 2025. A IoT desempenha um papel importante no cenário da saúde. Um exemplo foi como a IoT auxiliou no combate ao coronavírus em todo o mundo por meio do monitoramento de sinais vitais (NDIAYE et al., 2020; SINGH et al., 2020). A incorporação da IoT na área da saúde é denominada IoHT (*Internet of Healthcare Things*) (YASSEIN et al., 2019).

IoHT é um ecossistema de dispositivos médicos, como dispositivos inteligentes de monitoramento e assistência médica (por exemplo, marca-passo inteligente, medidor de glicose no sangue, etc.) que têm a capacidade de coletar, analisar e transmitir dados vitais dos pacientes para sistemas de saúde por meio de redes de comunicação. Um cenário típico de IoHT envolve dispositivos de saúde inteligentes que monitoram os sinais vitais de pacientes focais e transmitem esses dados diretamente para servidores remotos poderosos como a computação em nuvem (SILVA; FÉ; GONÇALVES, 2020). A computação em nuvem pode ser usada para armazenar e analisar dados de saúde, permitindo a tomada de decisão automatizada em relação às intervenções. No entanto, recursos de computação mais próximos são necessários para reduzir atrasos de processamento.

Fog e *edge computing* são camadas complementares que preenchem essa lacuna de processamento local. Infraestruturas de *Fog* deslocam parte da capacidade da nuvem para uma camada mais próxima ao conjunto IoT (BONOMI et al., 2012). Dispositivos de borda conectam sensores IoT às camadas de névoa e nuvem (CARVALHO et al., 2020). Essa arquitetura *cloud-fog-edge* funciona como uma via de mão dupla que aproxima e distribui o poder de processamento, fornecendo o suporte necessário para os dispositivos restritos de IoHT. Dessa forma, as tecnologias IoHT podem contribuir para melhorar o desempenho, reduzir custos e mitigar riscos (RAJASEKARAN et al., 2019). No entanto, embora a IoHT tenha sido adotada para monitorar pessoas em todos os lugares, há um lugar onde a tecnologia deve ser muito mais eficiente devido à sua criticidade inerente: o hospital.

Um *smart hospital* é um conceito que surgiu pela rápida digitalização nos setores

[1] (https://tinyurl.com/y83kkser)

de saúde com o uso de IoHT, análise de dados, disponibilidade de serviços personalizados, inteligência artificial e pela capacidade de processamento da associação *cloud-fog-edge* (USLU; OKAY; DURSUN, 2020). A rede de um *smart hospital* pode receber dados de várias fontes (salas de hospital), processar dados de forma descentralizada com recursos de computação digital superiores para tomar decisões mais rápidas e assertivas. No entanto, um importante requisito funcional em sistemas de hospitais inteligentes é o desempenho, ou seja, a rapidez na entrega das informações e a distribuição uniforme na utilização dos recursos de processamento. A avaliação de desempenho de sistemas hospitalares inteligentes é essencial para garantir seu funcionamento ideal. Avaliar o desempenho de IoHT suportado por recursos multicamadas com experimentos reais pode ser considerado caro e impraticável com pacientes reais. Deve-se também levar em consideração os muitos dispositivos e a forma de orquestrar e testar a capacidade de todo o sistema. Neste sentido, um modelo analítico permite avaliar o sistema todo em escala.

Existem muitos parâmetros envolvidos e modelos analíticos podem ser úteis neste contexto, fazendo previsões baseadas em probabilidades.

Em um relatório publicado pela *Grand View Research* [2] em 2020, estimou-se que o mercado global de dispositivos de Internet das Coisas (IoT) na área de saúde atingiria um valor de US$ 534,3 bilhões até 2027. Isso reflete a crescente adoção de dispositivos IoT para monitoramento de saúde, gerenciamento de doenças crônicas e melhoria da eficiência dos sistemas de saúde.

Tal evolução incentiva-nos a pesquisar sobre a possibilidade de expansão dos recursos da assistência hospitalar para que de fato se tenha um acompanhamento dos pacientes no pós alta, principalmente àqueles com maior risco à recaídas. Nesse contexto é necessário a integração direta dos diversos dispositivos que cada vez mais permitem monitorar dados de saúde de seus usuários, como por exemplo smartphones, smartwatches, sensores de pele, tags inteligentes, etc.

Implementação de novas arquiteturas requerem estudos adequados, visto que devemos nos ater a questões de viabilidade e mitigação de investimentos que possam ocorrer de forma desnecessária. A utilização de modelagem de sistemas além de prevenir custo, agiliza a análise e oferece aos projetistas maior praticidade na calibração e definição dos componentes mais adequados à sua realidade. Dito isso, **é possível encontrarmos uma proposta de sistema de *Smart Hospital* distribuído em multicadas que possua a capacidade de integrar-se diretamente com os pacientes no pós alta atentando-se para um desempenho com baixos tempos de resposta, utilização uniforme dos componentes e sem gargalos de processamento?**

Existem trabalhos relevantes que avaliam o desempenho de sistemas IoHT com uso

[2] https://www.grandviewresearch.com/

de redes de Petri estocásticas (SPN) (RODRIGUES et al., 2021a; ANDRADE; NOGUEIRA, 2020a). O trabalho de Nguyen et al. (2021a) apresenta em seu artigo mecanismos que possibilitam a atualização dos componentes, visto que com o tempo eles tendem a ficar mais propícios a falhas. Silva et al. (2021a) utilizou um modelo de filas para avaliar o desempenho através de métricas tradicionais como MRT, utilização, vazão e taxa de perdas. O funcionamento padrão de um sistema precede, a priore, de onde e como os dados são gerados. Na modelagem, podemos abstrair as diversas fontes de dados em conjuntos. No contexto em questão, dizemos que todos os sensores e dispositivos do hospital pertencem a um conjunto de fonte de dados interna, e os dispositivos utilizados pelos pacientes fora do hospital pertencem ao conjunto de fonte externa. Os trabalhos citados acima não possuem o aspecto de dois conjuntos de fontes de dados ao mesmo tempo o qual é característica do presente trabalho.

Esta dissertação apresenta um modelo de redes de Petri estocásticas (SPN), bem como uma avaliação de desempenho e performabilidade sobre os cenários e configurações propostas. O intuito é abstrair um sistema IoHT de um *smart hospital* que contenha processamento multicamadas em sequência, bem como esteja preparado para receber de forma direta requisições dos pacientes ao saírem da assistência hospitalar.

1.1 Objetivos

O objetivo geral desta dissertação é **auxiliar, através de uma análise de performabilidade, projetistas e administradores no planejamento de sistemas de hospitais inteligentes que baseiam-se em arquitetura de processamento multicamadas e que possam receber simultaneamente requisições de duas fontes de dados**.

Os objetivos específicos deste trabalho são:

- Fornecer um modelo de sistema baseado em IoHT adaptável à mudanças que tenha capacidade para configurar diversos parâmetros (incluindo tempos de processamento e *delay*, tempo de falha e recuperação de componentes e capacidades paralelas de processamento).
- Possibilitar a identificação dos componentes e configurações de processamento que mais se destacam no aspecto de performabilidade de um sistema de *Smart Hospital*;

1.2 Estrutura do Trabalho

A seguir, o trabalho está organizado da seguinte forma: O Capítulo 2 apresenta os principais conceitos necessários. O Capítulo 3 apresenta os trabalhos relacionados, bem

como uma tabela comparativa à este trabalho. O Capítulo 4 apresenta a metodologia desde a arquitetura do modelo até as análises dos dados. O Capítulo 5 apresenta a arquitetura definida e o modelo SPN, bem como as métricas de desempenho utilizadas. O Capítulo 6 apresenta os resultados de desempenho dos dois cenários definidos, as análises de sensibilidade e a análise de performabilidade. Por fim, o Capítulo 7 apresenta um processo de validação do Modelo SPN.

2 Referencial Teórico

Este capítulo apresenta conceitos essenciais para a compreensão da proposta do modelo, análise dos dados e trabalhos relacionados. Contêineres para processamento paralelo são abordados, bem como conceitos de modelagem SPN, análise DoE e análise de performabilidade.

2.1 Contêineres de Processamento

Contêineres implementam o isolamento de processos no nível do sistema operacional, evitando assim essa sobrecarga desnecessária de recursos de hardware. Esses contêineres são executados no mesmo *kernel* do sistema operacional compartilhado da máquina *host* subjacente e um ou mais processos podem ser executados em cada contêiner. Os contêineres foram criados como forma de simplificar o processo de instalação e de lançamento de serviços, oferecendo um provisionamento escalável. Desenvolvedores de microsserviços, infraestruturas para processamento de fluxos de dados, processamento na borda da Internet, entre outros sistemas complexos, adotam estrutura baseada em contêineres para realizar o seu provisionamento (ASSUNCAO; VEITH; BUYYA, 2018).

A virtualização baseada em contêiner oferece uma alternativa leve à virtualização por *hypervisors*, ao proporcionar isolamento no nível do sistema operacional em vez de virtualizar o hardware. Enquanto os *hypervisors* criam uma camada de abstração de hardware que impõe sobrecarga ao executar sistemas operacionais completos em máquinas virtuais, os contêineres compartilham o mesmo *kernel* do sistema operacional do *host*, evitando essa sobrecarga e permitindo a execução de processos isolados de maneira mais eficiente. A arquitetura de um contêiner é apresentada na Figura 1.

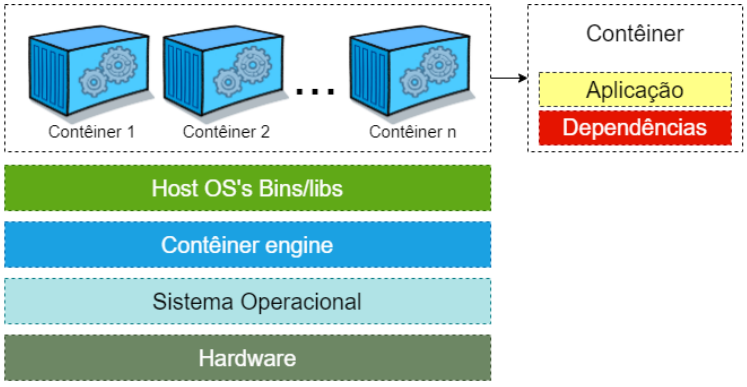


Figura 1 – Principais componentes em uma arquitetura de contêiner.

Um contêiner pode ser configurado com diversas aplicações afim de atender as demandas necessárias. Nessa forma de processamento paralelo, as aplicações de cada

contêiner instanciado compartilham do mesmo kernel de sistema operacional trazendo alguns benefícios, como por exemplo a não necessidade de utilização de *hypervisor* de virtualização, o que diminui a perda de desempenho no geral (NASCIMENTO, 2019). Além disso, a conteinerização evita duplicação de código, entre outros. Hoje em dia, muitos provedores estão adotando contêineres e expondo serviços de contêiner, como *Google Container Engine*, *Docker Datacenter* e *Amazon Elastic Container Service* (ECS) (HELALI; OMRI, 2021).

Docker é a implementação de contêiner mais usada, que é uma ferramenta de gerenciamento de código aberto para contêineres que automatiza a implantação de aplicativos (MERKEL et al., 2014). Outro novo concorrente emergente de tecnologia de contêiner do *Docker*, chamado *Rocket*, foi projetado para ser uma solução de contêiner mais segura, interoperável e aberta (AL-DHURAIBI et al., 2017).

Devido à falta da camada de hardware virtual, os contêineres podem ser criados (MAVRIDIS; KARATZA, 2017) e migrados mais rapidamente (MONDESIRE et al., 2019), e iniciar mais rapidamente do que as VMs (WATADA et al., 2019), que levam muito tempo para serem criadas (CHAE; LEE; LEE, 2019). Os contêineres não exigem a reinicialização de nenhum sistema operacional para se reiniciarem (CHAE; LEE; LEE, 2019). Este método de virtualização permite uma implantação densa com um número maior de contêineres por máquina do que VMs, uma vez que os contêineres são mais leves (MAVRIDIS; KARATZA, 2017). Os contêineres executados nativamente são mais eficientes em termos de recursos do sistema [(PAHL, 2015) e requerem menos hardware pelo fato de reduzirem o número de camadas a serem gerenciadas (MAVRIDIS; KARATZA, 2017). Exige-se também menos espaço para manter múltiplas cópias de um contêiner e, então, fornecem uso otimizado de armazenamento com espaço mínimo (WATADA et al., 2019).

Nesta dissertação, estabelece-se que cada requisição seja processada por uma instância de contêiner *Docker*, contendo todas as aplicações necessárias. Essas instâncias operam de forma paralela, distribuídas entre as camadas de *fog* e *cloud* do sistema

2.2 Redes de Petri Estocásticas

As Redes de Petri são um formalismo matemático muito usado na modelagem e simulação de sistemas, os quais podem ser concorrentes, assíncronos, paralelos e distribuídos. Uma *Stochastic Petri Net* (SPN) é representada graficamente por arcos (setas) que podem ter pesos, transições temporizadas, transições imediatas, lugares e tokens (LIMA; CALLOU; ANDRADE, 2021). A Figura 2 ilustra um exemplo de uma rede SPN de comportamento circular que é composta pelos lugares P1 e P2, transição imediata T1 e transição temporizada T2. No lugar P1 existe um token associado. Transação imediata não demanda tempo e ocorre de modo prioritário em relação à transição temporizada, ou

seja, com o tempo equivalente a zero. Já uma transição temporizada é caracterizada por um delay associado. Os tempos associados às transições temporizadas são distribuídos exponencialmente, o que é a principal característica de uma SPN (MURATA, 1989). Há também funções de guarda em SPNs. As funções de guarda são expressões booleanas que controlam o disparo de uma transição, declarando alguma condição em relação à marcação da rede. Se a função de guarda de uma transição produz um valor verdadeiro, é capaz de disparar, caso contrário, a transição é desabilitada (MARSAN et al., 1998).

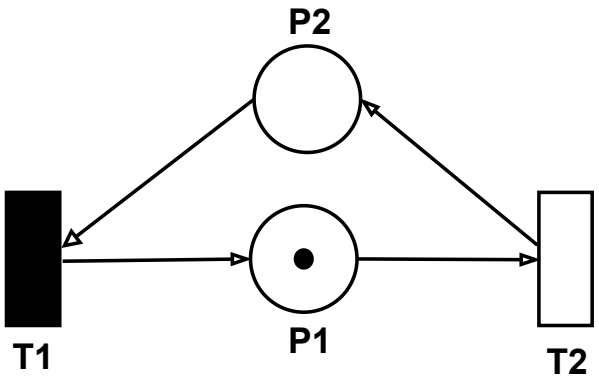


Figura 2 – Exemplo de modelo SPN.

No modelo da Figura 2 o deslocamento do token que inicialmente encontra-se em P1 é acionado quando a transição T2 cumpre seu tempo. Logo após, o token passa à P2 que imediatamente retorna a P1 pois T1 é uma transição imediata.

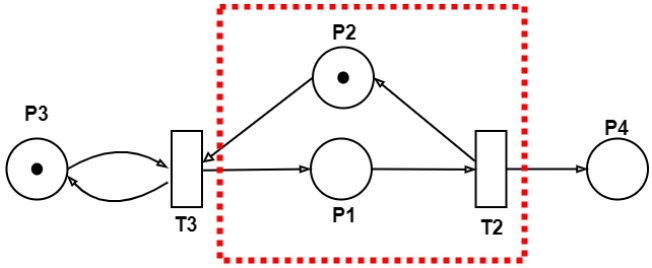


Figura 3 – Exemplo de implementação SPN com limitação de processamento simultâneo.

A Figura 3 apresenta um modelo com implementação capacidade de processamento. Nesse exemplo, tokens infinitos são gerados a cada tempo T3 e enviados à P1. O objetivo é que os tokens cheguem à P4 e para isso é necessário que eles passem pelo temporizador T2. Por propriedade, um token só pode sair de P3 até P1 somente se houver ao menos um token em P2, e quando o token consegue ser enviado de P1 à P4 o número de tokens em P2 (capacidade dinâmica de processamento) é incrementada. Sendo assim, utilizamos esse "triângulo" destacado em vermelho para representar uma máquina genérica que possui capacidade de processamento simultâneo.

Essa modelagem também é utilizada para definir a capacidade de fila que recebe os tokens antes de serem enviadas ao processamento, nesse caso há dois "triângulos"em

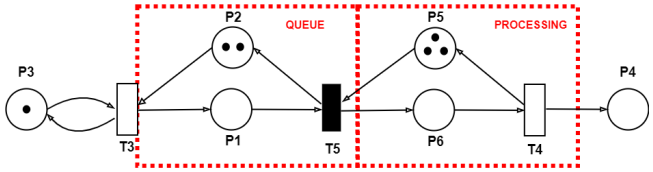


Figura 4 – Exemplo de SPN com fila e processamento.

sequência: um para definir a fila e outro para definir o processamento, conforme Figura 4. Nesse exemplo, o número de tokens em P2 e P5 representam a capacidade de fila e processamento, respectivamente. Em tempo de execução, à medida que o intervalo de requisições em T3 diminui ou aumenta, o número de tokens no modelo tende a variar e é com base nisso que podemos obter algumas métricas e analisar o desempenho do sistema. Neste trabalho, utilizamos a ferramenta gráfica Mercury (MACIEL et al., 2017) para criação do modelo SPN.

2.3 Análise de Sensibilidade e DoE

O *Design of Experiments* (DoE) corresponde a um conjunto de técnicas estatísticas que aprofundam o conhecimento sobre o produto ou processo em estudo (KLEIJNEN, 1995). Também pode ser definido por uma série de testes nos quais o pesquisador altera o conjunto de variáveis ou fatores de entrada para observar e identificar os motivos das mudanças na resposta de saída.

Os projetistas de sistemas costumam adotar análises de sensibilidade para avaliar o quão "sensível" uma métrica é a mudanças no modelo (SANTOS et al., 2021). Os parâmetros a serem alterados são definidos por meio de um plano de experimento. O objetivo é gerar a quantidade mais significativa de informações com o mínimo de experimentos possíveis. O comportamento do sistema com base nas mudanças de parâmetro pode ser observado usando conjuntos de saídas. Na literatura, os gráficos geralmente mais adotados para experimentos com DoE são Pareto e Interação.

O gráfico de Pareto permite detectar qual fator ou interação de fatores são mais importantes para o processo ou estudo de otimização de projeto com o qual se deve lidar. Nesse gráfico são exibidos os valores absolutos dos efeitos e uma linha vertical de referência. Qualquer efeito que ultrapasse essa linha de referência é potencialmente importante. Um gráfico de Pareto é construído como na Figura 5, por exemplo. O gráfico mostra que os fatores B (tool geometry) e C (cutting angle) e a interação AC possui maior impacto. O eixo x representa uma determinada métrica quantitativa e o eixo y representa os fatores e suas combinações na análise.

Relações de interação entre os diversos parâmetros do modelo e seus efeitos no desempenho podem serem identificadas, logo, as interações entre os efeitos são cruciais.

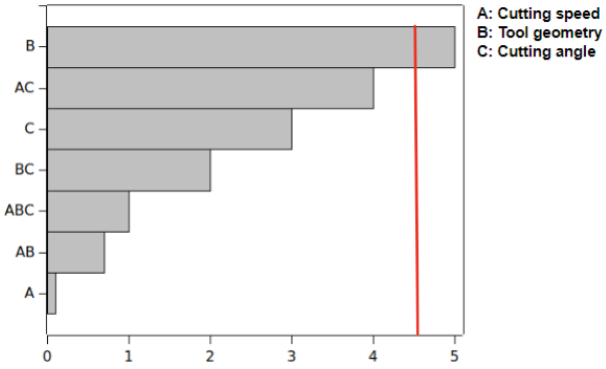


Figura 5 – Exemplo de gráfico de Pareto com efeitos padronizados.

As interações ocorrem quando o efeito de um fator depende do nível de outro fator. Uma medida de design sempre aborda vários fatores. Entender como esses fatores interagem permite que se possa escolher a melhor combinação de medidas, revelando combinações de fatores com efeito cumulativo ou degradante. A interação entre os fatores A e B pode ser calculada usando a Equação 2.1.

$$I_{A,B} = \frac{1}{2}(E_{A,B(+1)} - E_{A,B(-1)}) \tag{2.1}$$

O $E_{A,B(+1)}$ é o efeito do fator A no nível alto do fator B e $E_{A,B(-1)}$ é o efeito do fator A no nível baixo do fator B.

Para determinar se dois parâmetros de processo estão interagindo ou não, pode-se usar uma ferramenta gráfica simples, porém poderosa, chamada de gráficos de interação. O eixo x representa os dois valores da variação do fator A, as linhas representar os valores da variação do fator B e as extremidades dessas linhas representam os valores quantitativos da métrica avaliada em relação à interação dos dois fatores. Se as linhas no gráfico de interação forem paralelas, não haverá interação entre os parâmetros do processo. Isso implica que a mudança na resposta média do fator A não depende dos níveis do fator B. Por outro lado, se as linhas não são paralelas, existe uma interação entre os fatores. Quanto maior o grau de afastamento de ser paralelo, mais forte o efeito de interação. Para interação sinérgica, as linhas no gráfico não se cruzam. Por exemplo, a Figura 6(a) é um exemplo de interação sinérgica.

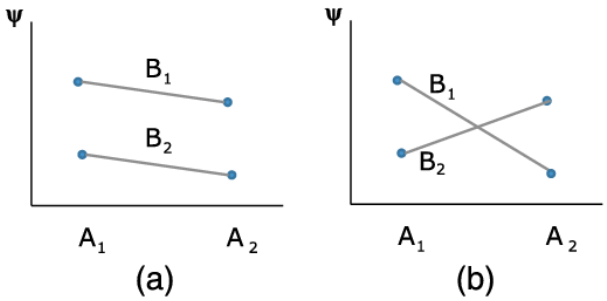


Figura 6 – Gráficos de interações sinérgica e antagônica.

Na interação antagônica, as linhas no gráfico se cruzam. Isso pode ser ilustrado na Figura 6(b). Neste caso, a mudança na resposta média para o fator A no nível A_1 é alta em comparação ao nível A_2. As mudanças nos níveis do fator A para a resposta média, indica uma dependência do fator A em relação aos níveis do fator B.

2.4 Performabilidade

Em 1980, John Meyer introduziu o termo performabilidade - do inglês *performability* - no contexto da avaliação de computadores de controle de aeronaves de alta confiabilidade de uso da *National Aeronautics and Space Administration* (NASA) (MEYER, 1980). Originalmente, o termo foi proposto para refletir confiabilidade e outros atributos de desempenho, como disponibilidade e manutenção. Entretanto, isso representava apenas parcialmente as métricas de desempenho. Em seguida, outros conceitos foram complementados, tais como, sustentabilidade, segurança e qualidade (MISRA et al., 2008).

A avaliação de performabilidade abrange a utilização de métodos de avaliação de desempenho e dependabilidade (HAVERKORT; NIEMEGEERS, 1996). A partir da avaliação é possível investigar o efeito de eventos de falhas e atividades de reparo na degradação do desempenho de sistemas. Usualmente avalia-se separadamente os atributos de desempenho e dependabilidade, no entanto, o desempenho é dependente da dependabilidade. Neste sentido, é notável que a ocorrência de falhas implicará na qualidade do serviço oferecido pelo sistema, sendo esse comprometimento uma degradação no desempenho (ARAÚJO, 2019).

De forma objetiva, a performabilidade é uma análise que abrange como as métricas de desempenho se comportam em conjunto com as métricas de dependabilidade dos componentes que integram o sistema. O conceito de dependabilidade em sistemas computacionais é definido em (AVIZIENIS et al., 2004) como a propriedade do sistema que integra os atributos como confiabilidade, disponibilidade, segurança, confidencialidade, integridade e manutenabilidade. Em outras palavras, a **dependabilidade** diz respeito a **capacidade de entregar um serviço confiável**, evitando defeitos inaceitáveis para os seus usuários. Esses atributos representam o nível de confiança atribuído a determinado sistema em relação a seu correto funcionamento (SOUZA et al., 2017):

- **Confiabilidade**: atender a especificação dentro das condições definidas e continuidade do serviço correto;
- **Disponibilidade**: prontidão para o serviço correto;
- **Segurança**: ausência de consequências catastróficas para o(s) utilizador(es) e para o ambiente;

- **Confidencialidade**: ausência de divulgação não autorizada de informações;
- **Integridade**: ausência de alterações impróprias no estado do sistema;
- **Manutenabilidade**: capacidade de submeter-se a reparações e modificações (SOUZA et al., 2017).

Para esta dissertação, o modelo proposto aborda o aspecto de disponibilidade dos componentes. Portanto, a performabilidade aplicada neste trabalho é uma junção de desempenho e disponbilidade.

2.5 Técnicas de Validação de Modelos

A validação de modelos é essencial para garantir que as suposições usadas no desenvolvimento sejam razoáveis e que o modelo produza resultados próximos aos observados em sistemas reais. As técnicas de validação variam de acordo com as suposições e os sistemas modelados. Ao contrário das técnicas de verificação, que são geralmente aplicáveis, as técnicas de validação podem não se aplicar a outras simulações (SANTNER et al., 2003; JAIN, 1991).

A validação do modelo envolve a verificação de três aspectos principais: premissas, valores e distribuições dos parâmetros de entrada, bem como os valores e conclusões de saída. Esses aspectos podem ser testados comparando-os com a intuição de especialistas, medições de sistemas reais e resultados teóricos (JAIN, 1991).

2.5.1 Medições de Sistemas Reais

A comparação com sistemas reais é a forma mais confiável e preferida para validar um modelo de simulação. Na prática, no entanto, isso muitas vezes não é viável devido à inexistência de sistemas reais ou ao alto custo das medições necessárias. Mesmo assim, uma ou duas medições podem aumentar significativamente a validade de uma simulação.

É fundamental que as suposições, valores de entrada e saída, cargas de trabalho, configurações e comportamento do sistema sejam comparados com aqueles observados no mundo real. Técnicas estatísticas, como intervalo de confiança para a média, comparação de duas alternativas e determinação do tamanho da amostra, podem ser empregadas para verificar se dois sistemas são idênticos, possibilitando a comparação entre o modelo de simulação e os dados reais coletados. Por exemplo, as distribuições de entrada podem ser determinadas por técnicas de ajuste de distribuição estatística ou usando testes estatísticos, como o teste qui-quadrado (chi-square test) (JAIN, 1991).

Em simulações baseadas em rastreamento, é essencial utilizar múltiplos rastreamentos obtidos em diferentes ambientes para validar o modelo. Esse processo garante que

o modelo não apenas reflita condições específicas, mas também seja robusto o suficiente para generalizar a partir de diferentes cenários. Utilizar essas técnicas avançadas permite uma validação mais rigorosa e confiável, proporcionando maior confiança nos resultados da simulação e na sua aplicabilidade em situações reais (JAIN, 1991; SANTNER et al., 2003).

2.5.2 Resultados Teóricos

Em alguns casos, é possível modelar analiticamente o sistema sob suposições simplificadoras. Também pode ser possível determinar analiticamente quais deveriam ser as distribuições de entrada. Nesses casos, a similaridade dos resultados teóricos e de simulação é usada para validar o modelo de simulação. Da mesma forma, ao desenvolver modelos de sistemas de filas de sistemas de computador, os analistas muitas vezes descobrem que o comportamento do sistema real é muito difícil de modelar exatamente (JAIN, 1991). Nesses casos, eles desenvolvem uma técnica aproximada e a validam comparando os resultados com aqueles obtidos de uma simulação.

A validação por comparação de resultados teóricos e de simulação deve ser usada com cuidado, pois ambos podem ser inválidos no sentido de que ambos podem não representar o comportamento de um sistema real. No entanto, é uma técnica útil, desde que alguns pontos (de teoria ou simulação) tenham sido validados usando a intuição de especialistas ou medições de sistemas reais (JAIN, 1991). A modelagem analítica permite então a validação para casos mais complexos. Por exemplo, um modelo de simulação de multiprocessadores pode ser validado usando alguns sistemas reais de um ou dois processadores e, em seguida, a teoria pode ser usada para validar para um número maior de processadores (SANTNER et al., 2003).

A validação de modelos é essencial para garantir a confiabilidade e precisão das simulações. Ela envolve verificar se as suposições do modelo são realistas e se os resultados se aproximam dos observados em sistemas reais. Existem três técnicas principais de validação: intuição de especialistas, medições em sistemas reais e comparação com resultados teóricos. A escolha da técnica depende das características do sistema modelado e dos recursos disponíveis para validação(JAIN, 1991).

2.5.3 Técnicas Estatísticas para Validação

As técnicas estatísticas cruciais para a análise de dados de simulação e validação de modelos será apresentada nesta Subseção. Inicialmente, exploramos os intervalos de confiança para a média, que estimam a média populacional com base em amostras e fornecem uma faixa de valores esperada com certo nível de confiança. Essa ferramenta é essencial para quantificar a precisão das estimativas (JAIN, 1991).

Além disso, detalhamos métodos de comparação entre duas alternativas, conside-

rando observações pareadas ou não pareadas. Essas técnicas permitem avaliar diferenças significativas entre sistemas. Também discutimos a importância do tamanho adequado da amostra para estimativas confiáveis (JAIN, 1991).

2.5.4 Intervalo de Confiança para a Média

A técnica do intervalo de confiança para a média é uma das ferramentas mais fundamentais e amplamente utilizadas em estatística. Essa técnica nos permite estimar a média de uma população com base em uma amostra e fornecer uma faixa de valores dentro da qual esperamos que a média verdadeira da população se situe com um determinado nível de confiança, geralmente 95% ou 99%. Para calcular um intervalo de confiança, precisamos primeiro determinar a média amostral ($\bar{x}$), que é a soma dos valores observados na amostra dividida pelo número de observações (n). A média amostral serve como um ponto central para o intervalo de confiança. Em seguida, calculamos o erro padrão da média (SE), que é o desvio padrão da amostra (σ) dividido pela raiz quadrada do número de observações ($\sqrt{n}$). O erro padrão fornece uma medida da variabilidade ou dispersão dos valores da amostra em torno da média (JAIN, 1991).

Equação	**Descrição**
$\bar{x} = \frac{\sum x_i}{n}$	Média amostral
$SE = \frac{\sigma}{\sqrt{n}}$	Erro padrão da média
$\bar{x} \pm z \times SE$	Intervalo de confiança

Tabela 1 – Equações para cálculo de intervalo de confiança

Uma vez que temos a média amostral e o erro padrão, podemos calcular o intervalo de confiança usando a fórmula $(\bar{x}) \pm z \times SE$, onde z é o valor crítico da distribuição normal para o nível de confiança desejado. A Tabela 1 apresenta as devidas equações. Por exemplo, para um nível de confiança de 95%, z é aproximadamente 1.96. Isso significa que esperamos que 95% dos intervalos de confiança calculados a partir de amostras aleatórias dessa população contenham a média verdadeira. O intervalo de confiança é, portanto, uma faixa de valores simétrica em torno da média amostral, que se expande ou contrai dependendo da variabilidade dos dados (refletida pelo erro padrão) e do tamanho da amostra. Esta técnica é extremamente útil porque, além de fornecer uma estimativa pontual (a média amostral), também nos dá uma medida de incerteza associada a essa estimativa (JAIN, 1991).

Por exemplo, considere que estamos analisando o tempo de resposta de um sistema de TI e coletamos uma amostra de 30 tempos de resposta, encontrando uma média amostral de 20 segundos e um desvio padrão de 5 segundos. O erro padrão seria calculado como $\frac{5}{\sqrt{30}} \approx 0.91$. Para um intervalo de confiança de 95%, multiplicamos o erro padrão por 1.96, resultando em aproximadamente 1.79. Portanto, o intervalo de confiança para a média de

tempo de resposta seria de 20 ± 1.79, ou seja, de 18.21 a 21.79 segundos. Este intervalo sugere que, com 95% de confiança, a média verdadeira do tempo de resposta do sistema está entre 18.21 e 21.79 segundos. Utilizar intervalos de confiança é crucial em análises de desempenho e experimentos científicos porque permite aos analistas e pesquisadores fazer inferências sobre a população com base em amostras, considerando a variabilidade e proporcionando uma visão mais robusta do que simplesmente reportar uma média pontual.

2.5.5 Comparação de Duas Alternativas

Comparar duas alternativas é uma tarefa comum e crítica na análise de desempenho e na tomada de decisões. As técnicas estatísticas para esta comparação podem ser aplicadas a observações pareadas ou não pareadas, dependendo da natureza dos dados. Observações pareadas são aquelas em que cada par de observações consiste em um valor de cada sistema sob a mesma condição ou tempo, permitindo um controle mais rigoroso das variáveis externas. A análise das diferenças entre cada par de observações fornece insights sobre se uma das alternativas tem um desempenho consistentemente melhor. A diferença entre cada par é calculada, e a média dessas diferenças, juntamente com seu intervalo de confiança, é utilizada para determinar se há uma diferença significativa entre os sistemas. Por exemplo, se tivermos os tempos de resposta de dois sistemas A e B sob seis workloads diferentes e calcularmos as diferenças para cada workload, poderemos construir um intervalo de confiança para a média dessas diferenças. Se o intervalo de confiança não incluir zero, isso indica que há uma diferença significativa entre os sistemas (JAIN, 1991).

Para observações não pareadas, onde as observações dos dois sistemas são independentes, a abordagem é um pouco diferente. Neste caso, calculamos a diferença entre as médias dos dois grupos e construímos um intervalo de confiança para essa diferença. Isso envolve considerar as variâncias de ambos os grupos, pois a dispersão dos dados pode influenciar a interpretação dos resultados. A fórmula para o intervalo de confiança da diferença entre duas médias independentes é mais complexa e envolve o cálculo do erro padrão combinado das duas amostras. Por exemplo, suponha que tenhamos dois sistemas de TI, A e B, e que coletamos amostras independentes de tempos de resposta para cada sistema. Se a média e o desvio padrão dos tempos de resposta para o sistema A forem 20 segundos e 5 segundos, respectivamente, com 30 observações, e para o sistema B forem 22 segundos e 4 segundos, com 35 observações, podemos calcular o erro padrão combinado e o intervalo de confiança para a diferença entre as duas médias (JAIN, 1991).

A comparação de duas alternativas é essencial para validar melhorias ou mudanças em sistemas. Um exemplo prático disso seria avaliar duas versões de um algoritmo de compressão de dados. Suponha que as diferenças nos tempos de compressão entre a versão antiga e a nova para um conjunto de arquivos sejam medidas. Se a média das diferenças de tempo for significativamente diferente de zero, com um intervalo de confiança que não

inclua zero, podemos concluir que uma versão do algoritmo tem um desempenho melhor que a outra. Essa análise pode ser complementada por gráficos de boxplot ou histogramas das diferenças, que visualmente reforçam a interpretação dos dados estatísticos. Assim, comparar duas alternativas com técnicas estatísticas robustas nos permite tomar decisões informadas sobre qual sistema ou método adotar, baseando-se em evidências quantitativas e não apenas em intuições ou suposições (JAIN, 1991).

2.5.6 Determinação do Tamanho da Amostra

A determinação do tamanho adequado da amostra é um aspecto crítico na condução de estudos estatísticos e experimentos, garantindo que os resultados sejam confiáveis e precisos. O tamanho da amostra necessário para obter uma estimativa precisa depende da variabilidade dos dados, do nível de confiança desejado e da precisão requerida. Para estimar uma média, o tamanho da amostra pode ser calculado para garantir que o intervalo de confiança tenha uma largura desejada. A fórmula para calcular o tamanho da amostra para a média $n = \left(\frac{z\sigma}{r}\right)^2$ é, onde z é o valor crítico da distribuição normal para o nível de confiança desejado, é o desvio padrão da amostra e r é a precisão desejada (ou a metade da largura do intervalo de confiança). Esta fórmula permite determinar quantas observações são necessárias para estimar a média da população com a precisão desejada (JAIN, 1991).

Além de médias, a determinação do tamanho da amostra é crucial ao trabalhar com proporções. A fórmula para calcular o tamanho da amostra para proporções é $\frac{z^2p(1-p))}{r^2}$, onde p é a proporção estimada da população e r é a precisão desejada. A Tabela 2 apresenta as equações utilizadas. Por exemplo, se estamos interessados em estimar a proporção de páginas ilegíveis de uma impressora e estimamos que essa proporção seja 0.001, desejando uma precisão de 0.000001 a 95% de confiança, precisamos calcular o tamanho da amostra necessário para garantir essa precisão. Utilizando a fórmula apropriada, podemos determinar o número de páginas que devem ser examinadas para obter uma estimativa confiável da proporção de páginas ilegíveis.

Equação	**Descrição**
$n = \left(\frac{z\sigma}{r}\right)^2$	Tamanho da amostra para estimar a média
$n = \frac{z^2p(1-p)}{r^2}$	Tamanho da amostra para estimar proporções

Tabela 2 – Equações para cálculo do tamanho da amostra

A determinação do tamanho da amostra também é essencial ao comparar duas alternativas, como sistemas ou algoritmos. Neste caso, o tamanho da amostra deve ser suficiente para garantir que os intervalos de confiança das médias dos dois sistemas não se sobreponham, indicando uma diferença significativa. Os cálculos para o tamanho da amostra neste contexto envolvem as variâncias e médias das duas amostras. Fórmulas mais complexas são usadas, como aquelas que consideram o erro padrão combinado das duas

amostras. Garantir um tamanho de amostra adequado é fundamental para evitar conclusões incorretas devido a amostras insuficientes, o que poderia levar a uma interpretação errônea dos resultados e a decisões inadequadas (JAIN, 1991).

Para ilustrar, suponha que estamos comparando a eficiência energética de dois tipos de motores. Se queremos detectar uma diferença média de 5 unidades de eficiência com 95% de confiança e sabemos que o desvio padrão da eficiência é 10, podemos usar a fórmula para determinar o tamanho necessário da amostra. Considerando a precisão desejada e o nível de confiança, o cálculo nos forneceria o número de medições necessárias para fazer uma comparação estatisticamente significativa entre os dois motores. Este processo de determinação do tamanho da amostra é crucial em muitos campos, desde a engenharia até a medicina, onde a coleta de dados pode ser custosa e demorada. Portanto, a técnica não apenas assegura a precisão e a validade dos resultados, mas também otimiza os recursos utilizados na condução dos estudos (JAIN, 1991).

3 Trabalhos Relacionados

Este capítulo apresenta trabalhos que possuem contexto ou abordagens pertinentes ao tema abordado neste estudo. A seleção dos trabalhos levou em consideração o contexto, o método de avaliação, se os trabalhos realizavam uma avaliação de disponibilidade ou uma avaliação de desempenho. Quando um trabalho realizava uma avaliação de desempenho, eram verificadas as métricas analisadas. Desse modo, para uma exploração mais aprofundada das informações, os artigos foram agrupados com base no contexto. A decisão de agrupar os trabalhos com base em seu contexto é fundamental ao analisar dados no contexto hospitalar ou de assistência médica. As fontes internas de dados, originadas dentro do ambiente hospitalar, e as fontes externas, relacionadas à indústria de saúde em geral, possuem informações divergentes que desempenham um papel crucial na tomada de decisões por profissionais da área. A Tabela 3 apresenta as principais contribuições dos trabalhos relacionados a esta dissertação.

3.1 Trabalhos com o Contexto Específico em *Smart Hospital*

A escolha de avaliar o desempenho e/ou a disponibilidade em *Smart Hospitals* é essencial para garantir a qualidade da assistência, eficiência operacional, segurança dos pacientes e a tomada de decisões embasadas. O trabalho de Chen et al. (2011) propõe investigar o desempenho de três políticas de agendamento quando há apenas alguns componentes de pacientes no sistema de um *Smart Hospital*. O estudo de Gomez-Sacristan et al. (2015) propõe avaliar a qualidade das comunicações em um *Smart Hospital* com serviços diversos conectados via Metro-Ethernet (Ethernet para redes metropolitanas). O trabalho de Rodrigues et al. (2021b) apresenta uma avaliação de desempenho e disponibilidade de um sistema hospitalar inteligente sem a necessidade de investir previamente em equipamentos reais.

O trabalho de Nguyen et al. (2021b) propõe um modelo SPN de performabilidade abrangente de um *Smart Hospital* baseado em borda/névoa para a quantificação de performabilidade de transações e serviços de dados médicos em hospitais locais ou centros médicos. O trabalho de Rodrigues et al. (2019) adota a abordagem de modelagem analítica para avaliar o desempenho de um sistema hospitalar inteligente, sendo possível calibrar a capacidade de recursos do servidor e o tempo de serviço. O trabalho de Rodrigues et al. (2020) propõe avaliar a disponibilidade de um sistema de *Smart Hospital* e apresenta uma análise de sensibilidade que identifica os componentes da arquitetura mais críticos.

3.2 Trabalhos com o Contexto Geral da *Healthcare*

A realização de uma avaliação do desempenho e/ou disponibilidade em serviços *Healthcare* é vital para assegurar a qualidade dos cuidados, otimizar recursos, garantir a continuidade dos serviços. O estudo de Silva et al. (2021b) propõe um modelo de rede de filas M/M/c/K para a avaliação de desempenho de uma infraestrutura de *Internet of Healthcare Things* (IoHT) em associação com um continuum de computação em nuvem/névoa/borda de três camadas. O trabalho de Khayal e Farid (2021) desenvolve um modelo de simulação matemática dinâmica para prestação de cuidados de saúde personalizados e resultados de saúde individuais gerenciados.

O trabalho de Santos et al. (2020) utiliza uma abordagem interdisciplinar que combina modelos estocásticos com algoritmos de otimização para analisar como as falhas afetam a disponibilidade do sistema de monitoramento de e-health. O estudo de Kafhali e Salah (2019) propõe um modelo analítico para um sistema *Medical Internet of Things* (ou MIoT) integrado com *fog* e *cloud computing*, analisando a redução do custo dos recursos computacionais e, ao mesmo tempo, garantindo restrições de desempenho. O trabalho de Santos et al. (2022a) utiliza uma abordagem para maximizar a disponibilidade de sistemas *e-health* usando modelos estocásticos e otimização assistida, considerando orçamento e comparando algoritmos. O estudo de Lisboa et al. (2018) propõe modelos estocásticos para analisar como as falhas afetam a disponibilidade em uma de monitoramento de *e-health* baseada em sensores e cenários de infraestrutura em *cloud* e *fog computing*.

O trabalho de Nguyen et al. (2021c) propõe quantificar as medidas de segurança de uma infraestrutura de IoMT baseada em uma arquitetura física integrada de paradigmas de computação em nuvem/névoa/borda (CFE). O trabalho de Valentim et al. (2023) apresenta uma abordagem de modelagem baseada em Redes de Petri Estocásticas Generalizadas (GSPN) para avaliar a disponibilidade de uma arquitetura de IoMT baseada em uma nuvem privada. O estudo de Strielkina et al. (2018) apresenta uma abordagem para desenvolver um conjunto de modelos de *Markov* para uma infraestrutura de IoT *healthcare* que permite levar em consideração questões de segurança e proteção. O trabalho de Tang e Xie (2021) um modelo de disponibilidade de um sistema IoT healthcare composto por dois grupos de estruturas descritas por modelos de espaço de estados de *Markov* separados. O trabalho de (SANTOS et al., 2018) propõe uma arquitetura de monitoramento de *e-health* baseada em sensores que depende de infraestruturas de nuvem e nevoeiro para lidar e armazenar dados de pacientes.

3.3 Discussão Sobre as Contribuições

Este trabalho propõe um modelo SPN para avaliação de desempenho e disponibilidade de uma arquitetura de sistema hospitalar multicamadas (*edge-fog-cloud*). Para

uma análise mais aprofundada das contribuições deste trabalho em relação à literatura apresentada nas seções anteriores, destacamos quatro pontos principais.

1. **Fonte de dados:** A principal contribuição deste trabalho está relacionada à utilização de duas fontes de dados. A primeira é uma fonte interna que representa os sensores e dispositivos IoT que fornecem dados dos pacientes dentro do contexto específico do *Smart Hospital*. A segunda fonte é externa, representando dispositivos que podem fornecer dados de saúde de pacientes fora do hospital após a alta hospitalar, de forma mais geral. A combinação do uso dessas duas fontes de dados, juntamente com a modelagem analítica, representa uma diferenciação em relação aos trabalhos da literatura.

2. **Método de avaliação:** A modelagem proposta neste trabalho permite avaliar o desempenho e a disponibilidade de um sistema hospitalar inteligente sem a necessidade de investimento antecipado em equipamentos reais. A avaliação do modelo foi realizada utilizando a ferramenta Mercury (MACIEL et al., 2017). Embora exista uma quantidade considerável de trabalhos na literatura que empregam modelagem analítica, optamos por manter esse padrão. A modelagem oferece uma visão mais preditiva e possui um alto poder de representatividade, sendo um dos principais benefícios seu baixo custo e a relativa aceitação dos resultados.

3. **Desempenho e disponibilidade:** Neste trabalho avaliamos o desempenho e a disponibilidade do modelo proposto. A avaliação de desempenho levou em consideração métricas como o MRT, o DP e a utilização de recursos. Parâmetros de disponibilidade foram colocados no modelo a fim de impactarem sobre o desempenho analisado bem como serem utilizados para uma análise de sensibilidade através de um cálculo de *Percentage Difference*. A integração dessas duas avaliações enriquece a dissertação, proporcionando maior valor na análise e no planejamento de *Smart Hospitals*.

4. **Análise de sensibilidade:** Neste trabalho foi empregada uma análise de sensibilidade com o DoE em um Smart Hospital. A análise com o DoE tem o objetivo de identificar como os fatores e suas relações entre si influenciam no desempenho da métrica MRT. O objetivo foi obter a maior quantidade de informações com o mínimo de experimentos e identificar os fatores críticos que impactam no modelo proposto.

Os trabalhos acima relacionados na Tabela 3 possuem em sua proposta de arquitetura apenas uma fonte de dados de requisições, com exceção dos trabalhos de Andrade e Nogueira (2020b) e Araujo et al. (2014), que baseiam-se em uma arquitetura de smart hospital com duas entradas de dados de forma simultânea. O primeiro propõe uma abordagem baseada em redes de Petri para modelagem e análise de soluções de recuperação de

Tabela 3 – Trabalhos relacionados.

Trabalho	**Contexto**	**Método de Avaliação**	**Desempenho**	**Disponibilidade**
Silva et al. (2021b)	*General Healthcare*	Modelagem	*Mean Response Time, Utilization, Drop Rate* and *Throughput*	×
Chen et al. (2011)	*Specific Smart Hospital*	Simulação	*Scheduling Policies*	×
Gomez-Sacristan et al. (2015)	*Specific Smart Hospital*	Simulação	*Human Type Communications and Machine Type Communications*	×
Rodrigues et al. (2021b)	*Specific Smart Hospital*	Modelagem	*Mean Response Time, Utilization, Discard Number and Throughput*	✓
Khayal e Farid (2021)	*General Healthcare*	Modelagem	*Healthcare Delivery System and Health State Evolution*	×
Nguyen et al. (2021b)	*Specific Smart Hospital*	Modelagem	*Recover Token Rate, Mean Response Time, Drop Probability, Throughput and Utilization*	✓
Santos et al. (2020)	*General Healthcare*	Modelagem	×	✓
Kafhali e Salah (2019)	*General Healthcare*	Modelagem	*Mean Response Time*	×
Rodrigues et al. (2019)	*Specific Smart Hospital*	Modelagem	*Mean Response Time, Utilization, Discard Number and Throughput*	×
Santos et al. (2022a)	*General Healthcare*	Modelagem	×	✓
Lisboa et al. (2018)	*General Healthcare*	Modelagem	×	✓
Nguyen et al. (2021c)	*General Healthcare*	Modelagem	×	✓
Valentim et al. (2023)	*GeneralHealthcare*	Modelagem	×	✓
Strielkina et al. (2018)	*General Healthcare*	Modelagem	×	✓
Tang e Xie (2021)	*General Healthcare*	Modelagem	*Probabilities of Full Service, Degraded Service and System Unavailability*	✓
Santos et al. (2018)	*General Healthcare*	Modelagem	*Service Time and Throughput*	✓
Rodrigues et al. (2020)	*Specific Smart Hospital*	Modelagem	×	✓
Andrade e Nogueira (2020b)	*General Healthcare*	Modelagem	*RTO, Delivery Probability*	✓
Araujo et al. (2014)	*General Healthcare*	Modelagem	*Delivery Probability*	✓
This work	*Specific Smart Hospital and General Healthcare*	Modelagem	*Mean Response Time, Drop Probability and Utilization*	✓

desastres em um sistema IoT de saúde. O segundo apresenta um modelo de alto nível que caracteriza o comportamento de um sistema mHealth, cujo principal objetivo é identificar a probabilidade de uma mensagem ser entregue no tempo t, considerando que vários componentes envolvidos podem falhar. A base teórica destes trabalhos auxiliaram de forma especial na elaboração do presente trabalho, o qual foca em gerar *insights* de planejamento no dimensionamento da capacidade dos componentes que compõe a arquitetura proposta.

4 Metodologia

O objetivo principal deste trabalho é propor um modelo multicamadas de uma arquitetura básica de *smart hospital* que possua duas fontes de dados simultâneas. Na avaliação do modelo foi observado aspectos de métricas de desempenho e disponibilidade em dois cenários de variação de configuração. A Figura 7 apresenta um fluxograma que resume a estratégia metodológica que foi desenvolvida pelo autor deste trabalho e que serve como base para utilização dos leitores.

Definição de Arquitetura: Definição dos componentes que fazem parte do sistema distribuído proposto. Bem como a organização e disposição das camadas de processamento envolvidas e habilitação de duas fontes de dados simultâneas. A ideia é projetar um sistema que integre os diversos sensores IoHT com outros dispositivos comumente utilizados que forneçam dados de saúde e ultrapasse as barreiras do hospital para possibilitar uma maior abrangência da assistência fora do ambiente hospitalar.

Reutilizar Modelo e Definições: Caso o avaliador queira utilizar o modelo base com as métricas envolvidas e os parâmetros de configuração definidas por este autor, poderá partir diretamente para a definição dos cenários de avaliação. Caso queira ajustar alguma definição, segue para o fluxo de atividade padrão.

Identificação das Métricas de Interesse: As métricas de interesse são definidas a serem utilizadas na análise de desempenho e performabilidade. O MRT é fundamental para comparar os tempos de resposta dentre os cenários e configurações de capacidades implementadas. A métrica de Utilização aferiu os componentes Antena, Máquina de *Edge*, Máquina de *Fog* e Máquina de *Cloud*. A métrica de *Drop Probability* foi utilizada para analisar o comportamento de possíveis descartes em um cenário de sobrecarga de requisições.

Definição dos parâmetros Os parâmetros são definidos, conforme a literatura e experiências anteriores. Esses parâmetros definem o comportamento e capacidade de recursos de cada componente. Neste trabalho, os parâmetros adicionados foram os tempos MTTR e MTTF dos componentes, capacidades e tempos de processamento.

Criação do Modelo O modelo SPN é desenhado com a ferramenta *Mercury*. Os componentes definidos na arquitetura são transportados, ligados entre si e os parâmetros definidos são configurados. O modelo possui também aspectos de disponibilidade com a utilização de "losangos" de MTTR e MTTF em conjunto com arcos inibidores no início de cada componente. Adicionou-se a figura de fila de processamento nos três últimos componentes do sistema (*edge, fog e cloud*). A escolha do modelo de filas se dá para um maior controle das requisições que possam não serem processadas de forma correta em um

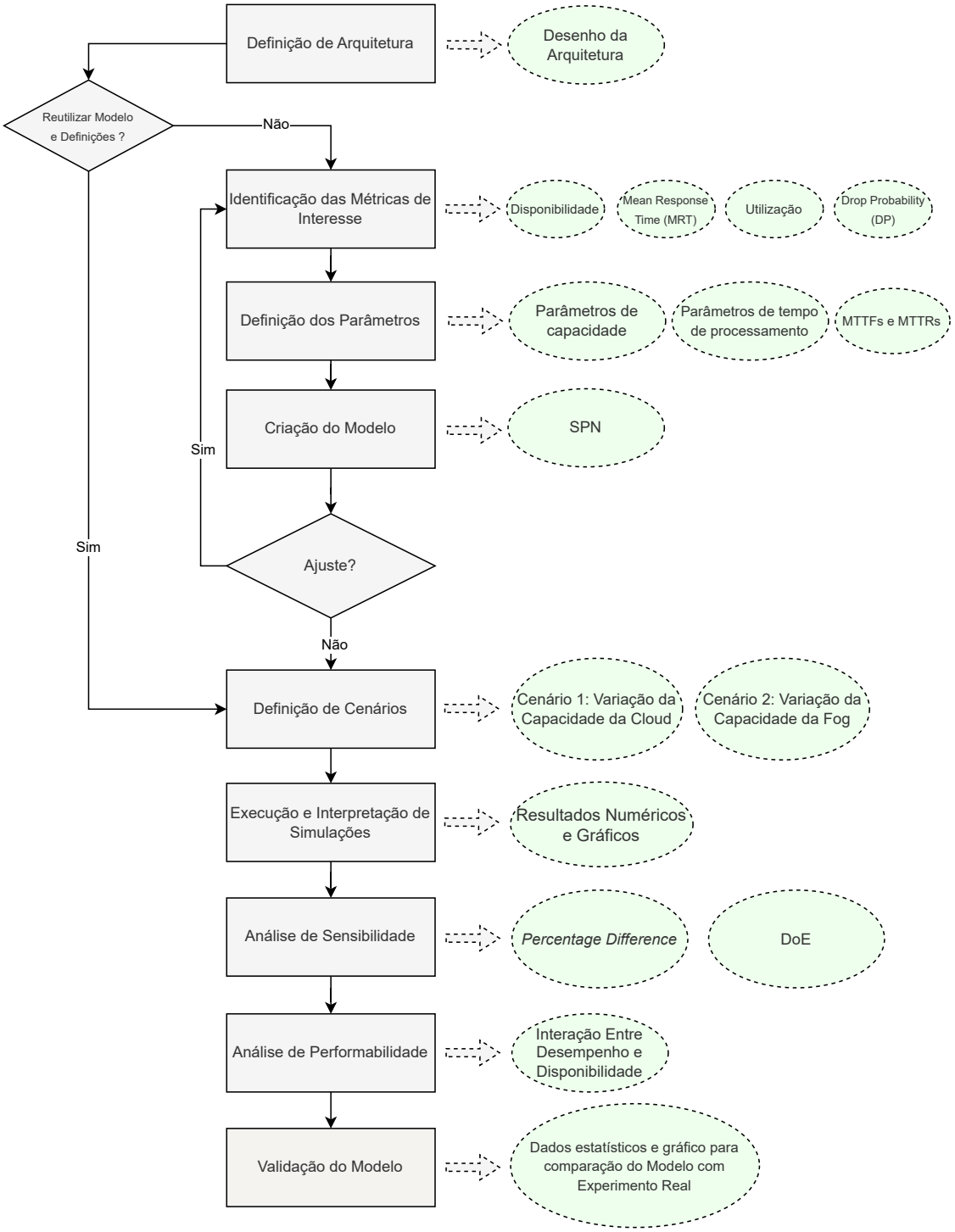


Figura 7 – Metodologia utilizada na Dissertação

possível cenário de sobrecarga. Como contribuição importante, foi levado em consideração a modelagem de dois conjuntos de fonte de dados que alimentam o sistema de forma simultânea.

Ajuste Após a concepção do modelo, o experimento é realizado através de Simula-

ção Estacionária. Os dados são coletados e analisados inicialmente em gráfico de planilha afim de que se encontre e ajuste os parâmetros que retornem a maior quantidade de informação possível. Bem como reavaliar a definição das métricas inicialmente escolhidas.

Definição de Cenários De posse do modelo, foi definido dois cenários de variação de capacidade de processamento. As configurações foram variadas nas duas últimas camadas do modelo, fog e cloud. O Cenário 1 analisa a variação do número de contêineres na cloud e o Cenário 2 a mesma variação somente na fog.

Execução e Interpretação de Simulações: Organizou-se os dados resultantes do experimento na ferramenta Mercury e posteriormente plotados com a ferramenta R Studio para geração dos gráficos. Foi realizado ajustes nos gráficos e legendas para melhor visualização, afim de facilitar o discorrer da análise. Foi realizada uma análise minuciosa do comportamento de cada métrica definida em relação aos componentes e configurações de capacidade dos cenários estudados. Dados numéricos importantes e inferências foram citadas, bem como a comparação entre as informações gráficas dos cenários estudados.

Análise de Sensibilidade: Utilizou-se uma análise *Percentage Difference* para encontrar o índice de sensibilidade dos componentes em relação à disponibilidade total do sistema. E para encontrar os parâmetros de desempenho mais influentes, foi utilizado uma análise DoE sobre as capacidades de processamento dos componentes. No DoE, analisou-se a influência desses fatores em relação à métrica MRT. Essa análise apresenta resultados considerando fatores e níveis predefinidos. A partir disso, é possível identificar os fatores mais relevantes para os resultados das métricas escolhidas, e o modo como a interação entre os fatores e as variações dos seus níveis impactam no desempenho.

Análise de performabilidade: Com a análise *Percentage Difference* e a análise DoE, foi escolhido dois parâmetros para análise de performabilidade, um de disponibilidade (MTTF_CO) e outro de desempenho (C_C). Os valores foram variados para mais e para menos conforme uma escala, e utilizados para medir o valor relativo do MRT do sistema no bojo desta interação. Os dados foram agrupados e analisados em um gráfico de superfície 3D.

Validação do Modelo: Capítulo dedicado para a validação do modelo. Um ambiente de sistema distribuído em máquinas virtuais é utilizado e parametrizado conforme parte do modelo SPN. Os dados reais são coletados para comparar e validar os dados do modelo analítico.

A seguir, é apresentado o Capítulo dedicado à Arquitetura e ao Modelo SPN, que representa as atividades iniciais na construção deste trabalho.

5 Arquitetura e Modelo SPN

Este capítulo apresenta o projeto de arquitetura e o modelo transposto em SPN. Observando o contexto, foi definido os componentes que atendam às funcionalidades do sistema. Partindo da definição da arquitetura, um modelo SPN foi criado e seus parâmetros definidos.

5.1 Arquitetura Computacional de um *Smart Hospital*

Esta seção apresenta uma arquitetura proposta de *Smart Hospital* com níveis multicamadas de processamento e com duas fontes de dados simultâneas.

A arquitetura proposta apresentada na Figura 8 tem o objetivo de sintetizar a interação dos componentes necessários para um sistema de *smart hospital*. A arquitetura contempla uma camada de geração de dados e três camadas em sequência de processamento dos dados. A camada de geração de dados é composta por duas fontes. A primeira é uma fonte interna que representa os sensores e dispositivos IoT. Tais dispositivos monitoram as informações vitais do paciente. Esses dados também podem ser complementados com informações contextuais, como dias de internação, hora, temperatura ambiente, quantidade de pacientes na sala, etc. Conhecer o contexto nos permite identificar padrões incomuns e fazer inferências sobre a situação. No hospital, cada grupo de pacientes internos é designado a um ponto de acesso via wi-fi organizado de acordo com os andares e pavimentos do prédio, por exemplo. Um roteador central conecta os APs e recebe todas essas requisições internas.

A segunda fonte é uma fonte externa que representa os dispositivos que possam fornecer dados de saúde de pacientes fora do hospital no pós-alta. Fora do hospital, podem ser usados *smartwatches*, *smartphones*, sensores vestíveis, PDAs, Micro-dispositivos de sinais vitais, etc. Os dados são transmitidos por uma rede 5G, por exemplo. De fora do hospital, os dados são transmitidos por torres de transmissão, mas esta questão pode ser abstraída caso os dispositivos tenham acesso à rede de ambientes fechados. Como exemplo de abstração podemos utilizar uma faixa dedicada da banda 5G ou até mesmo uma torre de transmissão via rádio. As duas fontes de dados alimentam o *gateway* de borda.

Após a admissão, a requisição passa por três camadas de processamento em sequência: *edge*, *fog* e *cloud*. As camadas podem ter responsabilidades com níveis diferentes de criticidade, a *cloud* pode efetuar trabalhos mais pesados do que a *edge*, por exemplo.

A máquina de *edge* permanece interna à infraestrutura do hospital e seu papel é de receber através de duas interfaces de rede as requisições internas e externas de forma

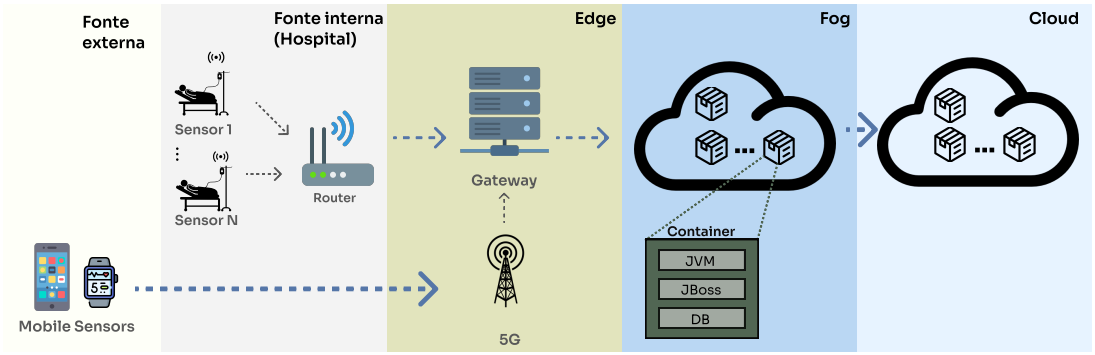


Figura 8 – Arquitetura de um sistema de *smart hospital* com duas fontes de dados.

igualitária, ou seja, compartilhando a mesma fila para as duas entradas. Porém, o poder de processamento deve ser priorizado para as requisições advindas do hospital, por terem uma maior nível de criticidade.

A camada de *fog* baseia-se em receber diretamente as requisições da máquina de *edge* e os processa paralelamente em contêineres, onde cada instância conta com três aplicações: JVM, JBoss e Banco de Dados Mysql. O Mysql é utilizado para guardar os dados tratados pela edge. A JVM e o JBoss são utilizados para rodar aplicação de processamento em Java.

A camada de *cloud*, bem como na *fog*, utiliza contêineres de processamento em seus dados recebidos, efetua o processamento final e armazena a informação. Apesar de conter uma infraestrutura semelhante à camada de *fog*, seus parâmetros de tempo de processamento e capacidade são abstraídos de forma diferente por estarem em outro contexto.

Cada camada possui objetivos de processamento diferentes. Para demonstrar o comportamento deste fluxo damos o seguinte exemplo prático: ao receber as requisições de *smartphones* (fonte externa) e sensores vitais dos pacientes internados (fonte interna), a máquina de *edge* é responsável pelo tratamento inicial dos dados de saúde. Os dispositivos internos e externos possuem parâmetros diversos e, portanto, se faz necessário uma padronização desses dados para que a próxima camada os processe de uma forma mais rápida e unificada. Na camada de *fog*, os dados transformam-se em informação e alimentam o sistema do smart hospital com as devidas inferências, sugestões de acompanhamento e procedimentos de saúde para o paciente em questão, como por exemplo alertar ao médico responsável que a saturação de um paciente está baixa e necessita com urgência de intervenção ou enviar mensagem aos contatos de emergência de um paciente pós alta que tratou problema cardíaco recentemente no hospital e encontra-se naquele momento com a frequência alta mesmo estando em repouso.

Na última camada (*cloud*), é realizada a conferência e armazenamento das informações, bem como o controle e gerenciamento de *feedback* sobre os procedimento adotados

com o auxílio do sistema pelos profissionais do hospital.

Uma questão muito importante é a abstração do que significa capacidade de processamento em cada camada. Na camada da *edge*, é definido uma máquina física de alta performance que compartilha uma mesma fila para as entradas de requisições e que contêm duas máquinas virtuais, uma para processar as requisições internas (do roteador) e outra para as externas (da antena). Para a máquina virtual das requisições internas, o poder de processamento deve ser maior do que na máquina das requisições externas, visto que os dados dos pacientes internados devem ser priorizados no decorrer do fluxo sistema.

Na camada de *fog* e *cloud*, a capacidade refere-se ao número de contêineres paralelos para processamento. Tal abstração teve papel importante no planejamento das capacidades e tempos de processamento do modelo. Na seção 5 é visto que tais capacidades são configuradas associando número de *tokens* disponíveis com respectivo tempo médio de serviço.

5.2 Modelo SPN

A Figura 9 apresenta o modelo SPN proposto. Todos os componentes do modelo seguem os mesmos padrões de nomes usados na descrição da arquitetura. A avaliação do modelo foi feita por meio da ferramenta Mercury (MACIEL et al., 2017).

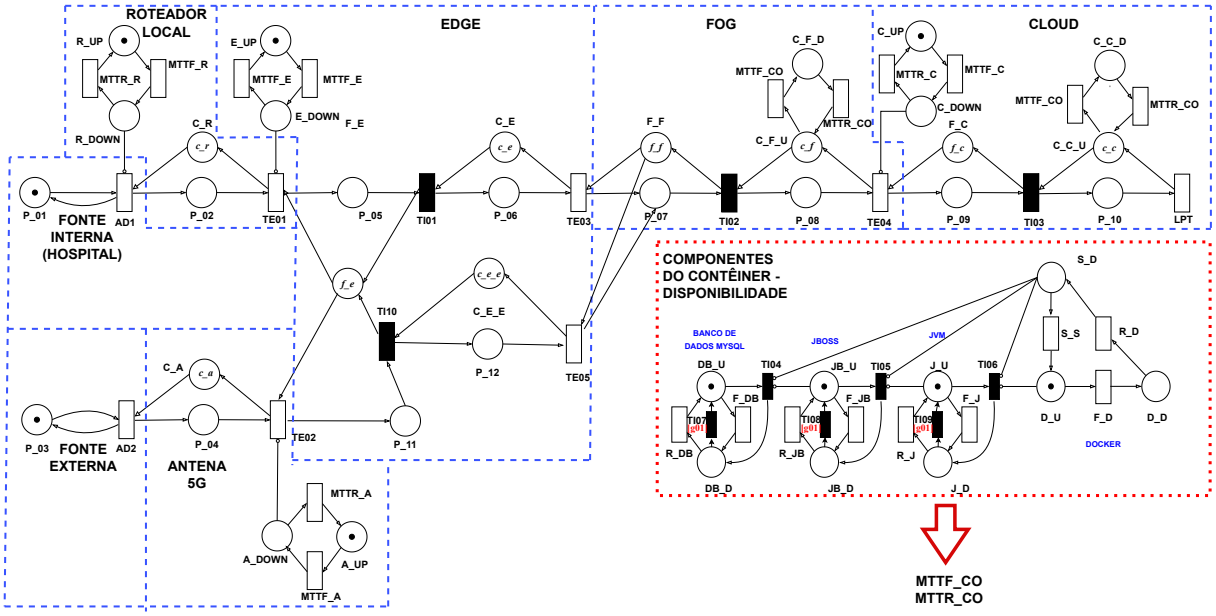


Figura 9 – Modelo base em SPN.

Os *tokens* são gerados seguindo uma distribuição específica, normalmente exponencial para este tipo de contexto. Os tempos entre chegadas de requisições são setados nas transições AD1 e AD2, correspondendo às entradas interna e externa respectivamente,

conforme Figura 10. Após o AD1, os *tokens* da fonte interna vão para o roteador. Após o AD2, os *tokens* da fonte externa seguem para a antena 5G.

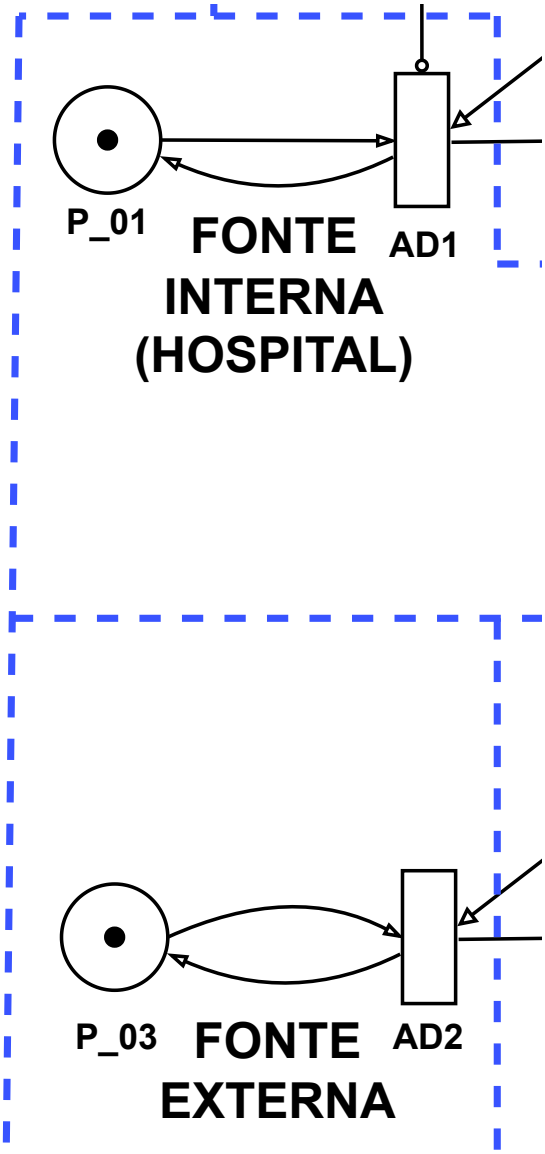


Figura 10 – Componentes de fontes de dados do modelo SPN.

A Figura 11 apresenta os componentes roteador, antena e máquina de *edge*. O lugar P_02 que pertence ao roteador central do hospital recebe os *tokens* da fonte interna e entra em uma fila de encaminhamento de tamanho *c_r*. O *delay* de encaminhamento do roteador é definido por TE01 e para que haja esse encaminhamento o mesmo precisa estar disponível (R_UP = 1). A antena recebe os *tokens* em P_04 e também quando disponível (A_UP = 1) os encaminha obedecendo ao tempo de TE02. Roteador e antena enviam seus *tokens* para a máquina de *edge* e entram numa fila de espera de processamento representado pelo triângulo cujo lugar de capacidade é dado por F_E. P_05 recebe os *tokens* vindo do roteador e P_11 recebe os da antena. Ambos compartilham a mesma capacidade de fila de tamanho *f_e*.

Após passarem pela fila, os *tokens* oriundos de P_05 são enviados para o triângulo posterior que possui uma capacidade de processamento simultâneo de tamanho *c_e* e processados com o tempo definido em TE03, já os *tokens* de P_11 são processados pelo tempo definido em TE05 com uma capacidade simultânea de tamanho c_e_e.

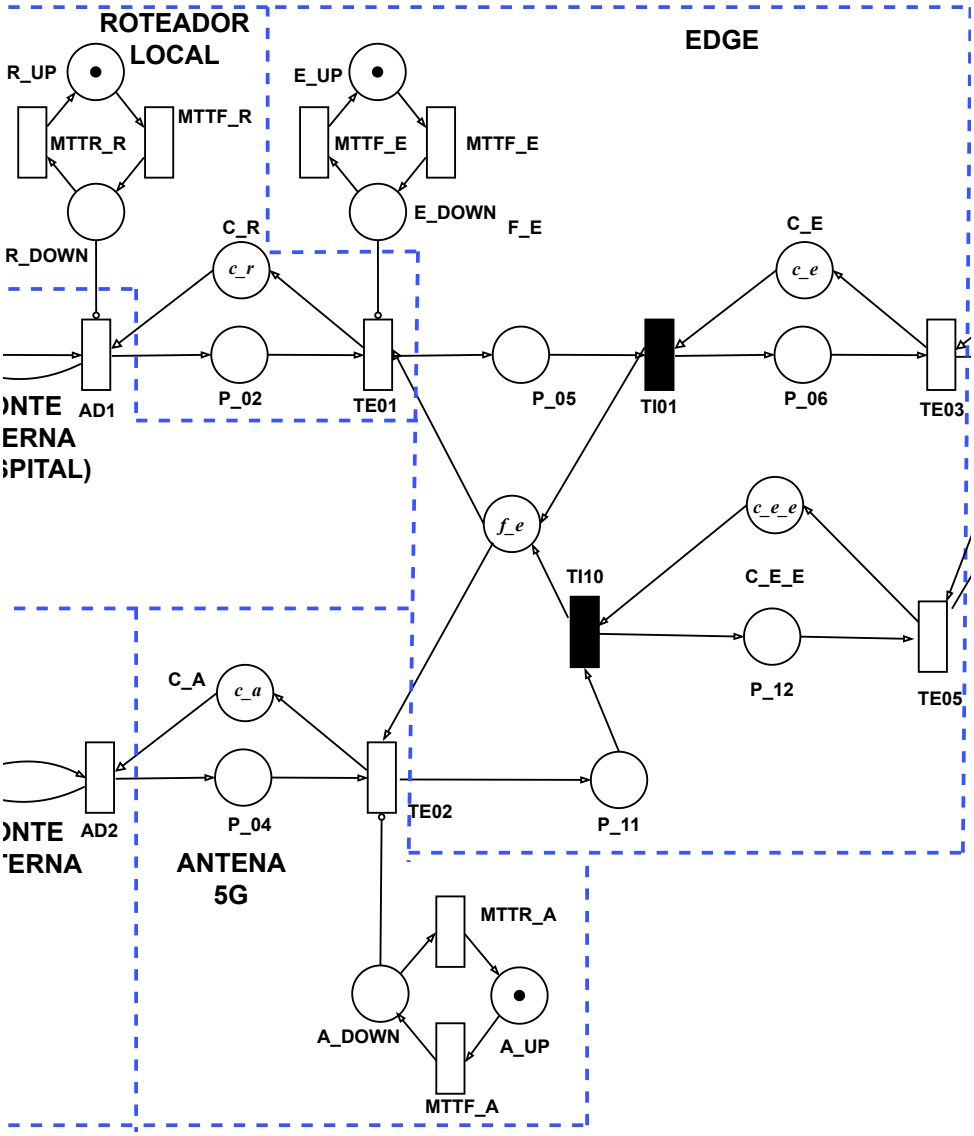


Figura 11 – Roteador, antena e máquina de edge do modelo.

Conforme Figura 12, os *tokens* seguem para a camada de *fog* e *cloud*. Ambas possuem filas de entrada de dados e processamento paralelo, aqui idealizadas como contêineres. A fila da máquina de *fog* é definida em *f_f* e a quantidade de contêiner de processamento simultâneo em *c_f*. De forma semelhante, o tamanho da fila da máquina de *cloud* é definida em *f_c* e seu número de contêineres em *c_c*. Os *tokens* saem da *fog* obedecendo o tempo TE04 e precisam que a máquina de *cloud* esteja disponível (C_UP = 1). O último tempo de processamento do sistema é definido em LPT (*Last Processing Time*), ele é composto pelo tempo utilizado das três aplicações em cada contêiner da *cloud*. Após passar pelo LPT, o *token* sai do modelo representando uma requisição finalizada.

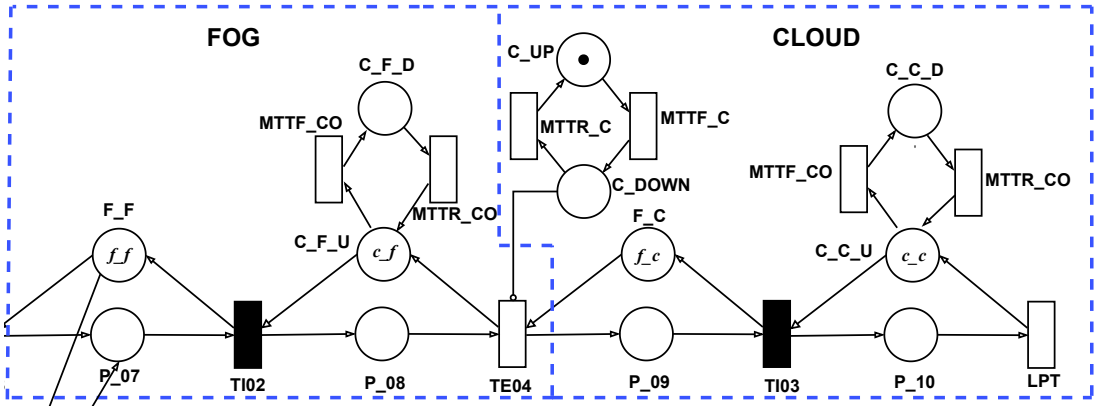


Figura 12 – Máquinas de fog e cloud do modelo.

O fluxo dos *tokens* depende, além do processamento nas camadas, da disponibilidade modelada na entrada de alguns componentes ligados ao modelo com arcos inibidores. São eles: roteador, máquina de *edge*, antena 5G e máquina de *cloud* - não encontramos na literatura os parâmetros específicos para incluir parâmetros de disponibilidade de uma máquina de *fog*. Os componentes que compõem cada instância de processamento nas camadas de *fog* e *cloud* possuem, assim como os demais acima, restrições de disponibilidade. Para que um *token* seja processado é necessário que pelo menos um contêiner na sua camada esteja disponível, e para que o contêiner esteja disponível é necessário que suas aplicações (JVM, JBoss e DB) também estejam. O modelo tracejado em vermelho (Figura 13) configura a disponibilidade do contêiner e suas três aplicações: Banco de Dados, JBoss e JVM. Para que uma instância de contêiner Docker esteja disponível é necessário que também suas aplicações estejam ((D_U = 1) AND (DB_U = 1) AND (JB_U = 1) AND (J_U = 1)). No momento da falha do Docker (F_D) as demais aplicações caem em cascata. Após a recuperação do Docker(R_D), é necessário que se passe um tempo para restabelecer os serviços das aplicações (S_S), o que ativa a expressão de guarda [g01] (#S_D=1) nas transições imediatas TI07, TI08 e TI09. Tal modelo de auxílio (13) foi utilizado antes do modelo base para computar os valores de MTTF_CO e MTTR_CO que representam o tempo médio de falha e recuperação do contêiner, respectivamente.

Os parâmetros de capacidades foram definidos de forma empírica baseado em experiências anteriores. Os parâmetros de transição temporizada de processamento, *delay*, MTTR e MTTF, os quais foram configurados com variação de probabilidade exponencial e influenciaram diretamente na calibração do sistema, tomaram como base os números encontrados na literatura - Tabelas 4 e 8, elaboradas com os valores de (COSTA et al., 2016), (BRITO et al., 2021), (SANTOS et al., 2022b). Tais parâmetros devem ser definidos para a realidade específica do avaliador.

O objetivo do modelo proposto é auxiliar administradores de sistemas, que utilizam essa arquitetura multicamada de duas fontes de dados, na tarefa complexa de ajustar vários parâmetros adequadamente para alcançar níveis de desempenho desejáveis. Logo, o

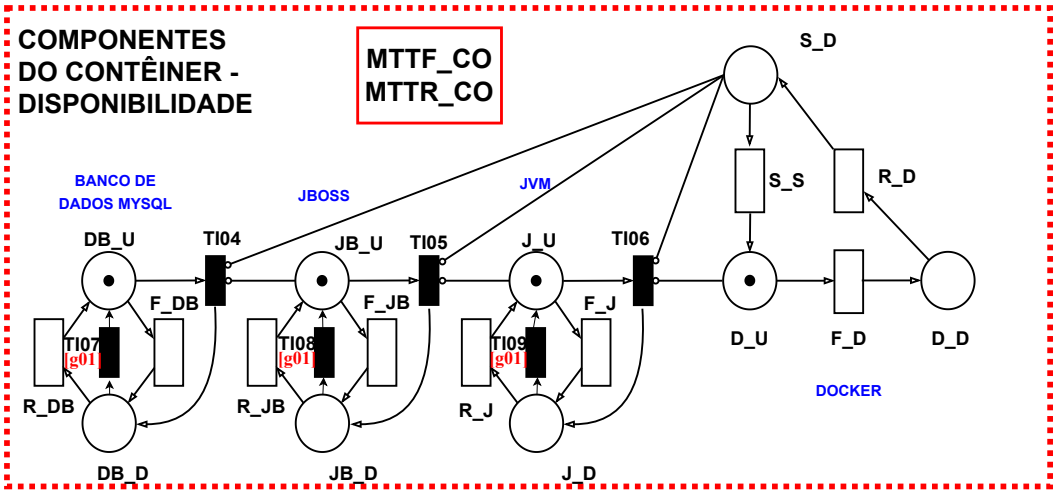


Figura 13 – Modelagem de um contêiner utilizada para obtenção de MTTR_CO e MTTF_CO.

Tabela 4 – Descrição das principais transições do modelo base.

Tipo	**Elemento**	**Descrição**	**Valor (ms)**
Transições temporizadas	AD1	Tempo de espera entre as requisições internas	10 a 100
	AD2	Tempo de espera entre as requisições externas	10 a 100
	TE01	Tempo de envio da requisição do router até a edge	20
	TE02	Tempo de envio da requisição da antena até a edge	20
	TE03	Tempo de processamento na edge (requisições internas)	30
	TE04	Tempo de processamento na fog	50
	TE05	Tempo de processamento na edge (requisições externas)	30
	LPT	Tempo de processamento na cloud - último processamento	250
	S_S	Tempo para iniciar os serviços após a recuperação do contêiner	500
Transições imediatas	TI01	Saída interna da fila na edge	-
	TI02	Saída da fila na fog	-
	TI03	Saída da fila na cloud	-
	TI04, TI05 e TI06	Transições de dependência entre os componentes	-
	TI07, TI08 e TI09	Usadas pra forçar reinício dos componentes pela [g01]	-
	TI10	Saída externa da fila na edge	-

modelo deve ser útil para checar o efeito de mudanças no sistema, antes mesmo que elas sejam implementadas. O modelo proposto é generalizável para outros contextos, porém foi inspirado em uma arquitetura tradicional de *smart hospital* (RODRIGUES et al., 2021a).

5.2.1 Métricas

As métricas adotadas nesse trabalho são: *Mean Response Time* (MRT), utilização individual de cada componente, e *drop probability* (DP). Para cálculo do MRT foi adotada a Lei de Little (LITTLE, 1961). Tal lei afirma que o tempo médio de resposta (MRT) é proporcional ao número médio de requisições em progresso (*RequestInProgress*) e inversamente proporcional à taxa de chegada (*ArrivalRate*). A taxa de chegada é calculada como o inverso do tempo de espera entre as requisições (*ArrivalDelay*). A Lei de Little requer um sistema estável, ou seja, que possua uma taxa de chegada menor do que a taxa de processamento interno do sistema, assim ele não ficará sobrecarregado.

Normalmente o MRT é igual ao produto da quantidade de requisições em progresso, multiplicado pelo tempo entre chegadas de requisições. No entanto, o modelo proposto possui o diferencial de ter duas entradas de dados ao mesmo tempo. Assim, para calcular o MRT no modelo proposto foi necessário utilizar o tempo da última transição do modelo, denominado LPT (*Last Processing Time*), conforme *Eq01* da Tabela 5. Para encontrar o número de *RequestsInProgress*, é necessário obter a quantidade momentânea total de *tokens* que estão sendo processados desde o início do modelo até o final. Para esse cálculo basta somar a esperança estatística de existir *tokens* em todos os lugares onde a requisição irá passar conforme *Eq02* (RODRIGUES et al., 2021a). Por fim, o MRT deverá ser multiplicado pela probabilidade de se ter ao menos um *token* no place que antecede o LPT (P_10).

A métrica de Utilização para cada componente é calculada pela relação entre quantidade de *tokens* que estão à espera do processamento dividido pela capacidade máxima de seus processadores. A utilização da camada da *edge*, por exemplo, é calculada conforme equação *Eq03*. A utilização da *fog* e da *cloud* seguem a mesma ideia.

Para o cálculo da métrica Drop Probability (DP), são utilizadas as equações *Eq04* e *Eq05*. Nelas são calculadas a probabilidade de haver perdas de requisições causadas pelo enfileiramento e consequentemente o gargalo em um ou mais componentes no momento em que a requisição chega após a admissão no sistema. Do ponto de vista prático, a requisição é descartada quando no momento em que ela tenta entrar no sistema não há mais vagas na capacidade da fila de espera. Como foram consideradas duas fontes de dados, para calcular o DP foi utilizada uma decisão de dois caminhos (*OR*) entre os DPs nas duas fontes de dados (interna e externa), conforme *eq06*.

Tabela 5 – Equações utilizadas nas métricas.

Ref	**Equação**
Eq01	$MRT = (RequestsInP\ rogress \times LPT) \times P\{(\#P_10 > 0)\}$
Eq02	$RequestsInProgress = (Esp(\#P_05)+Esp(\#P_06)+Esp(\#P_07)+Esp(\#P_08)+Esp(\#P_09)+Esp(\#P_10)+Esp(\#P_11)+Esp(\#P_12))$
Eq03	$U_EDGE = Esp(\#P_06)/C_E$
Eq04	$DP1 = P\{(\#P_01 > 0)AND(\#C_R = 0)\}$
Eq05	$DP2 = P\{(\#P_03 > 0)AND(\#C_A = 0)\}$
Eq06	$DP_TOTAL = P\{((\#P_01 > 0)AND(\#C_R = 0))OR((\#P_03 > 0)AND(\#C_A = 0))\}$

6 Resultados e Discussão

Este capítulo apresenta os resultados obtidos através da análise das métricas (Seção 5.2.1) em dois cenários de simulação, bem como duas análises de sensibilidade: uma baseada na disponibilidade do sistema e outra realizada com a técnica DoE sobre a métrica de Tempo Médio de Resposta (MRT). Ao final, é realizado uma análise de performabilidade em relação aos parâmetros mais influentes de disponiblidade e desempenho do sistema.

6.1 Análise de Desempenho

Esta seção apresenta os resultados de desempenho do modelo proposto. A Tabela 7 apresenta os dados da parametrização do modelo com a configuração dos três macro-componentes de processamento (camada de *edge*, camada de *fog* e camada de *cloud*). As filas para redirecionamento das requisições do roteador e da antena também são definidas. Os conjuntos de resultados estão subdivididos em dois cenários conforme Tabela 6.

Tabela 6 – Configurações dos cenários analisados.

Cenário	Número de contêineres da *fog*	Número de contêineres da *cloud*
1	50	5, 25, 50, 75 e 100
2	[5, 25, 50, 75 e 100]	50

Neste estudo de caso variamos a taxa de chegada de requisições em função da variação da capacidade de processamento da *cloud* (Cenário 1) e da *fog* (Cenário 2). A taxa de chegada foi variada nas duas fontes de dados simultaneamente entre 0.01 msg/ms até 0.1 msg/ms, com 20 pontos amostrais. As capacidades adotadas referem-se à quantidade de contêineres paralelos em cada cenário, a saber: C01 = 5, C02 = 25, C03 = 50, C04 = 75, C05 = 100. As métricas analisadas foram: MRT, utilização da *edge*, utilização da torre, utilização da *fog*, utilização da *cloud* e *drop probalility* (DP).

Tabela 7 – Definição da quantidade de *tokens*.

Marcação	**Descrição**	**Quantidade**
c_r	Fila do roteador	50
f_e	Fila da *edge*	100
c_e	Cores de processamento na *edge* (requisições internas)	75
c_e_e	Cores de processamento na *edge* (requisições externas)	25
f_f	Fila da *fog*	150
c_f	Número de contêineres na fog	50 (no Cenário 1)
f_c	Fila da *cloud*	300
c_c	Número de contêineres na *cloud*	50 (no Cenário 2)
c_a	Fila da antena	50

6.1.1 Cenário 1 - Variação da capacidade da Cloud

Esta subseção apresenta os dados de desempenho referentes ao Cenário 1. Os dados analisados refletem a variação das taxas de chegada (Interna e Externa) em relação a variação do número de contêineres de processamento da *cloud*. A Figura 14(a) apresenta os dados da utilização da camada de *edge*. As configurações de 50, 75 e 100 contêineres não resultaram em significante utilização, os dados mantiveram-se sobrepostos e lineares e não ultrapassaram os 5%, salvo a configuração de 50 que atingiu 18% de utilização na taxa mais alta. Já as configurações de 5 e 25 contêineres obtiveram um alcance próximo a 100%. Um aumento acentuado logo nas menores taxas de chegada pode ser observado com a capacidade C01, saltando de menos de 1% para 83.90%, até chegar em 97.86%. Nessa configuração, após uma taxa de $\approx$ 0.03 msg/ms a utilização varia até a taxa de chegada final de forma não uniforme entre $\approx$ 82% e 97%. Com a capacidade C02 a utilização mostrou-se significativa a partir de uma taxa média de 0.05 msg/ms, ultrapassando os 50% e chegando até $\approx$ 89% na maior taxa de chegada (0.1 msg/ms). A variação da utilização da *edge* pela taxa de chegada alcançou números elevados com essas configurações de 5 e 25 contêineres.

A Figura 14(b) apresenta a utilização da *fog*. Os resultados mostraram-se semelhantes aos da *edge* nas configurações de 5 e 25 contêineres, no entanto com C01 a utilização já inicia com 34% na menor taxa de chegada. Nas configurações C03, C04 e C05 conseguiu-se atingir uma baixa utilização em torno de 20% na maior taxa de chegada. A utilização da Torre apresentada na Figura 14(d) também assemelha-se com a *edge*, evidenciando uma equilibrada carga de trabalho dos componentes que recebem as requisições do sistema. A fim de se obter uma maior utilização dos recursos investidos, em relação aos componentes iniciais do sistema, considera-se mais vantajoso investir em menos recursos computacionais do que os definidos nos parâmetros do modelo base.

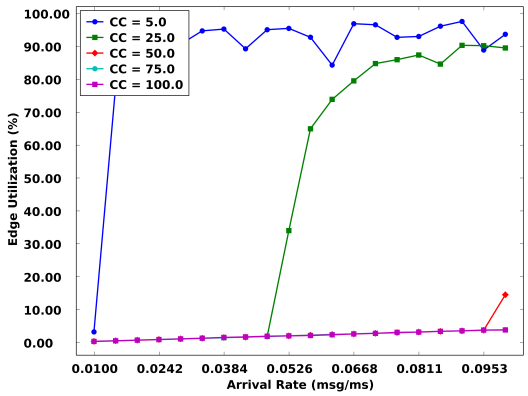
O gráfico da utilização da *cloud* 14(c) foi o que apresentou os dados mais variados e com menos sobreposições. A configuração de 5 contêineres resultou em uma utilização próxima aos 100% durante toda a avaliação, já a configuração de 25 contêineres resultou em sobrecarga de utilização da *cloud* nas taxas a partir de 0.04 msg/ms. As demais configurações mantiveram-se lineares e proporcionais à taxa de chegada, sendo que na taxa de chegada final (0.1 msg/ms) a configuração de 50 contêineres resultou em 97.66% de utilização, a configuração de 75 contêineres obteve $\approx$ 67% e a configuração de 100 contêineres $\approx$ 50%. O resultado da configuração de 50 contêineres apresentou uma utilização da *cloud* praticamente linear e proporcional à taxa de chegada, atingindo quase 100% na maior taxa e iniciando com 10% na menor taxa. Já com a configuração de 25 contêineres, a utilização apresenta índices acima de 99% quando a taxa de chegada atinge na metade da escala de avaliação (em torno de 0.05 msg/ms) e segue até o final com o mesmo comportamento da configuração de 5 contêineres. A configuração de 5 contêineres é a que

mais se difere das demais em relação ao índice medido na taxa de partida, a utilização obtida já inicia com um índice de 98.15%. Portanto, essa configuração fatalmente implica em uma sobrecarga total do sistema.

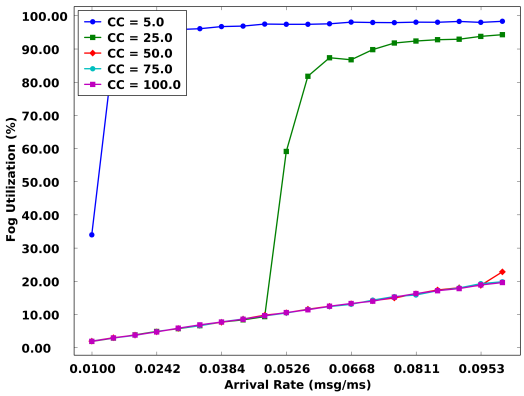
A Figura 14(e) apresenta os dados do MRT. As configurações de 50, 75 e 100 contêineres obtiveram resultados praticamente sobrepostos. Porém, na taxa de chegada final, a de 50 contêineres obteve um tempo de resposta de ≈ 60s e as configurações de 75 e 100 contêineres atingiram um MRT máximo de 19s. O tempo de resposta com a configuração de 5 contêineres atingiu 58s na taxa inicial e já na taxa seguinte chegou em números acima de 160s. Com a configuração de 25 contêineres, a métrica se mantém na faixa de 2s a 8s até uma taxa de ≈ 0.04 msg/ms. Nessa configuração (C02), uma forte aceleração do crescimento do tempo de resposta é apresentada entre a faixa média de ≈ 0.04 msg/ms e ≈ 0.053 msg/ms. A configuração de 25 contêineres de fato é a que entrega os dados mais equilibrados e espaçados do experimento em relação ao MRT com o cenário proposto da análise. Visto a significante diferença de MRT entre as configurações C02 e C03 nas taxas finais, é importante que se busque um número médio entre 25 e 50 contêineres na camada de cloud para que haja um equilíbrio entre recursos investidos e desempenho sem sobrecarga.

A Figura 14(f) mostra a probabilidade de descartes (DP). Somente com as configurações C1 e C2 o sistema consegue verificar algum descarte nas requisições. Nas demais configurações o índice permanece em 0%. Com 5 contêineres, foi observado um índice inicial de 42.58%, com uma taxa de 0.01473 msg/ms e aumentando gradualmente até atingir um índice de 95.72%. Com 25 contêineres, a probabilidade só parte de 0% a partir de uma taxa de chegada de 0.0478 msg/ms, e atinge ≈65% na taxa final. Na configuração de 5 contêineres e variação da taxa de chegada entre 0.01 msg/ms e 0.02 msg/ms, o DP cresce e chega a 63%. Com a configuração de 25 contêineres, o comportamento do DP é relativamente mais linear e proporcional se comparado ao obtido na C1.

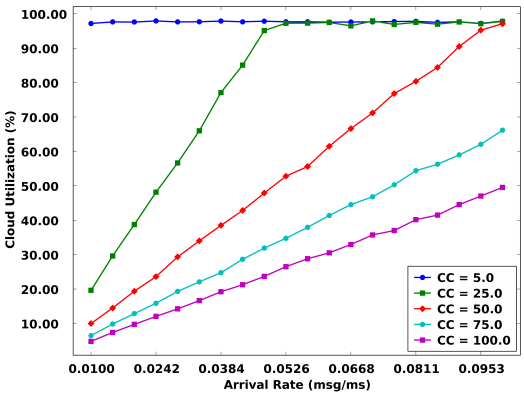
Concluindo, com uma configuração de 25 contêineres na *cloud* o sistema se torna mais controlável a possíveis aumentos repentinos na taxa de chegada das requisições do que com uma configuração de 5 contêineres, e consequentemente mais controlável ao crescimento do número de pacientes que entram e saem do hospital. Em C2 os dados são mais variados e espaçados conforme as escalas das métricas de avaliação. Na faixa mediana da taxa de chegada (entre 0.04 msg/ms e 0.06 msg/ms) há um "gargalo" no processamento das requisições, afetando diretamente nos resultados obtidos. A sobrecarga das requisições é facilmente notada quando a utilização da *cloud* chega a próximo de 100% com uma taxa de chagada de 0.048 msg/ms. A partir dessa taxa, todas as outras métricas aumentam seus índices (inclusive o DP parte de 0%), levando a comprovar que quando a camada de *cloud* chega à utilização máxima, todos os outros componentes e métricas avaliadas são afetados o que por sua vez poderá prejudicar no aumento do fluxo de novos pacientes.



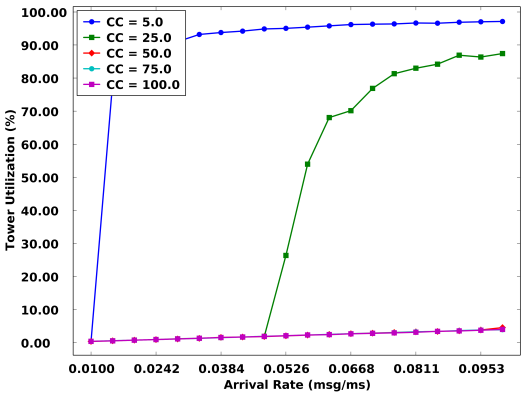
(a) Utilização da *edge*



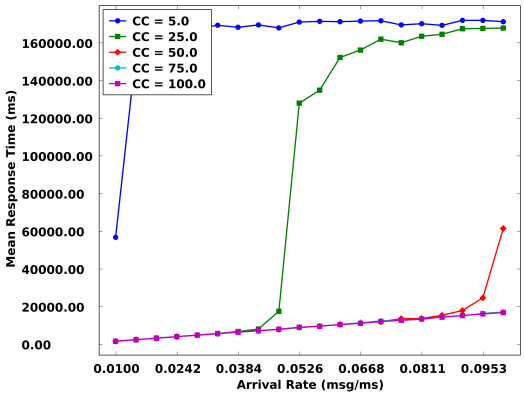
(b) Utilização da *fog*



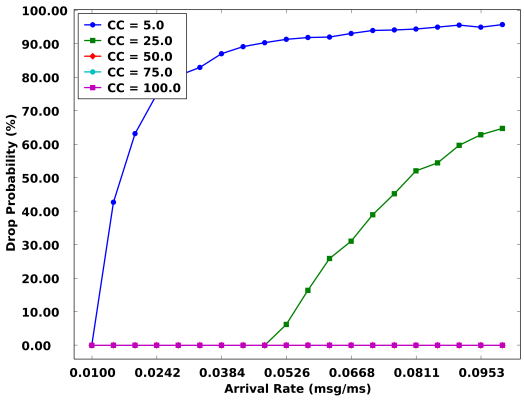
(c) Utilização da *cloud*



(d) Utilização da torre



(e) Mean Response Time (MRT)



(f) Drop Probability (DP)

Figura 14 – Análise de desempenho do cenário 1.

6.1.2 Cenário 2 - Variação da capacidade da Fog

Nessa subseção são apresentados os resultados em relação à variação de número de contêineres da *fog* e fixando o número de contêineres da *cloud* em 50. A Figura 15(a) apresenta os resultados da utilização da camada de *edge*. O comportamento da utilização nas cinco configurações é praticamente igual nas taxas iniciais até 0.047 msg/ms, partindo de um índice de pouco mais de 1% até 3%. O desempenho da métrica manteve-se linear nas configurações C02, C03, C04 e C05 com dados praticamente sobrepostos, sendo que na

C02 obteve-se um índice máximo de 3.95% e na C03, C04 e C05 13.80%, 7.22% e 3.99% respectivamente. A configuração C01 (5 contêineres) apresentou uma forte aceleração saltando de 10.16% para 67.62%, esse crescimento brusco da utilização é verificado em apenas um deslocamento na faixa da taxa de chegada, precisamente entre 0.047 msg/ms e 0.052 msg/ms, a partir daí o índice desacelera o crescimento, porém chega a alcançar 92.29% na taxa final.

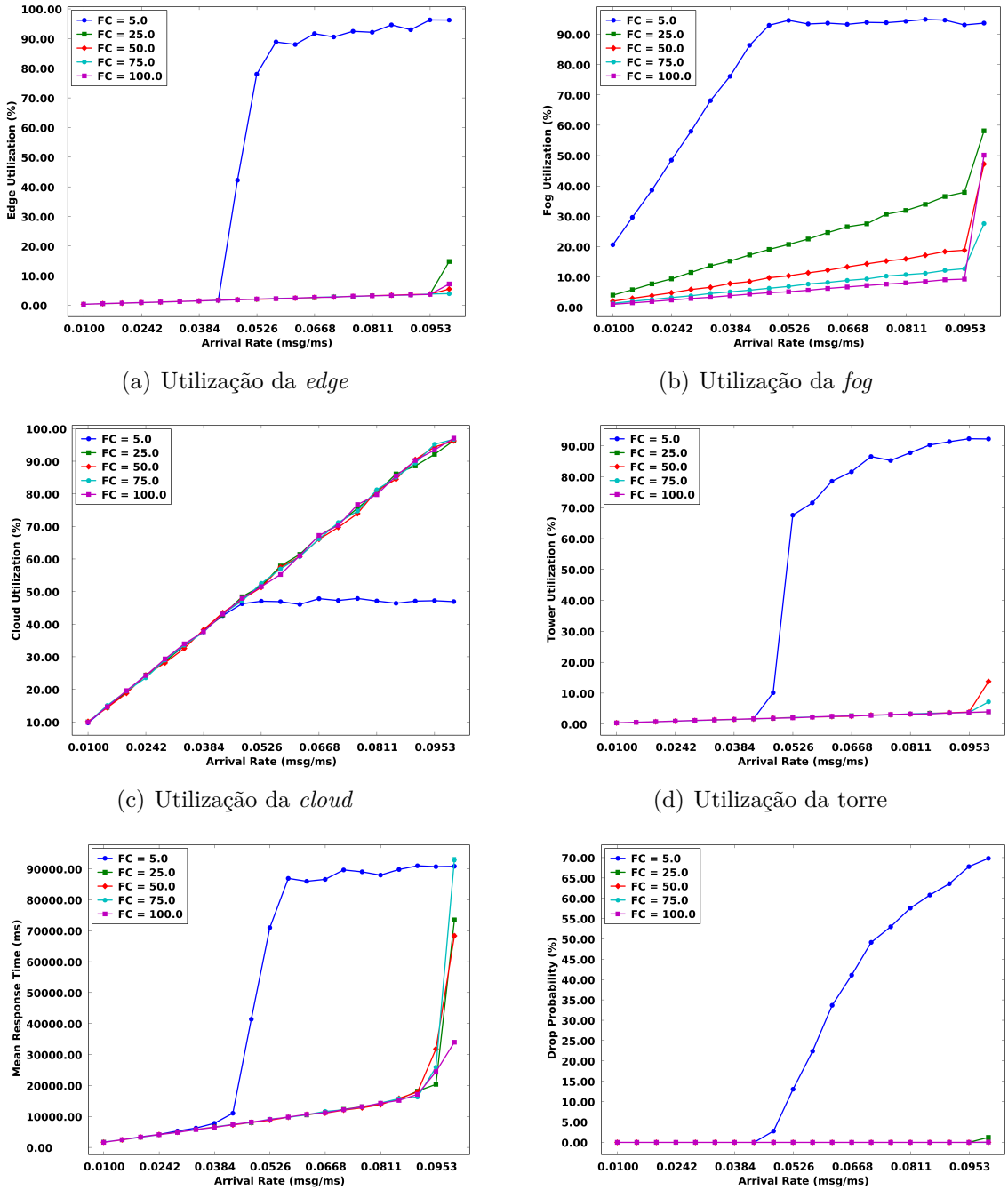
A utilização da *fog* da Figura 15(b) apresenta um desempenho cresceste e sem aceleração nas configurações C02, C03, C04 e C05. Já na C01, ao chegar na taxa de 0.052 msg/ms, a utilização atinge 94.59% e estagna o crescimento. A taxa de chegada inicial (0.01 msg/ms) resultou em uma utilização de 20.62% na configuração de 5 contêineres, 4.03% na de 25 contêineres, e nas demais um índice abaixo de 2%. O resultado obtido na maior taxa foi de 93.69%, 58.19%, 47.26%, 27.62% e 50.18% em C01, C02, C03, C04 e C05, respectivamente. Observa-se que na maior taxa de chegada de requisições a utilização da *fog* acelera em todas as configurações, com isso pode-se levantar a suspeita de que uma taxa de 0.1 msg/ms delimita um limite para que o sistema tenha uma sobrecarga de trabalho.

A Figura 15(c) apresenta o desempenho de utilização da *cloud*. As configurações de 25 a 100 contêineres obtiveram praticamente os mesmos resultados e se comportaram de forma crescente e linear em relação à escala da taxa de chegada, iniciando com um índice de 10% e na taxa final um índice de 97%. A configuração de 5 contêineres comportou-se semelhante as demais até uma taxa de chegada de 0.048 msg/ms, a partir daí a utilização parou de crescer e estacionou em um índice de 48%. A estagnação do crescimento da utilização da *cloud* é resultante da limitação do poder de processamento da camada anterior (*fog*), dessa forma pode-se afirmar que em um sistema distribuído em sequência não adianta somente investir na capacidade de processamento final visto que a camada anterior prover suas requisições.

A Figura 15(d) mostra a utilização da entrada externa (torre). Nas configurações acima de 5 contêineres os índices de utilização apresentam resultados praticamente iguais, partindo de 0.4% e alcançando 3.9% na taxa de chegada final. A configuração de 5 contêineres apresentou uma forte aceleração no crescimento por volta de 0.055 msg/ms saindo de 2% a 68%, a partir daí desacelera e atinge 92% na taxa de chegada final.

A Figura 15(e) traz os dados do MRT. A configuração de 5 contêineres apresenta uma dinâmica em formato de “S”, acelerando o crescimento em uma taxa de chegada de 0.04 msg/ms e desacelerando em 0.06 msg/ms. A métrica nessa faixa passa rapidamente de 10s para 88s e atinge na taxa final um tempo de resposta de 91s. As configurações C2 a C5 resultam em MRT iguais até a penúltima taxa de chegada - em torno de 18s -, porém na taxa final de 0.1 msg/ms a configuração C2 mediu 73s, a C3 mediu 68s, a C4 mediu 93s e a C5 mediu 34s.

O gráfico do DP apresentado na Figura 15(f) informa que somente na configuração de 5 contêineres o sistema obtém probabilidade de descarte. Uma taxa de 0.047 msg/ms resulta em 2.77% de DP, o índice a partir daí cresce de forma contínua e atinge 69.84% na maior taxa de chegada. Também nessa taxa de 0.1 msg/ms, a configuração de 25 contêineres apresenta um discreto DP (1.21%). Em comparação com os resultados do DP no cenário 1 14(f), verificamos que a variação da capacidade na camada de *fog* resulta em uma menor probabilidade de descarte de requisições se comparado com os resultados das mesmas variações na capacidade da camada de *cloud*.



(a) Utilização da *edge*

(b) Utilização da *fog*

(c) Utilização da *cloud*

(d) Utilização da torre

(e) Mean Response Time (MRT)

(f) Drop Probability (DP)

Figura 15 – Análise de desempenho do cenário 2.

Com os resultados desse cenário, podemos observar que com uma capacidade a partir de 25 contêineres na camada de *fog* o sistema como um todo consegue, com uma baixa utilização dos componentes, um MRT e *drop rate* menores se comparados aos resultados do Cenário 1 com a configuração equivalente.

6.2 Análise de Sensibilidade e Performabilidade

Esta seção apresenta duas análises de sensibilidade que são utilizadas para encontrar os parâmetros mais influentes de disponibilidade e desempenho dentro do modelo. Bem como uma análise de performabilidade sobre estes parâmetros em relação à métrica MRT.

6.2.1 Análise de Sensibilidade com Percentage Difference

A continuidade de um sistema em um smart hospital deve ser tratada de maneira prioritária por estar diretamente relacionada à manutenção da vida. Esta análise avalia o impacto dos componentes em relação à disponibilidade no estado estacionário do sistema. Foi utilizada a técnica de *Percentage Difference* (MATOS et al., 2015) para esta análise. Na técnica é definido um valor mínimo e um valor máximo para os parâmetros analisados e para cada definição é calculada uma saída utilizada para determinar a sensibilidade do parâmetro avaliado (SILVA et al., 2022b). A equação para determinar o índice de sensibilidade de cada parâmetro é dada por:

$$S_\theta\{Y\} = \frac{max\ \{Y(\theta)\} - min\ \{Y(\theta)\}}{max\ \{Y(\theta)\}} \tag{6.1}$$

As expressões $max\ \{Y(\theta)\}$ e $min\ \{Y(\theta)\}$ são, respectivamente, os valores máximo e mínimo de saída medidos ao variar o parâmetro θ. Se $Y(\theta)$ é a função que calcula o valor da saída de Y em relação ao parâmetro θ, então $S_\theta\{Y\}$ sempre será um valor positivo e dependerá diretamente da diferença entre o maior e o menor valor de saída calculado na variação e não necessariamente a maior saída será em relação ao valor máximo do parâmetro e vice-versa.

A saída analisada para esta análise é a métrica de disponibilidade total (SILVA et al., 2022a). Para que o modelo reflita disponibilidade no decorrer da simulação, todos os componentes do sistema precisam estar *up* ao mesmo tempo. Os componentes do modelo, bem como as aplicações que pertencem a cada contêiner de processamento possuem parâmetros de MTTF e MTTR. Os valores de MTTF e MTTR de todos os componentes foram variados em 4: 25% e 50% a menos e 25% e 50% a mais do valor base. Para cada parâmetro foi aplicada a Equação 6.1, que por definição precisou medir a disponibilidade em cada uma das 4 variações de valores.

A Tabela 8 apresenta o ranqueamento desses parâmetros pelo índice individual de sensibilidade ($S_\theta\{Y\}$). A transição JB_F (tempo médio para falha do JBoss) obteve o maior índice (0.7700), portanto, a aplicação JBoss que pertence a cada instância de contêiner Docker nas camadas de *fog* e *cloud* é o componente que estatisticamente possui maior poder para influenciar na disponibilidade total do sistema.

Tabela 8 – MTTR e MTTF dos componentes e análise de sensibilidade.

Transição	**Descrição**	**Valor (horas)**	**Índice de Sensibilidade ($S_\theta\{Y\}$)**
F_JB	MTTF do JBoss	4728	0.7700
F_DB	MTTF do Banco de Dados	8643	0.7124
R_DB	MTTR do Banco de Dados	108	0.4521
R_D	MTTR da instância do Docker	0.255	0.3841
MTTF_R	MTTF do router	44957	0.3397
MTTF_E	MTTF da máquina de edge	8760	0.3128
MTTF_A	MTTF da antena	150000	0.2972
MTTF_C	MTTF da máquina de cloud	768.35	0.2859
R_J	MTTR da JVM	72	0.2478
F_D	MTTF da instância do Docker	2516	0.2156
F_J	MTTF da JVM	4728	0.1919
MTTR_R	MTTR do router	5	0.1749
R_JB	MTTR do JBoss	72	0.1088
MTTR_A	MTTR da antena	5	0.1001
MTTR_E	MTTR da máquina de edge	8	0.0461
MTTR_C	MTTR da máquina de cloud	0.74	0.0250

6.2.2 Análise de Sensibilidade com DoE

Esta seção apresenta uma análise de sensibilidade com o método DoE (Design of Experiments) (FEITOSA et al., 2021; SANTOS et al., 2021). O DoE tem o objetivo de identificar como os fatores e suas relações entre si influenciam no desempenho do sistema. Foram analisados 5 fatores: capacidade de transmissão do roteador (c_r), capacidade de transmissão da antena (c_a), quantidade de núcleos nas duas máquinas de processamento na edge (c_e), número de contêineres da máquina de fog (c_f) e número de contêineres da máquina de cloud (c_c). É importante informar que a capacidade c_e representa nos gráficos e resultados a seguir o poder de processamento total da camada de edge (soma dos número de cores de processamento nas duas máquinas virtuais).

Como o modelo é composto por cinco macrocomponentes (roteador, antena, máquina da camada de edge, máquina da camada de fog e máquina da camada de cloud), os fatores acima os representam em relação à quantidade de unidades de encaminhamento ou processamento simultâneo. A métrica de interesse analisada é o MRT. Esta análise DoE tem como objetivo identificar como tais fatores e suas interações influenciam no tempo médio de resposta (MRT). Os níveis para cada fator são valores 50% a menos e 50% a mais

do valor base, portanto cada fator possui no experimento 2 níveis. A Tabela 9 apresenta os fatores e seus níveis.

Tabela 9 – Tabela de fatores e níveis do DoE.

Fator	**Nível baixo**	**Valor base**	**Nível alto**
c_r	25.0	50.0	75.0
c_a	25.0	50.0	75.0
c_e	50.0	100.0	150.0
c_f	25.0	50.0	75.0
c_c	25.0	50.0	75.0

Como observado na Seção 6.1, o desempenho muda conforme a variação da taxa de chegada. Para esta análise iremos simular a situação de quando o sistema encontra-se mais sobrecarregado, portanto com a maior taxa de chegada (0.1 msg/ms) nas duas entradas (interna e externa). Nas situações onde a taxa de chegada é a maior, há cenários e configurações que podem causar probabilidade de descarte, bem com um aumento acentuado do MRT levando a perca de desempenho. Visto o nível de criticidade das informações em uma possível situação de sobrecarga de requisições no sistema, é importante analisar como as relações entre os componentes (em suas capacidades de processamento ou enfileiramento) podem ajudar a mitigar tais problemas de performance.

A Tabela 10 apresenta os MRTs medidos das 32 execuções referentes a combinação dos 5 fatores em seus 2 diferentes níveis. Como cada fator pode possuir 2 valores (níveis) diferentes, a quantidade de combinações é dada por 2^5.

A Figura 16 apresenta os efeitos de impacto promovido pelos fatores e suas relações entre si. O número de contêineres da cloud (c_c) obteve o maior impacto na análise. A interação entre o número de núcleos da camada de edge (c_e) e o número de contêineres da cloud causa o segundo maior impacto e sua diferença para o fator c_c é consideravelmente grande. Tal diferença pode ser causada pela correlação direta entre o cálculo da métrica MRT e o tempo de processamento de cada contêiner na cloud (LPT).

O terceiro fator de maior efeito é o poder de processamento da camada de edge (c_e), o que pode ser justificado pelo fato desse componente ser o único que recebe requisições de outros dois componentes ao mesmo tempo no modelo. Os demais fatores e interações se equiparam em relação ao impacto sobre o MRT. Tais dados permitem que o avaliador priorize o poder de processamento nas camadas que possam eventualmente desafogar o sistema, neste caso a edge e a cloud.

O conjunto de gráficos da Figura 17 apresenta o comportamento do MRT em relação a interação em dupla entre os fatores e seus níveis de avaliação. A análise leva em consideração o aspecto de melhor desempenho quando o MRT tende a ser menor. A Figura 17(a) mostra a interação entre os níveis de c_r e c_a, os quais representam a capacidade simultânea de roteamento interno e transmissão da antena externa, respectivamente. A

Tabela 10 – Tabela de combinações entre os fatores.

c_r	**c_a**	**c_e**	**c_f**	**c_c**	**MRT (ms)**
25.00	25.00	50.00	25.00	25.00	114941.19
25.00	25.00	50.00	25.00	75.00	16802.61
25.00	25.00	50.00	75.00	25.00	117842.32
25.00	25.00	50.00	75.00	75.00	17071.25
25.00	25.00	150.00	25.00	25.00	118559.18
25.00	25.00	150.00	25.00	75.00	16340.07
25.00	25.00	150.00	75.00	25.00	126146.29
25.00	25.00	150.00	75.00	75.00	16220.81
25.00	75.00	50.00	25.00	25.00	112622.96
25.00	75.00	50.00	25.00	75.00	16726.71
25.00	75.00	50.00	75.00	25.00	115210.40
25.00	75.00	50.00	75.00	75.00	15527.67
25.00	75.00	150.00	25.00	25.00	118877.63
25.00	75.00	150.00	25.00	75.00	16399.57
25.00	75.00	150.00	75.00	25.00	122409.80
25.00	75.00	150.00	75.00	75.00	15554.30
75.00	25.00	50.00	25.00	25.00	107567.64
75.00	25.00	50.00	25.00	75.00	15952.84
75.00	25.00	50.00	75.00	25.00	115601.01
75.00	25.00	50.00	75.00	75.00	15784.27
75.00	25.00	150.00	25.00	25.00	126837.78
75.00	25.00	150.00	25.00	75.00	16295.96
75.00	25.00	150.00	75.00	25.00	124056.60
75.00	25.00	150.00	75.00	75.00	16155.42
75.00	75.00	50.00	25.00	25.00	112885.68
75.00	75.00	50.00	25.00	75.00	16354.59
75.00	75.00	50.00	75.00	25.00	117927.77
75.00	75.00	50.00	75.00	75.00	15541.16
75.00	75.00	150.00	25.00	25.00	123368.24
75.00	75.00	150.00	25.00	75.00	15656.96
75.00	75.00	150.00	75.00	25.00	126528.12
75.00	75.00	150.00	75.00	75.00	15153.52

avaliação sugere que quando o c_a tiver uma capacidade de 75, o c_r deve estar no nível inferior ao número base. Já os dois fatores em níveis iguais (25 ou 75) ao mesmo tempo resultam em um aumento do MRT em mais de 1s.

A Figura 17(b) apresenta o relacionamento dos fatores de c_r e c_e. A capacidade de processamento da edge (c_e) em 50 núcleos resultará em um menor MRT independentemente da capacidade de roteamento (c_r). Caso seja necessário um c_e em 150 núcleos, o desempenho será melhor quando o c_r estiver no nível inferior de 25. No mais, os fatores c_r e c_f apresentados na Figura 17(c) quase não se relacionam. Entretanto, para resultar em um menor MRT a configuração ideal nessa correlação é quando tanto a capacidade de roteamento quanto o número de contêineres de processamento da fog forem de 25. Já com

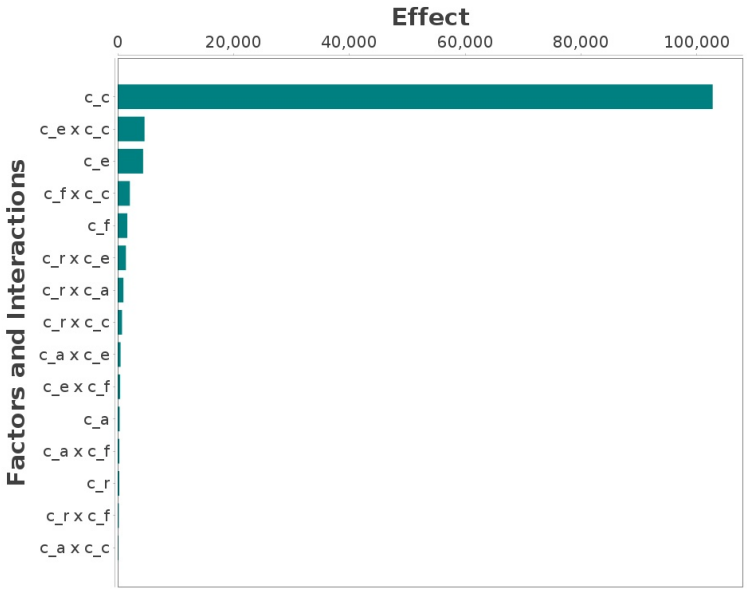


Figura 16 – Fatores, interações e seus respectivos efeitos sobre o MRT.

uma capacidade da fog em 75, o MRT aumenta quase 2s.

Conforme a Figura 17(d), os níveis dos fatores de c_r e c_c não possem correlação de influência. Independente do valor de c_r, o que vai determinar sobre a variação do desempenho do MRT é a quantidade de contêineres da camada de cloud (c_c). Quando a cloud utiliza 75 contêineres de processamento, resulta em um MRT ≈14,2s. Já quando utiliza 25, o MRT aumenta em quase 5 vezes e atinge ≈68s.

A Figura 17(e) apresenta a relação entre a capacidade de transmissão da antena externa (c_a) e a capacidade de processamento da edge (c_e). Uma capacidade na antena de 25 transmissões simultâneas resulta em melhor desempenho quando o c_e possui 25 núcleos. No entanto, quando a c_a permanece em 25 e c_e salta para 75 núcleos, o MRT aumenta ≈1,3s. Ao contrário da interação anterior, a Figura 17(f) mostra que quando a capacidade de transmissão da antena (c_a) está em 25, um menor MRT é atingido quando o número de contêineres da fog (c_f) é de 75. No entanto, o melhor desempenho (≈66,9s) se dá quando c_a é 75 e c_f é 25.

A Figura 17(g) apresenta a interação entre componentes extremos no modelo: a capacidade de transmissão da antena externa (c_a) e o número de contêineres da máquina de cloud (c_c). Ao contrário da ausência de interação do roteador com a cloud em 17(d), é observado uma correlação direta entre os níveis desses fatores e a variação do MRT. Um menor MRT (≈67s) se dá quando a capacidade da antena e a capacidade da cloud encontram-se no mesmo nível (25 ou 75) simultaneamente. Quando os níveis diferem-se entre si o desempenho tende a piorar chegando a ≈68,1s.

A Figura 17(h) mostra a interação entre a c_e e c_f. A análise sugere que, no relacionamento entre esses fatores, o poder de processamento da edge não interfere

significativamente no desempenho quando o número de contêineres da fog (c_f) está em 75. No entanto, quando c_f está em 25 há um melhor desempenho quando o fator c_e está em 50.

As Figuras 17(i) e 17(j) apresentam as interações das camadas edge-cloud e fog-cloud, respectivamente. A máquina de edge com uma configuração de 150 núcleos entrega um MRT de ≈65s quando o número de contêineres da cloud é 75, no entanto quando esse último número cai para 25 o MRT atinge ≈70,6s. A interação fog-cloud é semelhante. A fog com 75 e a cloud com 75 contêineres resulta em um MRT de ≈66,7s, já quando o poder de processamento da cloud cai para 25 o MRT atinge ≈68,5s. Portanto, um acréscimo no poder de processamento da última camada (cloud) requer que nas anteriores também haja esse upgrade para que se tenha uma utilização do sistema de forma mais equilibrada e consequentemente melhora no desempenho.

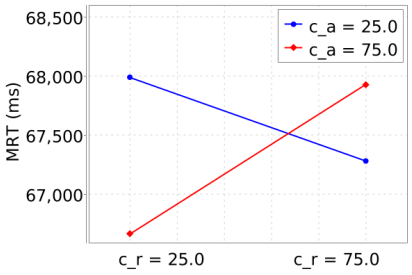
6.2.3 Análise de Performabilidade da Métrica MRT

A Figura 18 apresenta o gráfico de superfície 3D que mostra o comportamento do sistema considerando o desempenho da métrica MRT. Os parâmetros variados são: tempo de falha do contêiner (MTTF_CO) e a capacidade de processamento da *cloud* (C_C). Utilizou-se o MTTF_CO como parâmetro de disponibilidade pelo fato das aplicações JBoss, Banco de Dados e instância Docker estarem bem "rankeadas" na análise de sensibilidade da Seção 6.2.1 - Tabela 8. O parâmetro C_C foi escolhido por ser o fator de maior efeito sobre o desempenho do MRT conforme a análise DoE da Seção 6.2.2 - Figura 16. O MTTF_CO foi variado em quatro partes com intervalos de 1000 horas a menos e 1000 horas a mais e o C_C foi variado com intervalos de 20 contêineres a menos e a mais, conforme Tabela 11. Assim como as demais análises, a taxa de chegada utilizada nas duas fontes de dados foi a maior (0.1 msg/ms).

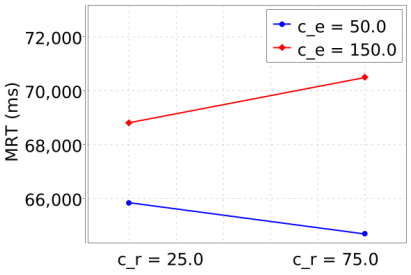
Tabela 11 – Tabela dos parâmetros utilizados na análise de performabilidade.

Parâmetros	**Variação**	**Valor no modelo base**
MTTF_CO (Horas)	516, 1516, 2516, 3516 e 4516	2516
C_C (Quant.)	10, 30, 50, 70 e 90	50

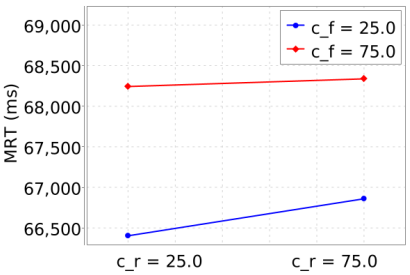
A legenda relaciona os níveis de MRT com algumas cores. A cor roxa representa números de MRT mais baixos e a cor vermelha os números mais altos. Um menor tempo de resposta (≈16s) é encontrado quando o número de contêineres da *cloud* passa de 70. Quando C_C cai para os níveis abaixo de 30, o MRT ultrapassa os 160s e atinge ≈172s com uma capacidade de 10 contêineres. Em ambas as faixas inferior e superior de C_C, o MTTF_CO não influencia no resultado da métrica. Já com o número base de 50 contêineres na *cloud*, a variação do MTTF_CO é inversamente proporcional ao MRT. Com um MTTF_CO = 4516 horas o tempo médio de resposta do sistema fica



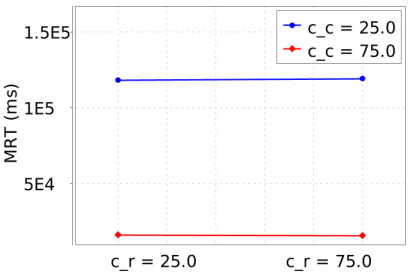
(a) c_r vs. c_a



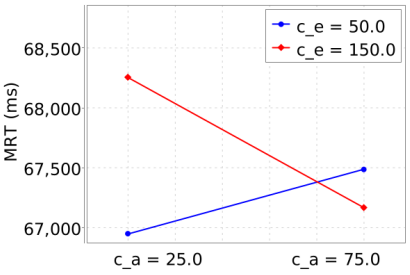
(b) c_r vs. c_e



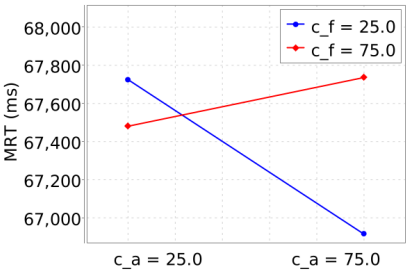
(c) c_r vs. c_f



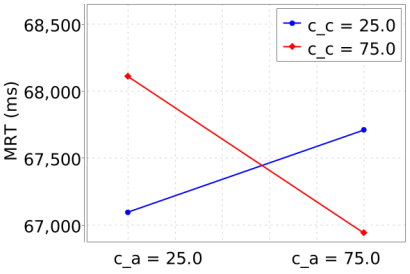
(d) c_r vs. c_c



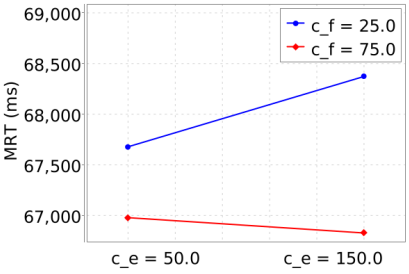
(e) c_a vs. c_e



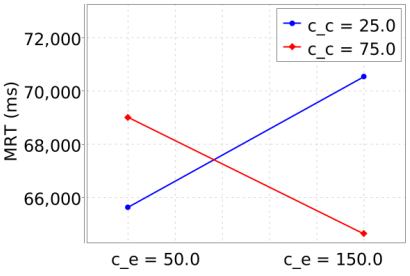
(f) c_a vs. c_f



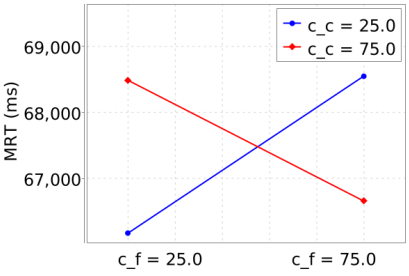
(g) c_a vs. c_c



(h) c_e vs. c_f



(i) c_e vs. c_c



(j) c_f vs. c_c

Figura 17 – Gráficos de interação entre os fatores.

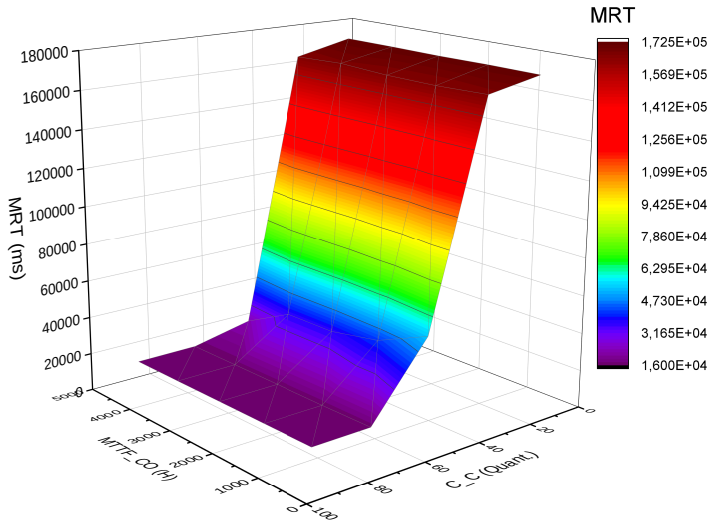


Figura 18 – Análise de performabilidade do MRT em relação a variação do MTTF_CO e C_C.

em ≈23s e com um MTTF_CO = 516 horas o MRT mais que duplica e atinge ≈53s. A Figura 19 evidencia em outro ângulo a influência do MTTF_CO sobre o MRT com um C_C = 50. Portanto, no que se refere ao tempo médio da requisição, considera-se necessário preocupar-se com a disponibilidade dos contêineres enquanto a capacidade de processamento da *cloud* não mude do número base de 50 contêineres.

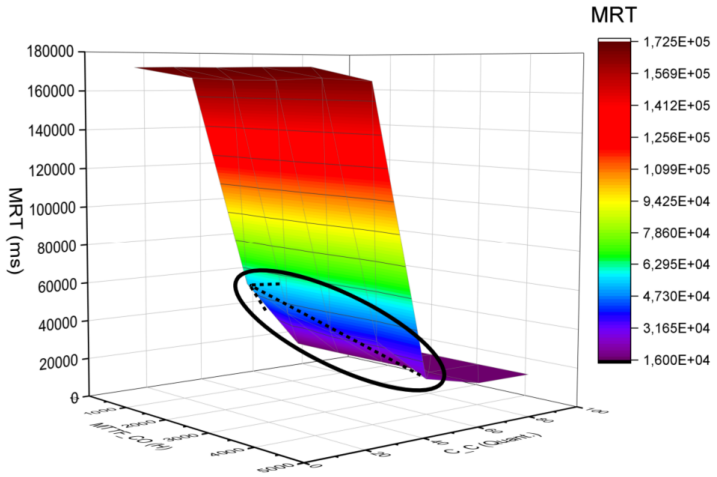


Figura 19 – Variação do ângulo de visão do gráfico da Figura 18.

7 Validação

Este capítulo apresenta a validação do modelo SPN através da comparação entre os seus números de desempenho e os dados extraídos de um experimento de rede real o qual simula os componentes e requisições.

7.1 Processo de Validação

O processo de validação descrito na Figura 20 foi definido nesta dissertação e consiste em uma série de etapas sistemáticas para assegurar que um modelo computacional ou experimental seja validado corretamente:

- **Simplificação do modelo:** O processo se inicia com a criação um novo modelo simplificado o qual é derivado do modelo base. Diminuiu-se o número de componentes replicados que foram apresentados no cenário e no modelo por questões de restrição de recursos computacionais. De forma que esta replicação de recursos computacionais refletem os mesmos parâmetros de quantidade e processamento na parte do modelo proposto para a validação, do ponto de vista estatístico, validar uma parte do modelo valida também o todo.

- **Montagem do experimento conforme o modelo:** Após definição do modelo, um experimento é montado conforme os mesmos parâmetros e número de componentes (Seção 7.2).

- **Sincronização dos relógios:** Utilizando uma ferramenta NTP (MILLS, 1991), as máquinas virtuais do experimento são sincronizadas para que o cálculo do MRT tenha o menor erro possível.

- **Execução do experimento com baixa carga (0 drop):** Neste ponto, o experimento é executado com uma carga mínima, ou seja, uma taxa de chegada baixa ao ponto de cada requisição ter o mínimo tempo de fila possível e assegurando que não haja perda de dados ou eventos (0 drop).

- **Coleta dos tempos intermediários do experimento:** Durante a execução com baixa carga, são coletados os tempos intermediários de cada etapa presente no experimento. Esses tempos são referentes aos "atrasos" na execução do experimento que precisam serem adicionados ao modelo. A grosso modo, a soma desses tempos intermediários refletem o *delay* das requisições no experimento descontando-se todos os tempos de processamento e espera nas filas.

- **Alimentação do modelo SPN com os tempos intermediários do experimento:** Os tempos coletados são utilizados para alimentar o novo modelo, ajustando-o conforme os dados experimentais obtidos. Portanto, o modelo SPN passa agora a conter também os "tempos sujos" do experimento, ou seja, aqueles que não fazem parte do processamento pré-definido na validação e que poderiam influenciar no resultado final da comparação.
- **Execução do experimento com variação de capacidade de processamento:** É definido uma variação entre 1 e 4 instâncias de processamento paralelo para cada máquina. O experimento é executado novamente, agora sob alta carga, e é coletado o Tempo Médio de Resposta (MRT - Mean Response Time) e o Desvio Padrão em cada cenário de poder de processamento.
- **Execução do modelo SPN com os mesmos parâmetros de processamento e capacidade do experimento:** O modelo ajustado é configurado para ser executado sob as mesmas condições de carga (taxa de chegada) e também é coletado o MRT e Desvio Padrão correspondente para cada variação de capacidade de processamento.
- **Execução do T-Test:** Utilizando-se dos dados obtidos, o *p-value* entre os MRTs do modelo e experimento em cada cenário são calculados, afim de que essa confiança estatística seja maior/igual a 0,05. O uso do T-Test na validação estatística entre duas amostras é fundamental para determinar se as diferenças observadas nos dados são significativas ou se podem ser atribuídas ao acaso, garantindo assim a robustez das conclusões tiradas em estudos comparativos.
- **Verificação estatística dos MRTs:** Se os MRTs são estatisticamente iguais, o processo é finalizado com sucesso. Caso contrário, retorna-se à etapa de coleta dos tempos intermediários do experimento para realizar novos ajustes no modelo.

Este ciclo contínuo de ajustes e validações é fundamental para garantir que o modelo se comporte de forma consistente com os resultados experimentais, assegurando sua robustez e precisão antes de ser utilizado em aplicações práticas ou teóricas.

7.2 Modelo, Ambiente do Experimento e Ferramenta de Validação

A Figura 21 apresenta o modelo SPN simplificado. No modelo é abstraído uma fonte de requisições (tokens) e duas máquinas em sequência.

Os valores de parâmetros foram definidos após calibração do modelo a fim de se encontrar resultados que expressam variação significativa do MRT. Os mesmos não refletem especificamente um valor real de capacidade e processamento, visto que foi utilizado um modelo simplificado que atenda as limitações dos recursos computacionais disponíveis

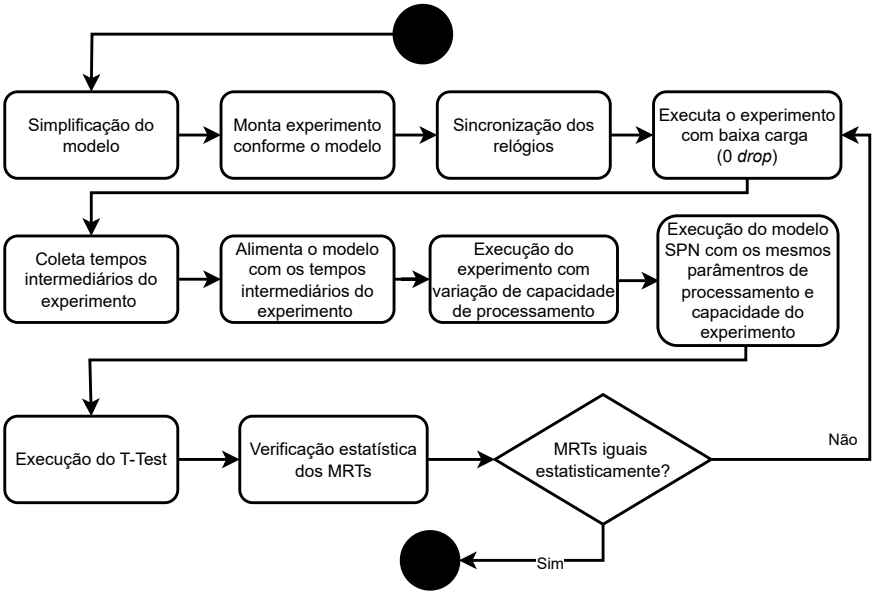


Figura 20 – Diagrama de validação.

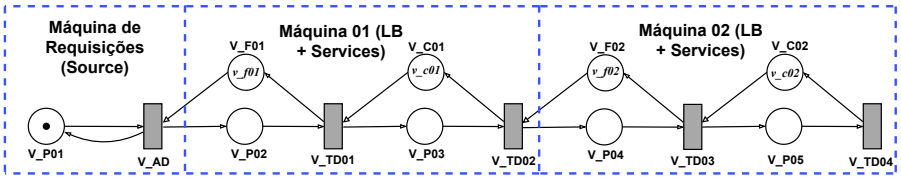


Figura 21 – Modelo SPN da validação.

na validação. É importante ressaltar que foram utilizadas transições determinísticas no modelo avaliado, pois os valores utilizados no experimento são numéricos absolutos. A Tabela 12 os apresenta e, obviamente, são os mesmo valores utilizados no experimento real. É importante frisar que a inclusão dos tempos intermediários é fundamental para sanar a diferença de "ruídos" de processamento do experimento a serem abstraídos junto ao modelo, bem como destacar que tais valores dos tempos intermediários derivam do experimento particular deste trabalho, portanto essa aferição deve ser tratada de forma individual para cada ambiente real configurado. Quando se trata de métricas de tempo, qualquer milissegundo pode ser crucial para afetar no resultado final da validação.

O ambiente real do experimento é composto por 3 máquinas virtuais iguais dentro do mesmo *hipervisor* com as seguintes configurações: Linux Ubuntu (22.04.3 LTS), Intel Xeon octacore 2.30GHz e 16GB de RAM. Uma máquina é responsável pela transmissão das requisições e as demais rodam (cada uma) um LoadBalancer jar e de um a quatro Services jar. A arquitetura montada para o experimento baseou-se em utilizar um ambiente de *hypervisor* unificado afim de minimizar os riscos de incompatibilidade, facilitar o gerenciamento dos recursos e reduzir a latência.

O software validador foi desenvolvido em linguagem Java (versão 8) no laboratório

Tabela 12 – Parâmetros de Transições Temporárias e Capacidades de Fila e Processamento utilizados no modelo do experimento

Parâmetro	**Descrição**	**Valor (ms ou capacidade)**
V_AD	Intervalo de tempo entre as requisições	100 ms
V_TD01	Tempo intermediário coletado no experimento entre a Máquina Source e a Máquina 01	95 ms
V_TD02	Tempo de serviço (processamento) na Máquina 01	1000 ms
V_TD03	Tempo intermediário coletado no experimento entre a Máquina 01 e a Máquina 02	93 ms
V_TD04	Tempo de servico (processamento) na Máquina 02	1000 ms
v_f01	Fila na Máquina 01	20
v_c01	Número de Services Simultâneos na Máquina 01	1 à 4
v_f02	Fila na Máquina 02	20
v_c02	Números de Services Simultâneos na Máquina 02	1 a 4

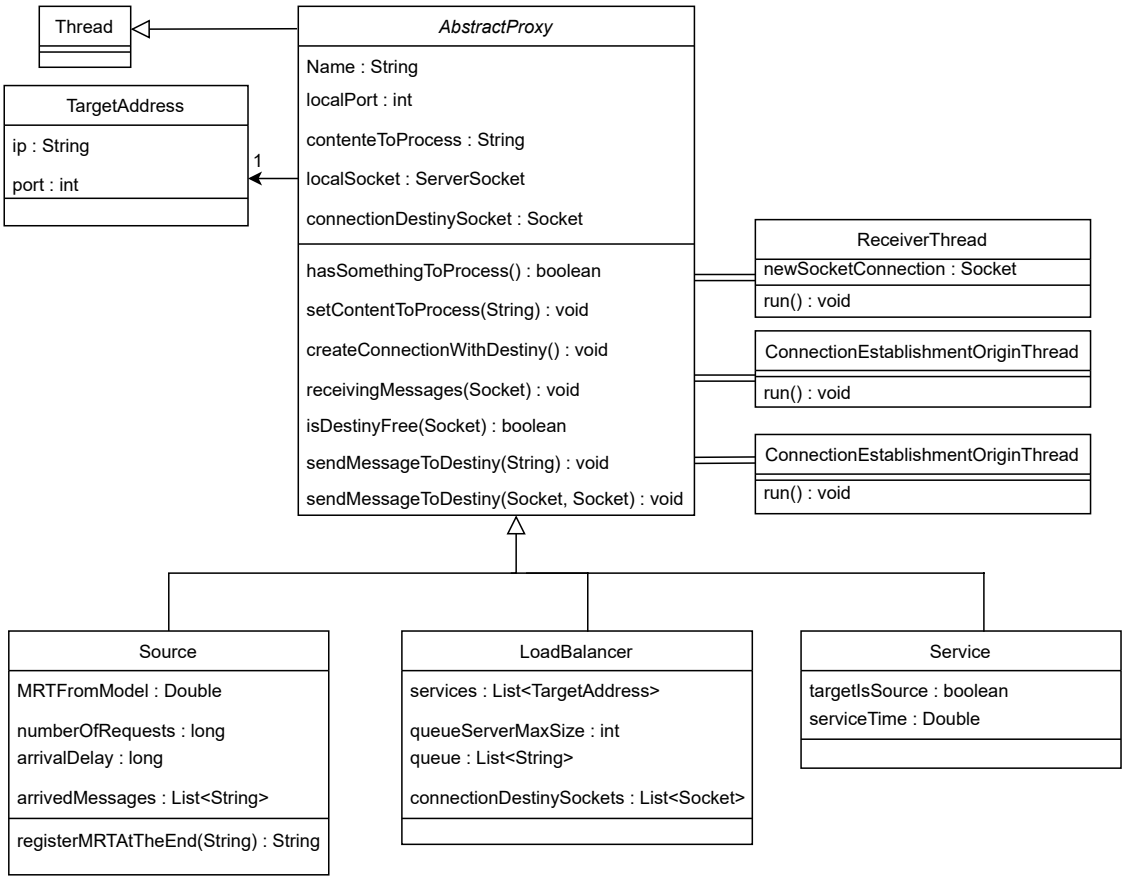


Figura 22 – Diagrama de Classes do Pasid Validator.

Pasid. O projeto possui o nome de Pasid Validator e pode ser adaptável à quaisquer definições de parâmetros de execução. O Pasid Validator (Figura 22) possui 3 aplicações executáveis (*main class*): Source, LoadBalancer e Service. O Source é utilizado para o envio e recebimento final de cada requisição, o LoadBalancer é responsável por receber

as requisições da máquina anterior e realizar o balanceamento de carga (roteamento em fila circular) e o Service realiza o processamento da requisição e a encaminha a próxima máquina. Cada Service jar atua no experimento, em abstração à arquitetura proposta deste trabalho, como um contêiner de processamento. A Figura 22 apresenta o Diagrama de Classes do projeto do Pasid Validator.

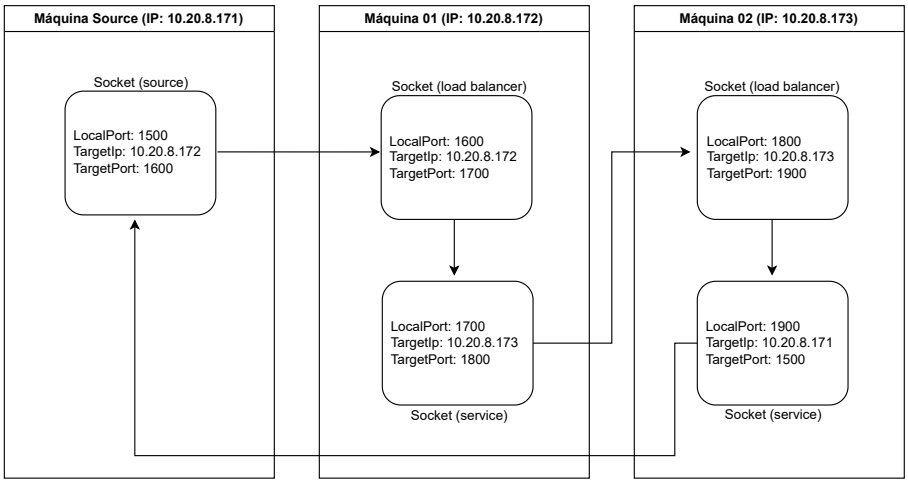


Figura 23 – Comunicação socket no cenário de 1 Service para cada Load Balancer.

O método de transferências de pacotes *Socket* é utilizado na implementação da comunicação entre os jar do sistema distribuído. A Figura 23 representa o fluxo de comunicação de uma requisição no cenário onde há 1 (um) Service para cada máquina. Conforme mostra a Figura, o transporte da requisição é norteado por portas e endereços específicos que são pré configurados antes do experimento. A comunicação socket estabelece uma conexão direta e contínua entre cliente e servidor, facilitando a transferência de informações de forma rápida e confiável.

A Figura 24 ilustra o fluxograma de comunicação entre os componentes definidos no experimento. O caminho da requisição inicia pela Máquina Source, ela é responsável por gerar, em um intervalo de tempo definido, requisições de pacote à Máquina 01. A requisição que encontrar a fila de espera cheia é descartada e as demais continuam o processo. A Máquina 01 recebe cada requisição e através do LoadBalancer a encaminha ao Service escolhido utilizando-se, neste experimento, do algoritmo de fila circular. O Service recebe a requisição e a executa através de *Thread.sleep()* (tempo especificado de pausa em execução), posteriormente a encaminha à Máquina 02. De modo semelhante, a Máquina 02 realiza o reteamento e processamento da requisição e a retorna à Máquina Source para que seja calculado o valor individual do MRT. É importante ressaltar que no corpo de cada requisição é enviado o *time millis* obtido no início da transmissão afim de ser subtraído ao tempo final no retorno para tal cálculo da métrica.

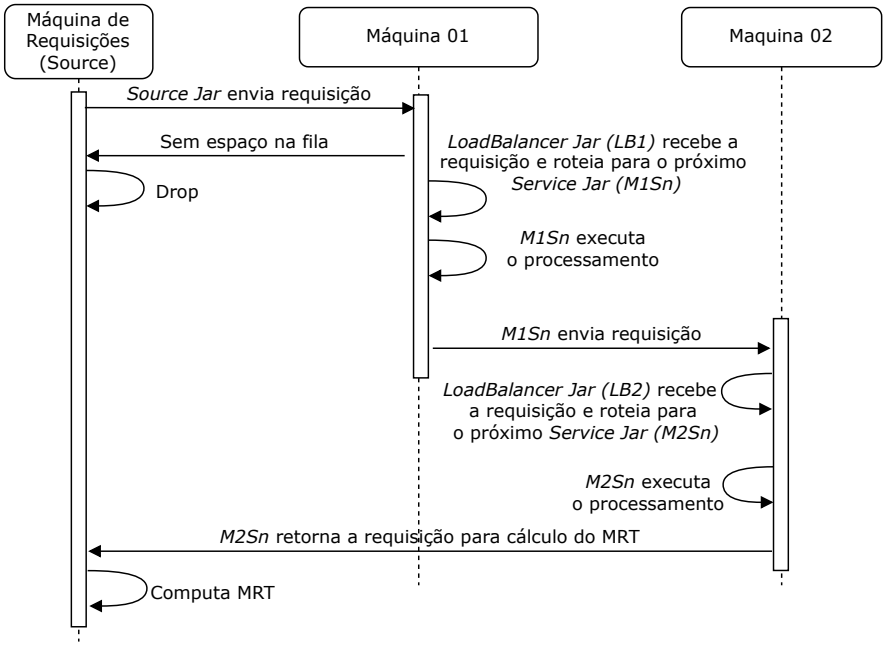


Figura 24 – Diagrama de sequência do experimento.

7.3 Resultados

Os resultados obtidos são oriundos de quarto configurações: 1, 2, 3 e 4 Services de processamento em ambas as máquinas ao mesmo tempo. Para cada configuração, o experimento real foi realizado inicialmente afim de coletar o MRT, Desvio Padrão e tempos intermediários. Antes da execução do modelo SPN (taxa máxima de erro: 5%), os tempos intermediários foram adicionados ao modelo. De posse dos dois MRTs e dos dois Desvios Padrão (Experimento e Modelo) passamos para o passo da análise estatística.

A Figura 25 apresenta o comportamento do experimento e modelo em relação aos dados obtidos. Nas quatro configurações, os valores observados no experimento apresentam uma maior variação em relação ao modelo, como evidenciado pelos desvios padrões mais amplos. No entanto, experimento e modelo seguem uma tendência similar, com MRTs diminuindo à medida que o número de Services aumenta.

Aplicamos o T-Test de uma amostra para comparar o MRT obtido nos experimentos com o MRT gerado pelo modelo. O T-Test é um tipo de teste estatístico usado para comparar as médias de dois grupos, ele pode ser usado quando as amostras satisfazem as condições de normalidade, igual variância e independência (KIM, 2015). Os resultados estatísticos estão elencados na Tabela 13. Para cada configuração, foram enviadas 500 requisições no experimento e por conseguinte o cálculo de 500 MRTs os quais resultaram numa média e no Desvio Padrão. Ainda sobre o experimento, é importante ressaltar q não houve congestionamento a ponto de uma requisição ser dependente da anterior, por isso não precisamos de amostras independentes. O modelo resultou em um valor único da

métrica, bem como o seu Desvio Padrão. Todas as amostras apresentaram distribuição normal.

Para verificar a significância do T-Test, observamos o valor p (*p-value*). O valor p é superior a 0,05 em todos as configurações analisadas. Nestes casos, consideramos que não há diferença significativa entre os dois grupos amostrais, e portanto não podemos refutar a hipótese nula de igualdade nos casos analisados, com 95% de confiança. Por fim, esta validação conclui que os resultados gerados pelo modelo são estatisticamente equivalentes ao experimento. O modelo simplificado reflete o ambiente real, o que também valida o modelo base. Tal resultado ratifica a utilização do modelo SPN para o planejamento da arquitetura proposta deste trabalho, bem como adaptações e reutilizações à critério do avaliador ou arquiteto.

Tabela 13 – Dados Estatísticos do T-Test

Número de Conteineres	Número de Amostras	Média do MRT (ms) - Experimento	Média do MRT (ms) - Modelo	Desvio Padrão - Experimento	Desvio Padrão - Modelo	P-Value
1	500	25101	24253	6561	5122	0,081
2	500	12923	13152	2498	1878	0,116
3	500	9680	9945	1647	3230	0,103
4	500	8290	8078	1302	3329	0,185

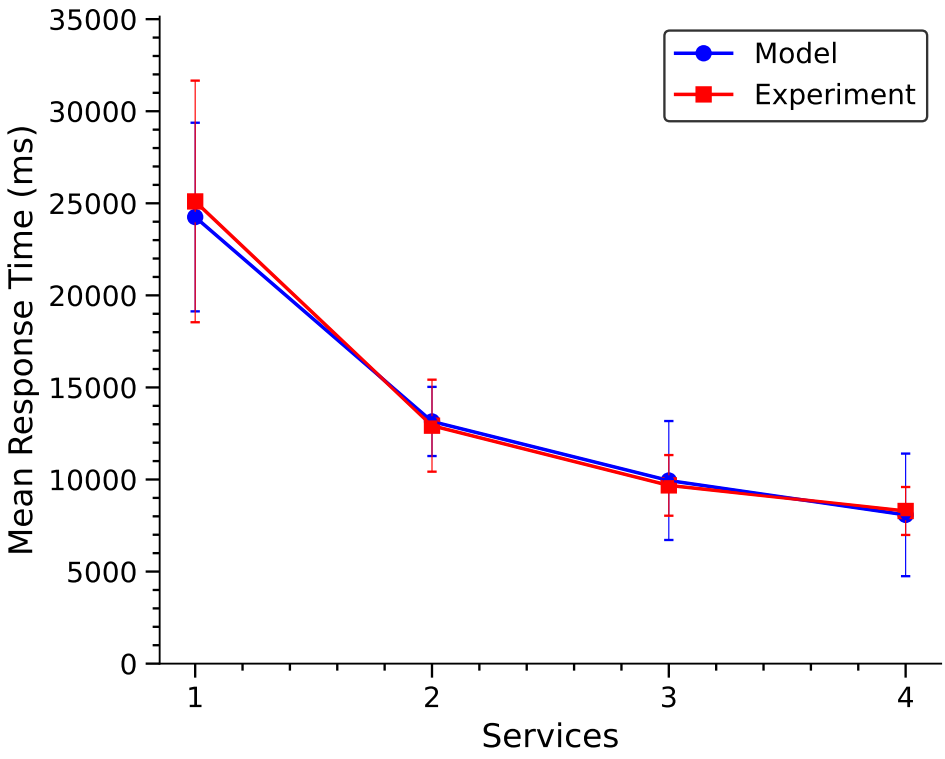


Figura 25 – MRTs obtidos do experimento e modelo em relação à variação de quantidade de Services.

8 Conclusão

Esta dissertação adotou um modelo SPN para avaliar o desempenho de um sistema hospitalar inteligente sem a necessidade de investir previamente em equipamentos reais. Implementação de novas arquiteturas requerem estudos adequados, visto que devemos nos ater a questões de viabilidade e mitigação de investimentos que possam ocorrer de forma desnecessária. O modelo consegue configurar até 32 parâmetros (incluindo tempos de processamento e *delay*, tempos de falha e recuperação e capacidades paralelas de processamento ou armazenamento) e possibilita avaliar o nível de utilização dos recursos computacionais, o tempo médio de resposta e a taxa de perda de dados de uma arquitetura IoHT.

No Cenario 1, a configuração de 25 contêineres na *cloud* se destacou ao entregar dados mais equilibrados em relação ao MRT, fornecendo ao administrador do sistema um maior controle para alocação de recursos quando por ventura o hospital venha a ter uma grande demanda de pacientes de forma repentina. No Cenário 2, a configuração de 25 contêineres na *fog* entregou um menor tempo de resposta e *Drop Rate* em relação ao Cenário 1, bem como uma menor taxa de utilização dos componentes.

Os resultados obtidos com a análise do modelo base devem ser observados como parâmetro no planejamento de dimensionamento dos componentes reais. Portanto, este trabalho pode auxiliar os administradores de sistemas na identificação do equipamento mais adequado em termos de custo e eficiência.

A validação de um modelo analítico é uma etapa fundamental de um trabalho como este. Com ela permite-se ter uma confiança maior que o modelo está consistente e que pode calcular corretamente as métricas de interesse. Na validação apresentada, apesar de ter sido sobre uma simplificação do modelo base, foi possível observar com um certo grau de confiança estatística que o resultado do modelo simplificado é semelhante ao resultado do experimento e portanto validando a utilização do modelo base para os devidos fins propostos.

Este trabalho pode ser útil para projetistas de sistemas que, a priore, planejam atuar em arquiteturas IoHT distribuídas e que tenham a possibilidade de ter duas fontes de dados simultâneas. Com os resultados analisados fica mais fácil a modificação e adaptação dos parâmetros da arquitetura e modelo propostos e que devam refletir a realidade de cada contexto e de cada necessidade.

9 Publicações

ROCHA, Francisco M. S.; SANTOS, Lucas Vinícius S. dos; SILVA, Francisco Airton. Avaliação de Desempenho de uma Arquitetura de Sistema de Monitoramento de Pacientes em Hospitais Inteligentes com Fontes de Dados Interna e Externa. In: **WORKSHOP EM DESEMPENHO DE SISTEMAS COMPUTACIONAIS E DE COMUNICAÇÃO (WPERFORMANCE)**, 22. , 2023, João Pessoa/PB. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2023 . p. 73-84. ISSN 2595-6167. DOI: https://doi.org/10.5753/wperformance.2023.230079.

Journal of Network and Systems Management (JNSM): Submetido e aguardando resposta após revisão.

Referências

AL-DHURAIBI, Y. et al. Elasticity in cloud computing: state of the art and research challenges. *IEEE Transactions on Services Computing*, IEEE, v. 11, n. 2, p. 430–447, 2017. 8

ANDRADE, E.; NOGUEIRA, B. Dependability evaluation of a disaster recovery solution for iot infrastructures. *The Journal of Supercomputing*, Springer, v. 76, n. 3, p. 1828–1849, 2020. 4

ANDRADE, E.; NOGUEIRA, B. Dependability evaluation of a disaster recovery solution for iot infrastructures. *The Journal of Supercomputing*, Springer, v. 76, n. 3, p. 1828–1849, 2020. 21, 22

ARAÚJO, C. J. M. Tomada de decisão multicritério em infraestruturas como serviço em nuvem: uma abordagem baseada em modelos de dependabilidade, performabilidade e custo. Universidade Federal de Pernambuco, 2019. 12

ARAUJO, J. et al. Dependability evaluation of a mhealth system using a mobile cloud infrastructure. In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2014. p. 1348–1353. 21, 22

ASSUNCAO, M. D. de; VEITH, A. da S.; BUYYA, R. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. *Journal of Network and Computer Applications*, Elsevier, v. 103, p. 1–17, 2018. 7

AVIZIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, IEEE, v. 1, n. 1, p. 11–33, 2004. 12

BONOMI, F. et al. Fog computing and its role in the internet of things. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.: s.n.], 2012. p. 13–16. 3

BRITO, C. et al. Stochastic model driven performance and availability planning for a mobile edge computing system. *Applied Sciences*, MDPI, v. 11, n. 9, p. 4088, 2021. 32

CARVALHO, D. et al. Mobile edge computing performance evaluation using stochastic petri nets. In: IEEE. *2020 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2020. p. 1–6. 3

CHAE, M.; LEE, H.; LEE, K. A performance comparison of linux containers and virtual machines using docker and kvm. *Cluster Computing*, Springer, v. 22, n. Suppl 1, p. 1765–1775, 2019. 8

CHEN, X. et al. Performance evaluation of scheduling policies in a smart hospital environment. In: *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. [S.l.: s.n.], 2011. p. 585–592. 19, 22

COSTA, I. et al. Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform. *Quality and Reliability Engineering International*, Wiley Online Library, v. 32, n. 7, p. 2191–2205, 2016. 32

FEITOSA, L. et al. Performance evaluation of message routing strategies in the internet of robotic things using the d/m/c/k/fcfs queuing network. *Electronics*, MDPI, v. 10, n. 21, p. 2626, 2021. 44

GOMEZ-SACRISTAN, A. et al. Evaluation of quality of service in smart-hospital communications. *Journal of Medical Imaging and Health Informatics*, American Scientific Publishers, v. 5, n. 8, p. 1864–1869, 2015. 19, 22

HAVERKORT, B. R.; NIEMEGEERS, I. G. Performability modelling tools and techniques. *Performance evaluation*, Elsevier, v. 25, n. 1, p. 17–40, 1996. 12

HELALI, L.; OMRI, M. N. A survey of data center consolidation in cloud computing systems. *Computer Science Review*, Elsevier, v. 39, p. 100366, 2021. 8

HUANG, D. Y. et al. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, ACM New York, NY, USA, v. 4, n. 2, p. 1–21, 2020. 3

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley New York, 1991. v. 1. 13, 14, 15, 16, 17, 18

KAFHALI, S. E.; SALAH, K. Performance modelling and analysis of internet of things enabled healthcare monitoring systems. *IET Networks*, Wiley Online Library, v. 8, n. 1, p. 48–58, 2019. 20, 22

KAUR, M.; MALHOTRA, J.; KAUR, P. D. A vanet-iot based accident detection and management system for the emergency rescue services in a smart city. In: IEEE. *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. [S.l.], 2020. p. 964–968. 3

KHAYAL, I. S.; FARID, A. M. A dynamic system model for personalized healthcare delivery and managed individual health outcomes. *IEEE Access*, v. 9, p. 138267–138282, 2021. 20, 22

KIM, T. K. T test as a parametric statistic. *Korean journal of anesthesiology*, Korean Society of Anesthesiologists, v. 68, n. 6, p. 540, 2015. 56

KLEIJNEN, J. P. Sensitivity analysis and optimization in simulation: design of experiments and case studies. In: IEEE. *Winter Simulation Conference Proceedings, 1995.* [S.l.], 1995. p. 133–140. 10

LIMA, J. W. S. d.; CALLOU, G.; ANDRADE, E. Queueing theory and stochastic petri net: A tutorial. *Research, Society and Development*, v. 10, n. 3, p. e2810312826, Mar. 2021. Disponível em: <https://rsdjournal.org/index.php/rsd/article/view/12826>. 8

LISBOA, M. F. F. da S. et al. Modeling the availability of an e-health system integrated with edge, fog and cloud infrastructures. In: IEEE. *2018 IEEE symposium on computers and communications (ISCC)*. [S.l.], 2018. p. 00416–00421. 20, 22

LITTLE, J. D. A proof for the queuing formula: L= λ w. *Operations research*, INFORMS, v. 9, n. 3, p. 383–387, 1961. 34

MACIEL, P. et al. Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: IEEE. *2017 IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)*. [S.l.], 2017. p. 50–57. 10, 21, 29

MARSAN, M. A. et al. Modelling with generalized stochastic petri nets. *ACM SIGMETRICS performance evaluation review*, ACM New York, NY, USA, v. 26, n. 2, p. 2, 1998. 9

MATOS, R. et al. Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory*, Elsevier, v. 50, p. 151–164, 2015. 43

MAVRIDIS, I.; KARATZA, H. Performance and overhead study of containers running on top of virtual machines. In: IEEE. *2017 IEEE 19th Conference on Business Informatics (CBI)*. [S.l.], 2017. v. 2, p. 32–38. 8

MERKEL, D. et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, v. 239, n. 2, p. 2, 2014. 8

MEYER. On evaluating the performability of degradable computing systems. *IEEE Transactions on computers*, IEEE, v. 100, n. 8, p. 720–731, 1980. 12

MILLS, D. L. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, Ieee, v. 39, n. 10, p. 1482–1493, 1991. 51

MISRA, K. B. et al. *Handbook of performability engineering*. [S.l.]: Springer, 2008. v. 1256. 12

MONDESIRE, S. C. et al. Combining virtualization and containerization to support interactive games and simulations on the cloud. *Simulation Modelling Practice and Theory*, Elsevier, v. 93, p. 233–244, 2019. 8

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. 8

NASCIMENTO, M. B. Cluster de alto desempenho para uso na disciplina de computação paralela e distribuída utilizando contêineres. 2019. 8

NDIAYE, M. et al. Iot in the wake of covid-19: A survey on contributions, challenges and evolution. *IEEE Access*, IEEE, v. 8, p. 186821–186839, 2020. 3

NGUYEN, T. A. et al. Performability evaluation of load balancing and fail-over strategies for medical information systems with edge/fog computing using stochastic reward nets. *Sensors*, MDPI, v. 21, n. 18, p. 6253, 2021. 4

NGUYEN, T. A. et al. Performability evaluation of load balancing and fail-over strategies for medical information systems with edge/fog computing using stochastic reward nets. *Sensors*, MDPI, v. 21, n. 18, p. 6253, 2021. 19, 22

NGUYEN, T. A. et al. Dependability and security quantification of an internet of medical things infrastructure based on cloud-fog-edge continuum for healthcare monitoring using hierarchical models. *IEEE Internet of Things Journal*, v. 8, n. 21, p. 15704–15748, 2021. 20, 22

PAHL, C. Containerization and the paas cloud. *IEEE Cloud Computing*, IEEE, v. 2, n. 3, p. 24–31, 2015. 8

RAJASEKARAN, M. et al. Autonomous monitoring in healthcare environment: Reward-based energy charging mechanism for iomt wireless sensing nodes. *Future Generation Computer Systems*, Elsevier, v. 98, p. 565–576, 2019. 3

RODRIGUES, L. et al. Stochastic model for evaluating smart hospitals performance. In: *2019 IEEE Latin-American Conference on Communications (LATINCOM)*. [S.l.: s.n.], 2019. p. 1–6. 19, 22

RODRIGUES, L. et al. Modelo estocástico para avaliaçao de disponibilidade de hospitais inteligentes. In: SBC. *Anais do XIX Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. [S.l.], 2020. p. 145–156. 19, 22

RODRIGUES, L. et al. Performance and availability evaluation of an smart hospital architecture. *Computing*, Springer, v. 103, n. 10, p. 2401–2435, 2021. 4, 33, 34

RODRIGUES, L. et al. Performance and availability evaluation of an smart hospital architecture. *Computing*, Springer, v. 103, p. 2401–2435, 2021. 19, 22

SANTNER, T. J. et al. *The design and analysis of computer experiments*. [S.l.]: Springer, 2003. v. 1. 13, 14

SANTOS, G. L. et al. Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures. *Journal of Cloud Computing*, SpringerOpen, v. 7, n. 1, p. 1–22, 2018. 20, 22

SANTOS, G. L. et al. The internet of things for healthcare: Optimising e-health system availability in the fog and cloud. *International Journal of Computational Science and Engineering*, Inderscience Publishers (IEL), v. 21, n. 4, p. 615–628, 2020. 20, 22

SANTOS, G. L. et al. Maximising the availability of an internet of medical things system using surrogate models and nature-inspired approaches. *International Journal of Grid and Utility Computing*, Inderscience Publishers (IEL), v. 13, n. 2-3, p. 291–308, 2022. 20, 22

SANTOS, G. L. et al. Maximising the availability of an internet of medical things system using surrogate models and nature-inspired approaches. *International Journal of Grid and Utility Computing*, Inderscience Publishers (IEL), v. 13, n. 2-3, p. 291–308, 2022. 32

SANTOS, L. et al. Data processing on edge and cloud: a performability evaluation and sensitivity analysis. *Journal of Network and Systems Management*, Springer, v. 29, n. 3, p. 1–24, 2021. 10, 44

SILVA, F. A. et al. Model-driven impact quantification of energy resource redundancy and server rejuvenation on the dependability of medical sensor networks in smart hospitals. *Sensors*, MDPI, v. 22, n. 4, p. 1595, 2022. 43

SILVA, F. A. et al. Supporting availability evaluation of a smart building monitoring system aided by fog computing. *Electronics Letters*, Wiley Online Library, v. 58, n. 12, p. 471–473, 2022. 43

SILVA, F. A.; FÉ, I.; GONÇALVES, G. Stochastic models for performance and cost analysis of a hybrid cloud and fog architecture. *JOURNAL OF SUPERCOMPUTING*, Springer, 2020. 3

SILVA, F. A. et al. Performance evaluation of an internet of healthcare things for medical monitoring using m/m/c/k queuing models. *IEEE Access*, IEEE, v. 9, p. 55271–55283, 2021. 5

SILVA, F. A. et al. Performance evaluation of an internet of healthcare things for medical monitoring using m/m/c/k queuing models. *IEEE Access*, v. 9, p. 55271–55283, 2021. 20, 22

SINGH, R. P. et al. Internet of things (iot) applications to fight against covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, Elsevier, 2020. 3

SOUZA, H. G. T. d. et al. Análise de disponibilidade de sistemas de mhealth fundamentados na internet das coisas. Universidade Federal Rural de Pernambuco, 2017. 12

STRIELKINA, A. et al. Availability models for healthcare iot systems: Classification and research considering attacks on vulnerabilities. In: IEEE. *2018 IEEE 9th international conference on dependable systems, services and technologies (DESSERT)*. [S.l.], 2018. p. 58–62. 20, 22

TANG, S.; XIE, Y. Availability modeling and performance improving of a healthcare internet of things (iot) system. *IoT*, MDPI, v. 2, n. 2, p. 310–325, 2021. 20, 22

USLU, B. Ç.; OKAY, E.; DURSUN, E. Analysis of factors affecting iot-based smart hospital design. *Journal of Cloud Computing*, SpringerOpen, v. 9, n. 1, p. 1–23, 2020. 3, 4

VALENTIM, T. et al. Availability assessment of internet of medical things architecture using private cloud. In: SBC. *Anais do L Seminário Integrado de Software e Hardware*. [S.l.], 2023. p. 13–23. 20, 22

WATADA, J. et al. Emerging trends, techniques and open issues of containerization: A review. *IEEE Access*, IEEE, v. 7, p. 152443–152472, 2019. 8

YASSEIN, M. B. et al. Iot-based healthcare systems: a survey. In: *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems*. [S.l.: s.n.], 2019. p. 1–9. 3