Daniel de Carvalho Borges
Orientador: Prof. Dr. Francisco Airton Pereira da Silva

Alocação de Servidores em Computação na Borda Móvel Usando Redes de Petri Estocásticas

Picos - PI
22 de novembro de 2019

Daniel de Carvalho Borges
Orientador: Prof. Dr. Francisco Airton Pereira da Silva

Alocação de Servidores em Computação na Borda Móvel Usando Redes de Petri Estocásticas

Monografia submetida ao Curso de Bacharelado em Sistemas de Informação como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Universidade Federal do Piauí
Campus Senador Heuvídio Nunes de Barros
Bacharelado em Sistemas de Informação

Picos - PI
22 de novembro de 2019

Agradecimentos

Primeiramente quero agradecer a Deus por nunca ter me deixado desistir e me concedido saúde e forças para superar as dificuldades e me tornar alguém mais digno.

Agradeço aos meus pais, José Wilson e Francilene, por todo o apoio que sempre me deram e por todo o esforço que fizeram para que eu pudesse chegar onde estou hoje, sempre serei grato por isso e um dia eu os recompensarei.

À toda a minha família que sempre torceram por mim e nunca deixaram faltar apoio e por todos os conselhos que me deram sempre que eu precisei. Quero agradecer especialmente aos meus avós, Otacílio e Lourdes, que tiveram um papel especial em relação aos demais.

Ao professor Dr. Francisco Airton, pela orientação, apoio e confiança, sem ele nada disso seria possível. Também agradeço à todos os outros professores da UFPI que foram importantes não só construção do meu conhecimento, como também foram fontes de inspiração e valores que carregarei para sempre.

Aos meus amigos Caio, Renan, Antônia, Maria, Firmino, Laécio, Max, Raiza e João Henrique por estarem presentes no meu dia a dia durante todo o curso, e a Milene, Marciana, João Victor e Gabriel por mesmo chegando na reta final, também sempre foram presentes e sempre estiveram ao meu lado quando eu mais precisei.

Quero agradecer especialmente à Vitória que, apesar de tudo, sempre esteve ao meu lado me dando todo o apoio que precisei e por todas as palavras de carinho que levarei comigo a vida toda.

À todos que fizeram parte, de alguma forma, da minha formação, o meu mais sincero muito obrigado.

O conhecimento lhe dará a oportunidade de fazer a diferença.

Claire Fagin

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Campus Senador Helvídeo Nunes de Barros
Biblioteca Setorial José Albano de Macêdo
Serviço de Processamento Técnico

B732a Borges, Daniel de Carvalho.
Alocação de servidores em computação na borda móvel usando redes de petri estocásticas. / Daniel de Carvalho Borges. -- Picos,PI, 2019.
47 f.
CD-ROM: 4 ¾ pol.

Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). – Universidade Federal do Piauí, Picos, 2020.
"Orientador(A): Prof. Dr. Francisco Airton Pereira da Silva."

1. Internet das Coisas. 2. Computação Móvel de Borda, 3. Alocação de Servidores. I. Título.

CDD 004.22

Elaborada por Rafael Gomes de Sousa CRB 3/1163

ALOCAÇÃO DE SERVIDORES EM COMPUTAÇÃO NA BORDA MÓVEL USANDO REDES DE PETRI ESTOCÁSTICAS

DANIEL DE CARVALHO BORGES

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 02 de dezembro de 2019

Prof. Francisco Airton Pereira da Silva

Profa. Juliana Oliveira de Carvalho

Prof. Iure de Sousa Fé

Resumo

O Mobile Edge Computing (MEC) é uma arquitetura de rede que tira proveito dos recursos de computação em nuvem (como alta disponibilidade e elasticidade) e utiliza recursos computacionais disponíveis na borda da rede para aprimorar a experiência do usuário móvel, diminuindo o serviço latência. Para atingir esse objetivo, as soluções MEC precisam alocar dinamicamente as solicitações o mais próximo possível de seus usuários. No entanto, o posicionamento da solicitação não depende apenas da localização geográfica dos servidores, mas também de suas configurações. Com base nesse fato, este trabalho propõe um modelo de Rede de Petri Estocástica (SPN) para representar um cenário do MEC e analisar seu desempenho, focando nos parâmetros que podem impactar diretamente o Tempo Médio de Resposta (MRT) e o Nível de Utilização. Para apresentar a aplicabilidade do nosso trabalho, apresentamos três estudos de caso com análises numéricas usando valores reais. O objetivo principal é fornecer um guia prático para ajudar os administradores de infraestrutura de computadores a adaptar suas arquiteturas, encontrando uma troca entre MRT e nível de uso.

Palavras-chaves: Computação Móvel de Borda, Internet das Coisas, Modelos Estocásticos, Alocação de Servidores.

Abstract

Mobile Edge Computing (MEC) is a network architecture that takes advantage of cloud computing capabilities (such as high availability and elasticity) and makes use of computational resources available at the edge of the network in order to enhance the mobile user experience by decreasing the service latency. To achieve this goal, MEC solutions need to dynamically allocate the requests as close as possible of their users. However, the request placement does not depend only on the geographical location of the servers, but also on their configurations. Based on this fact, this paper proposes a Stochastic Petri Net (SPN) model to represent a MEC scenario and analyze its performance, focusing on the parameters that can directly impact the Mean Response Time (MRT) and Utilization Level. In order to present the applicability of our work, we present three case studies with numerical analyzes using real values. The main objective is to provide a practical guide to help computer infrastructure administrators to adapt their architectures, finding a trade-off between MRT and level of use.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

MCC	Mobile Cloud Computing (Computação Móvel na Nuvem)
MEC	Mobile Edge Computing (Computação Móvel na Borda)
MRT	Medium Response Time (Tempo Médio de Resposta)
SPN	Stochastic Petri Net (Rede de Petri Estocástica)

Sumário

1 Introdução

De acordo com um relatório da CISCO, o número de dispositivos móveis chegará a cerca de 11,6 bilhões até 2020 (JUNG, 2016). Essa grande adoção é fundamentalmente impulsionada pelo aumento do número de novos usuários móveis, bem como pelo desenvolvimento de aplicativos interativos (KITANOV; MONTEIRO; JANEVSKI, 2016) (BECK et al., 2014). No entanto, o crescente número de dispositivos móveis tem um efeito extremo sobre as redes móveis, trazendo desafios significativos para as empresas de telecomunicações (CAU et al., 2016). Os dispositivos móveis têm baixa capacidade de armazenamento e as redes celulares são caracterizadas por alto consumo de energia, baixa largura de banda e alta latência (ORSINI; BADE; LAMERSDORF, 2015). Além disso, o crescimento exponencial da Internet das Coisas (IoT) promete tornar as redes sem fio ainda mais difíceis (BORGIA et al., 2016). Para lidar com esses desafios, muitos pesquisadores propuseram arquiteturas otimizadas para esse contexto, como o Mobile Cloud Computing (MCC) (MAROTTA et al., 2015; Sucipto et al., 2017).

A MCC é a integração da computação em nuvem e da computação móvel, que fornece recursos adicionais para dispositivos móveis ao centralizar seus recursos na infraestrutura de nuvem (DINH et al., 2013). No entanto, com vários dispositivos móveis, a MCC enfrenta desafios notáveis, como alta latência, vulnerabilidade de segurança e transmissão de dados limitada, porque os recursos da nuvem estão frequentemente longe dos usuários finais (JARARWEH et al., 2016a).

Como uma evolução da arquitetura do MCC, surgiu o Mobile Edge Computing (MEC). O principal objetivo do MEC é abordar os desafios que a MCC vem enfrentando, implantando recursos ainda mais perto dos usuários finais, ou seja, na borda da rede, permitindo que as operações de computação e armazenamento sejam realizadas mais perto do dispositivo de origem (JARARWEH et al., 2016b). O conceito do MEC surgiu com o objetivo de unir as operadoras de telecomunicações, TI e empresas de computação em nuvem para fornecer serviços em nuvem diretamente na borda da rede. Diferentemente dos sistemas de computação em nuvem tradicionais em que nuvens públicas remotas são utilizadas, os servidores MEC são de propriedade da operadora de rede e são implementados diretamente nas Estações Base (BSs) celulares ou nos Pontos de Acesso sem fio (APs) locais usando uma plataforma de computação genérica. Com essa posição, o MEC permite a execução de aplicativos nas proximidades dos usuários finais, reduzindo substancialmente o atraso de ponta a ponta e liberando a carga sobre as redes de *backhaul*. Como o MEC ainda é considerado um tópico recente, algumas oportunidades de pesquisa estão abertas para serem exploradas, como avaliação de desempenho de arquiteturas MEC (Tran et al., 2017). Outro aspecto está relacionado ao posicionamento dos servidores de borda no campo MEC. Pouca atenção foi dada ao efeito de descarregar as cargas de trabalho

de usuários móveis para servidores de borda e a colocação de servidores de borda no desempenho de aplicativos móveis.

Modelos analíticos são adequados para avaliar o desempenho de sistemas complexos, principalmente durante os estágios iniciais. A modelagem estocástica é um método formal adequado e amplamente utilizado para avaliação de desempenho em sistemas concorrentes com mecanismos de sincronização e comunicação. As Redes de Petro Estocásticas (SPNs) são casos especiais de modelos estocásticos (DESROCHERS; AL-JAAR; SOCIETY, 1995), que são capazes de configurar equações do estado, equações algébricas e outros modelos matemáticos que governam o comportamento dos sistemas. O uso de SPNs já foi aplicado com sucesso no contexto da MCC em trabalhos anteriores (SILVA et al., 2018; PINHEIRO et al., 2018b; PINHEIRO et al., 2018a). No entanto, de acordo com o nosso conhecimento, os modelos analíticos de desempenho nas arquiteturas do MEC são escassos até o momento presente.

Neste documento, nos concentramos na colocação do servidor MEC em um ambiente de computação de borda móvel que fornece cobertura de Internet sem fio para usuários móveis em uma área metropolitana de grande escala. Para isso, propomos um modelo SPN para representar a arquitetura MEC e avaliar o *trade-off* entre *tempo médio de resposta (MRT)* e *utilização de recursos*. O *MRT* significa o tempo para uma solicitação ser processada por servidores MEC e retornar para dispositivos clientes móveis. A *utilização de recursos*, como o nome indica, significa a porcentagem de recursos que estão no modo ativo. Em outras palavras, a utilização de recursos está relacionada ao número de computadores (máquina / VMs / *containeres* físicos) que estão sendo usados.

Em resumo, as principais contribuições deste trabalho são:

- apresentamos um modelo analítico, que serve como uma ferramenta útil para administradores de sistemas avaliarem o desempenho de diferentes arquiteturas MEC, especialmente durante a fase de planejamento antes da implementação;
- nós fornecemos um conjunto de análises numéricas com dados reais que servem como um guia prático para a análise de desempenho em arquiteturas MEC.

1.1 Definição do Problema

A capacidade dos servidores e a forma em que estão distribuídos podem influenciar diretamente no desempenho da arquitetura MEC. No entanto, avaliações de desempenho aprofundadas nesse aspecto possuem alta complexidade e alto custo financeiro.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é avaliar o desempenho de uma arquitetura MEC sob a perspectiva de planejamento de arquiteturas mais eficientes, e auxiliar na alocação dos recursos afim de obter o melhor desempenho possível.

1.2.2 Objetivos Específicos

1. Diminuir o tempo médio de resposta.
2. Reduzir o nível de utilização de recursos, evitando ociosidade.

1.3 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: O capítulo 2 apresenta os conceitos necessários para entender a proposta; O capítulo 3 discute os principais trabalhos relacionados; O capítulo 4 apresenta a arquitetura do MEC que foi usada como base para a construção do modelo; no capítulo 5, apresentamos nosso modelo SPN básico com as respectivas métricas e estudos de caso; no capítulo 6, apresentamos um modelo SPN estendido também com as respectivas métricas e estudos de caso; O capítulo 7 apresenta a validação do modelo SPN proposto com o objetivo de comparar os resultados obtidos no modelo e no experimento real, e o capítulo 8 traça algumas conclusões e trabalhos futuros.

2 Background

Este capítulo apresenta os principais conceitos necessários para melhor acompanhar a proposta deste trabalho. Primeiro, apresentamos os aspectos gerais dos SPNs e, em seguida, algumas características da arquitetura do MEC são detalhadas.

2.1 Redes de Petri Estocásticas

Petri Nets (PNs) são gráficos matemáticos de modelagem aplicável a muitos sistemas e útil para descrever e estudar sistemas de processamento de informações que são caracterizados como sendo concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e estocásticos. As redes de Petri Estocásticas (SPNs) são casos especiais de PNs. Os modelos SPN foram inicialmente projetados como uma ferramenta que permitia a integração da descrição formal, prova de correção e avaliação de desempenho. A Figura 1 apresenta os componentes usados para modelar um SPN.

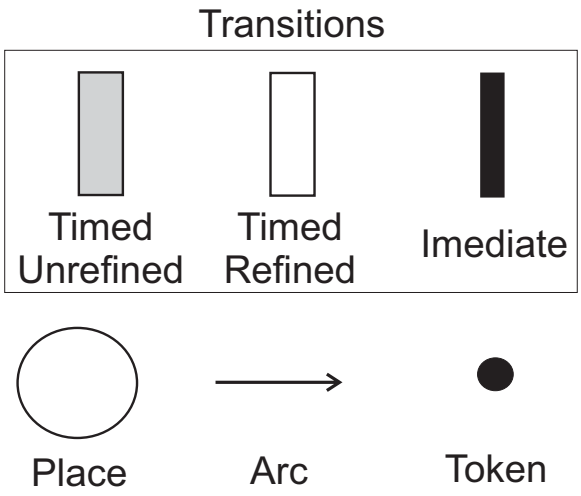

Figura 1: Componentes SPN.

A Figura 2 apresenta um modelo básico de SPN (OLIVEIRA et al., 2017) para avaliação de desempenho adotado neste trabalho. A transição “generate” gera tokens que correspondem a solicitações de serviço. Para cada token gerado colocado no lugar “generated’, é feita uma escolha. Um token pode ser enfileirado para ser processado pelo servidor se houver uma posição livre na fila. Se a fila estiver cheia, no entanto, o token será descartado. O lugar "free"corresponde aos recursos disponíveis (nós de serviço). Mais especificamente, o número de recursos disponíveis é marcado com k. Os tokens que esperam pelo serviço são colocados no local "queue". O serviço de transição representa o processamento dos pedidos enfileirados pelo servidor.

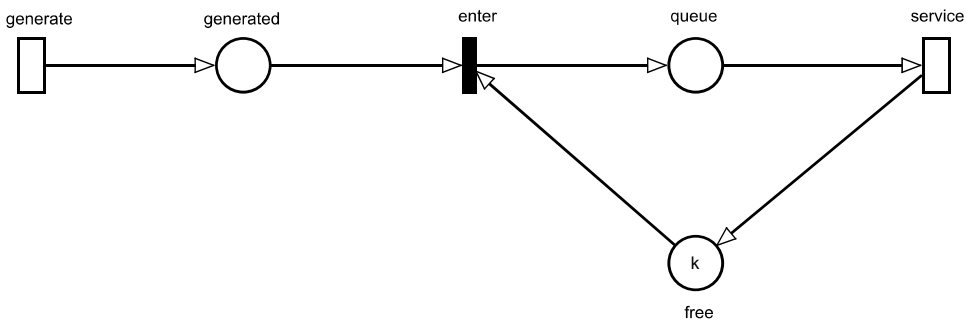

Figura 2: Exemplo de Modelo SPN.

Agora, vamos considerar um exemplo, ilustrado na Figura 3. Bob, um administrador do sistema, tem hoje um servidor executando um serviço da Web específico para seus usuários. O servidor hoje não é suficiente para atender a demanda, e ele quer comprar mais servidores, mas ao mesmo tempo evitando a ociosidade. *Como preditar quantos servidores Bob precisa usar no modelo básico SPN da Figura 2?*. Basicamente Bob precisa inserir três informações: (i) gerar transição - intervalo de tempo médio entre a chegada das solicitações atuais. (ii) transição de serviço - tempo médio necessário para processar apenas uma solicitação. (iii) marcação k - quantos servidores Bob está disposto a comprar. Depois disso, através de simulação ou análise, resolvemos o modelo SPN para calcular o MRT e a utilização de recursos. Essas métricas são detalhadas na seção 5.1.

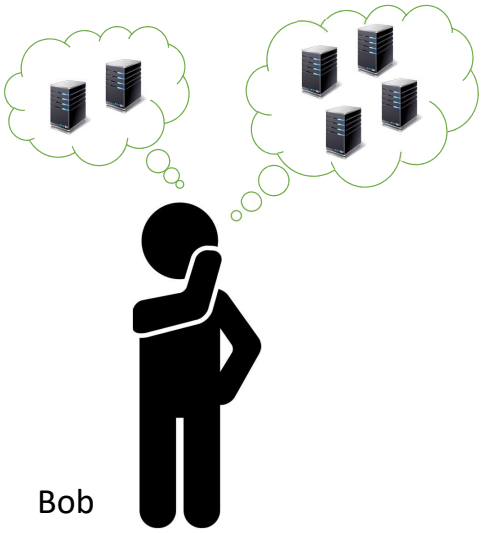

Figura 3: Administrador do Sistema com Perguntas Sobre Qual Servidor Montar.

2.2 Computação de Borda Móvel

O termo MEC foi padronizado pelo Instituto Europeu de Normas de Telecomunicações (ETSI)[1] e pelo Industry Specification Group (ISG). O grupo ISG inclui a Nokia Networks,

[1] https://www.etsi.org/technologies/multi-access-edge-computing

a Intel, a Vodafone, a IBM, a Huawei e a NTT DOCOMO, para citar algumas. O MEC também é reconhecido pela European 5G Parceria público-privada de infra-estrutura como uma tecnologia emergente para redes 5G (HU et al., 2015).

De acordo com o ETSI, o MEC é definido da seguinte forma: *"A computação de borda móvel fornece um ambiente de serviços de TI e recursos de computação em nuvem na borda da rede móvel, dentro da rede de acesso de rádio (RAN) e próxima aos assinantes móveis"* (HU et al., 2015). O MEC oferece recursos de computação em nuvem no RAN. Permitindo o tráfego móvel direto entre a rede principal e o usuário final, o MEC conecta o usuário diretamente a rede de borda habilitada para serviço de nuvem mais próxima. Implantar o MEC na estação base melhora a computação e evita gargalos e falhas do sistema (JARARWEH et al., 2016a; SATRIA; PARK; JO, 2017a). De acordo com o white paper publicado pela ETSTI, o MEC pode ser caracterizado pelas seguintes premissas:

Local: As plataformas MEC podem ser executadas isoladas do resto da rede, enquanto elas têm acesso a recursos locais. Isso é muito importante para cenários de máquina a máquina (M2M). A propriedade do MEC de segregação de outras redes também torna menos vulnerável.

Proximidade: Sendo implantado no local mais próximo, o MEC tem uma vantagem para analisar e materializar grandes volumes de dados. isto também é benéfico para dispositivos com fome de computação, como realidade aumentada (AR), análise de vídeo etc.

Latência Inferior: Os serviços MEC são implantados no local mais próximo aos dispositivos do usuário, isolando a movimentação de dados da rede da rede principal. Assim, a experiência do usuário é considerada de alta qualidade, com latência ultrabaixa e alta largura de banda.

Reconhecimento de local: Dispositivos distribuídos por borda utilizam sinalização de baixo nível para compartilhamento de informações. MEC recebe informações de dispositivos de borda dentro da rede de acesso local para descobrir a localização dos dispositivos.

Informações de Contexto da Rede: Aplicativos que fornecem informações de rede e serviços de dados de rede em tempo real podem beneficiar negócios e eventos implementando o MEC em seu modelo de negócios. Com base nas informações em tempo real da RAN, esses aplicativos podem estimar o congestionamento da célula de rádio e da largura de banda da rede. Isso os ajudará no futuro a tomar decisões inteligentes para uma melhor prestação de serviços aos clientes.

Portanto, o MEC traz recursos de computação e armazenamento para a borda da rede móvel, melhorando assim a velocidade com que os dados são processados (MOBILE...,).

Como uma tecnologia de ponta promissora, o MEC pode ser aplicado a cenários móveis, sem fio e com fio usando plataformas de software e hardware, localizadas na borda da rede e próximas aos usuários finais, enquanto ainda fornece integração perfeita de vários serviços de aplicativos de provedores. e fornecedores de assinantes móveis, empresas e outros (ABBAS et al., 2017).

Os servidores MEC são implantados na plataforma de computação genérica dentro da rede e permitem que aplicativos com reconhecimento de atraso sejam executados em estreita proximidade com os usuários finais. Essa abordagem alivia o *backhaul* e a rede principal e é crucial para permitir serviços móveis de baixa latência, alta largura de banda e agilidade (TRAN et al., 2016). Devido à distribuição geográfica densa, proximidade aos consumidores, suporte de alta mobilidade e plataforma aberta, o MEC pode suportar aplicativos e serviços com latência reduzida e melhor qualidade de serviço. Assim, o MEC está se tornando um importante facilitador de aplicativos e serviços de IoT centrados no consumidor que exigem operações em tempo real (CORCORAN; DATTA, 2016; CHEN et al., 2016; LI; WU, 2018; PREMSANKAR; TALEB, 2018).

A arquitetura tradicional do MEC permite o uso de seus serviços com baixa latência, reconhecimento de localização e suporte a mobilidade para compensar as desvantagens da computação em nuvem (SATRIA; PARK; JO, 2017b). No MEC, aplicativos sensíveis à latência e intensivos de computação, como realidade aumentada e processamento de imagens, podem ser hospedados na borda da rede (GUPTA; JAIN; CHAN, 2016).

A Figura 4 apresenta tal arquitetura. A mesma está dividido em três camadas:

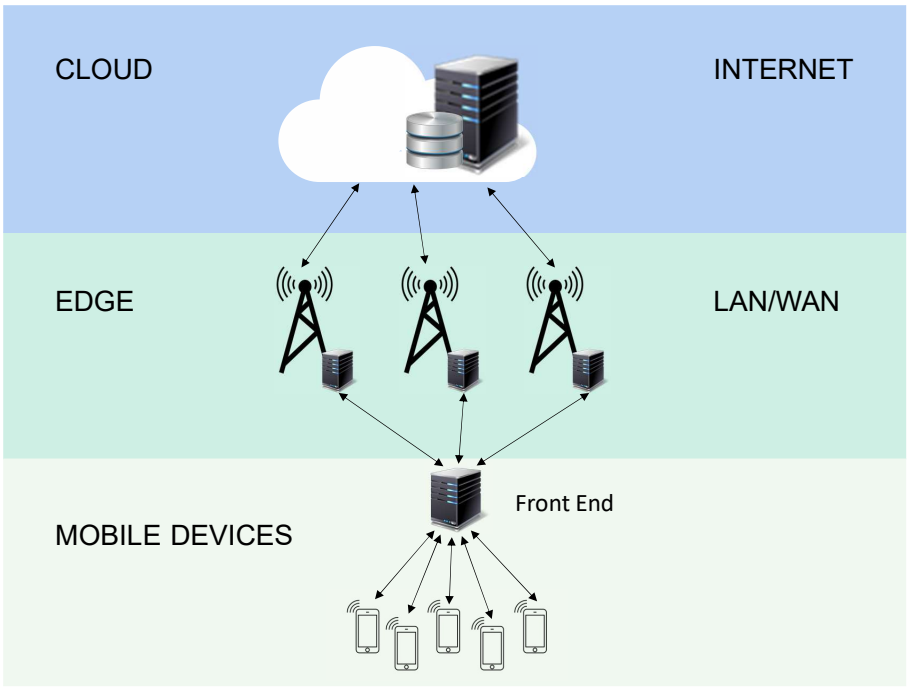

Figura 4: Arquitetura MEC Completa

1. Cloud: grande armazenamento de dados.
2. Edge: Exibe as torres onde os servidores serão instalados.
3. Dispositivos móveis: exibe dispositivos móveis (smartphones, tablets) junto com o FrontEnd.

Dispositivos móveis enviam solicitações para FrontEnd, que é responsável por encaminhá-las para o servidor na torre mais próxima. Quando a solicitação exige uma resposta mais rápida, o processamento ocorre no próprio servidor, sem a necessidade de ser roteado para a nuvem. A comunicação que ocorre nesta camada é feita através de LAN/WAN. Quando a solicitação demora mais para ser atendida, a fim de obter uma resposta mais precisa, ela vai para a camada de nuvem, onde a comunicação ocorre pela Internet, portanto, pode levar um pouco mais de tempo para ser resolvido. Com isso em mente, concentramos nosso trabalho nas duas primeiras camadas, onde o processamento ocorre mais próximo do usuário e onde a latência é menor. Mais especificamente, este trabalho explora o problema da colocação de servidores. Considerando essas três torres que estão localizadas em lugares distintos onde devo implantar servidores com poderes de computação distintos.

3 Trabalhos Relacionados

Alguns trabalhos relacionados propõem arquiteturas MEC em **cooperação com outras camadas**. No entanto, esses documentos não exploram diferentes configurações de recursos computacionais. *Tong et. al* propõe implantar servidores em nuvem na borda da rede e posicionar os servidores em hierarquias distribuídas geograficamente para usar a nuvem para atender aos picos de cargas de solicitações provenientes do MEC (TONG; LI; GAO, 2016). *Chen et. al* foca em um algoritmo de otimização para calcular onde processará a carga de trabalho (MEC ou nuvem). Os autores usam como validação, aplicativos genéricos que exigem um alto nível de processamento (CHEN et al., 2016). Autores em (LI; WU, 2018) concentram-se na capacidade do MEC e da nuvem de processar grandes cargas de dados. O objetivo principal do trabalho é tentar reduzir a latência entre as duas camadas atribuindo pesos às complexidades das tarefas a serem executadas remotamente.

Alguns artigos enfocam o problema do consumo de energia de dispositivos móveis com o auxílio de uma arquitetura MEC. *Zhang et. al* estudaram mecanismos de *offloading* investigando uma arquitetura MEC e redes 5G heterogêneas (KE; S; L, 2016). Os autores formularam um problema de otimização para minimizar o consumo de energia, observando o tempo de processamento e a transferência de dados. *Trinh et. al* estudaram o potencial do MEC para mitigar a limitação de bateria de dispositivos IoT (TRINH; YAO, 2017). Os autores utilizaram uma aplicação de reconhecimento facial para demonstrar a viabilidade de descarregar políticas de decisão. *Mao et. al* também propuseram um algoritmo de otimização de bateria em dispositivos móveis (MAO; ZHANG; LETAIEF, 2016). O algoritmo inclui frequências do servidor MEC, ciclo da CPU e taxa de latência. Uma vantagem desse algoritmo é que as decisões dependem apenas das informações instantâneas do lado do servidor, sem exigir informações sobre a distribuição das solicitações da tarefa.

Ao contrário desses estudos, nosso trabalho não analisa diretamente o gasto de energia de dispositivos clientes. No entanto, as métricas de MRT adotadas aqui estão proporcionalmente relacionadas ao gasto de energia de dispositivos clientes. Quanto mais tempo a solicitação levar para executar, maior será o consumo de energia.

Os trabalhos relacionados mais próximos são aqueles que lidam com o **planejamento de infraestrutura do MEC**. *Premsankar et. al* realizara uma avaliação experimental real com um jogo eletrônico de alta interatividade, mas com a limitação de ter apenas um cliente móvel (PREMSANKAR; TALEB, 2018). O trabalho de *Jararweh et. al* desenvolveu um simulador de arquitetura que incluía uma camada Cloudlet e uma camada MEC (JARARWEH et al., 2016a). Os autores tentam aumentar a área de cobertura para usuários móveis, onde os usuários podem fazer solicitações com custos mínimos em termos de gasto de energia. Algumas limitações deste trabalho advêm do fato de que os autores só adotam o número de pedidos como um parâmetro de entrada e o uso de um simulador

customizado sem explicações detalhadas sobre suas características. *Liu et. al* adotaram um modelo de cadeia de Markov como um processo de decisão sobre onde as tarefas devem ser executadas (localmente ou no servidor MEC) (LIU et al., 2016). O modelo leva em consideração o status de enfileiramento do buffer de tarefas, o estado da execução do dispositivo móvel e o estado da rede. No entanto, os autores não consideram o servidor MEC com vários nós paralelos e, portanto, não aproveitam o potencial do paralelismo de servidor. *Badri et. al* também usou cadeias de Markov para decidir onde executar solicitações em várias torres MEC (BADRI et al., 2017). O algoritmo leva em consideração o movimento dos usuários, o custo de comunicação entre usuários e servidores, o custo de execução de cada servidor e o custo de alocação. Nosso trabalho não se concentra em vários servidores, mas no planejamento de uma infraestrutura mínima com um único servidor MEC, considerando o paralelismo em vários *containeres*.

Ao contrário da nossa proposta, os trabalhos mencionados acima não exploram o nível de utilização de recursos e apenas alguns consideram a métrica MRT. Todos os artigos são limitados em termos de parametrização das avaliações, com no máximo três parâmetros de configuração de arquitetura. Nenhum dos artigos (com exceção de (PREMSANKAR; TALEB, 2018)) abordou aplicativos com um alto nível de interação. Além disso, nenhum dos jornais adotou redes de Petri Estocásticas.

4 Visão Geral da Arquitetura

Este trabalho apresenta um SPN para modelar uma arquitetura MEC em que os recursos de servidores únicos são paralelizados com microsserviços em execução em vários *containeres*. A Figura 5 ilustra a arquitetura do MEC que estamos considerando para modelagem e análise de desempenho. A arquitetura é composta de três partes principais: **Mobile Devices**, **FrontEnd** e **Edge Computing**. Os dispositivos móveis solicitam serviços para o FrontEnd e o FrontEnd distribui solicitações entre a computação de borda.

Os fluxos de dados são gerados por aplicativos em execução no **Mobile Devices**. Esses aplicativos podem variar de aplicativos de monitoramento de integridade (RAHMANI et al., 2018) para aplicativos de jogos (ZHANG et al., 2019), por exemplo. Como mencionado anteriormente, neste documento, nos concentramos na borda, e não na nuvem, com o objetivo de trabalhar com aplicativos altamente responsivos nos quais é necessária uma resposta rápida. Portanto, a proximidade de servidores e o alto poder do computador são obrigatórios para fornecer disponibilidade eficiente de recursos e baixo tempo médio de resposta.

O **FrontEnd** é responsável por receber solicitações dos dispositivos móveis e encaminhá-las aos servidores. Para isso, o servidor mestre é responsável por escolher qual servidor é mais adequado para executar a solicitação. Neste trabalho, para tomar essa decisão, estamos considerando dois parâmetros diferentes: *(a)* cada servidor pode ter diferentes configurações (por exemplo, número de núcleos) e *(b)* a distância da estação base para os dispositivos móveis, o que afeta a decisão de posicionamento do servidor. Dada essa arquitetura, o objetivo principal deste trabalho é propor um SPN para avaliar o desempenho de diferentes configurações e encontrar a solução de posicionamento de recursos mais eficiente.

Na camada **Edge Computing**, estamos considerando servidores MEC com latência distinta (Servidor A, Servidor B e Servidor C, como mostrado na Figura 5). Cada servidor tem um conjunto de nós escravos que são basicamente micro serviços executados em *containeres*. Cada *container* é executado em um único núcleo de servidor. Embora as máquinas virtuais tenham sido altamente adotadas no campo de pesquisa do MEC, os *containeres* permitem maior flexibilidade para dimensionar o poder de computação de acordo com a demanda dinâmica.

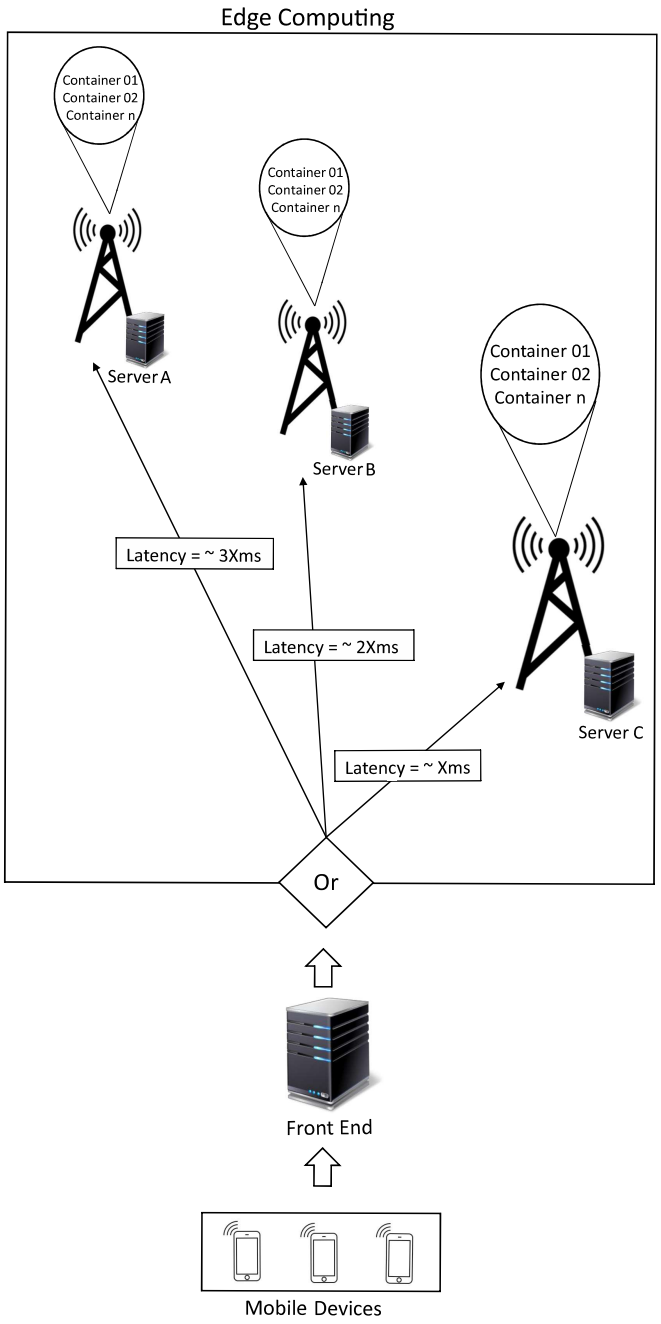

Figura 5: Arquitetura MEC Adotada para Avaliação de Desempenho.

5 Modelo SPN Base

Um modelo de SPN é útil quando um gerente de serviços deseja planejar e analisar alterações no sistema antes de implementá-las. A Figura 6 apresenta o modelo SPN proposto, composto por três blocos principais (destacados em vermelho):

1. **Bloco de Admission**: responsável por gerar solicitações de usuários;
2. **Bloco FrontEnd**: representa o servidor FrontEnd e é responsável por receber as solicitações do usuário e encaminhá-las para um dos servidores disponíveis. As políticas de balanceamento de carga podem ser aplicadas neste momento. Neste trabalho as solicitações são encaminhadas para qualquer um dos servidores disponíveis;
3. **Bloco Edge Computing**: representa os servidores (A, B e C) que são responsáveis por receber e processar os pedidos, distribuindo os dados entre os escravos do nó, que em nosso trabalho são *containers*.

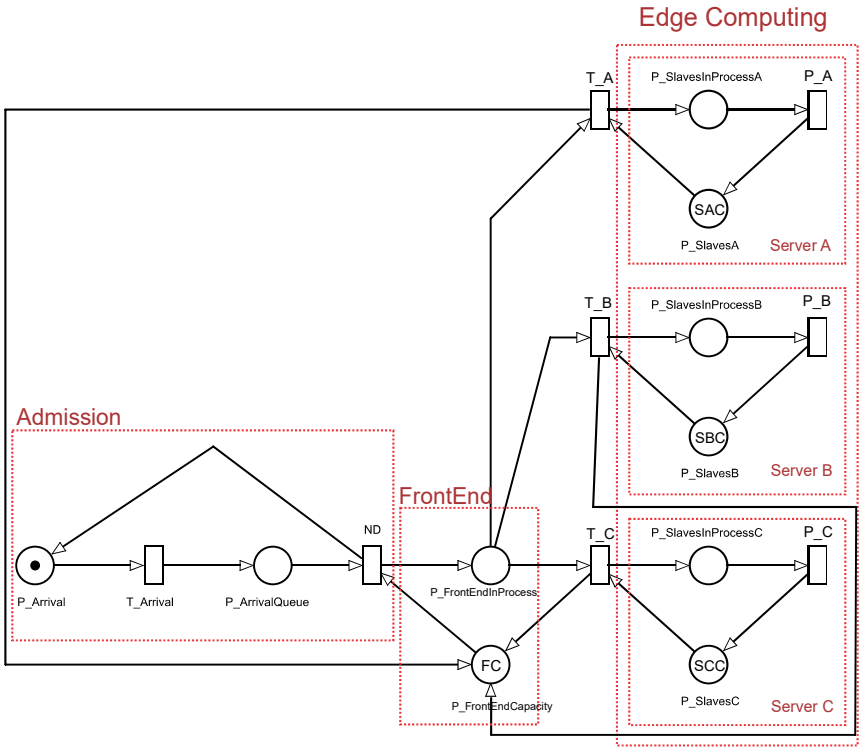

Figura 6: Modelo SPN para a Arquitetura MEC.

Tabela 1 apresenta a descrição dos elementos do modelo. As transições temporizadas são configuradas com distribuições de probabilidade e o administrador do sistema deve preencher essas distribuições de acordo com a literatura ou realizar medições do sistema.

Todas as transições temporizadas têm infinitas semânticas de servidor, o que significa que as transições são disparadas em paralelo quando as transições são acionadas.

Tabela 1: Descrição dos Elementos do Modelo SPN

Tipo	**Elemento**	**Descrição**	**Semântica do Servidor**
Lugares	P_Arrival	Aguarda novos pedidos	X
	P_InputQueue	Fila de entrada antes de acessar o servidor FrontEnd	X
	P_FrontEndInProcess	Solicitações na fila do servidor FrontEnd	X
	P_FrontEndCapacity	Capacidade do FrontEnd	X
	P_SlavesInProcessA	Solicitações na fila do servidor A	X
	P_SlavesA	Capacidade do Servidor A	X
	P_SlavesInProcessB	Solicitações na fila do servidor B	X
	P_SlavesB	Capacidade do Servidor B	X
	P_SlavesInProcessC	Solicitações na fila do servidor C	X
	P_SlavesC	Capacidade do Servidor C	X
Transições Temporizadas	ND	Atraso na rede	Servidor Infinito
	T_A	Tempo de transferência para o servidor A	Servidor Infinito
	T_B	Tempo de transferência para o servidor B	Servidor Infinito
	T_C	Tempo de transferência para o servidor C	Servidor Infinito
	P_A	Tempo de processamento do servidor A considerando uma solicitação	Servidor Infinito
	P_B	Tempo de processamento do servidor B considerando uma solicitação	Servidor Infinito
	P_C	Tempo de processamento do servidor C considerando uma solicitação	Servidor Infinito
Marcações de Lugares	FC	Capacidade máxima do FrontEnd	X
	SAC	Capacidade máxima do servidor A	X
	SBC	Capacidade máxima do servidor B	X
	SCC	Capacidade máxima do servidor C	X

O bloco *Admission* consiste em dois lugares, *P_Arrival* e *P_ArrivalQueue*. *P_Arrival* representa a geração de solicitações de usuários e *P_ArrivalQueue* representa a aceitação desses pedidos na fila. A transição *T_Arrival* é configurada com a taxa de solicitação de

chegada. Neste modelo, consideramos que o tempo de chegada é distribuído exponencialmente, mas pode ser facilmente modificado para se adequar a outras distribuições. A transição *ND* (atraso de rede) representa os tempos de envio e recebimento da solicitação para o servidor FrontEnd. *ND* é acionado assim que há um token em *P_ArrivalQueue* e pelo menos um token em *P_FrontEndCapacity*.

Quando a transição *ND* dispara, o bloco *FrontEnd* é atingido e um token é consumido dos lugares *P_ArrivalQueue* e *P_EncurtoCapacidade* e um token é produzido em locais *P_FrontEndInProcess* e *P_Arrival*. A quantidade de tokens em *P_FrontEndInProcess* representa o enfileiramento de pedidos em *FrontEnd*. A fila acontece porque não há recursos disponíveis nos servidores. Quando o processamento da solicitação é iniciado, os tokens são consumidos a partir do *P_FrontEndInProcess*. Se houver recursos disponíveis em um dos três servidores, uma das transições T_A, T_B ou T_C será acionada e a solicitação seguirá o caminho a ser processado .

As transições T_A, T_B e T_C representam o tempo de transferência dos servidores *FrontEnd* para o MEC. Os blocos de servidores do MEC têm a mesma operação. Então, considerando o Servidor A, a transição P_A representa o início do processamento. O fogo P_A está condicionado ao número de nós disponíveis para processar pedidos em *P_SlavesA* (marcação SAC). Dado o modelo proposto, é possível configurar 15 parâmetros (ver Tabela 1), dando a possibilidade de executar um grande número de diferentes cenários. Os parâmetros configuráveis são: 8 transições temporizadas e 4 lugares relacionados às capacidades dos recursos. Qualquer configuração desses parâmetros impacta substancialmente o tempo médio de serviço e, consequentemente, o custo da infraestrutura. A possibilidade de executar um grande número de cenários é uma das principais contribuições deste trabalho.

5.1 Métricas de Desempenho

Nesta seção, definimos as métricas de desempenho que usamos para avaliar a arquitetura MEC com base no modelo de SPN proposto. Neste trabalho, consideramos duas métricas: o tempo médio de resposta (MRT), que se refere ao tempo necessário para que uma solicitação seja executada no sistema, e o nível de utilização que lida com o consumo de uma dada execução nas máquinas presentes nos servidores. O MRT pode ser obtido a partir da Lei de Little (LITTLE, 1961), que relaciona o número médio de pedidos em andamento em um sistema, $RequestsInProcess$, a taxa de chegada de novas solicitações, ARR e tempo médio de resposta, MRT. A taxa de chegada é o inverso do atraso de chegada, AD. A Lei de Little requer um sistema estável, o que significa que a taxa de chegada é menor que o tempo de serviço. Portanto, Equação 5.1 corresponde à Lei do

Little para o MRT usado em nosso modelo:

$$\mathbf{MRT} = \frac{RequestsInProcess}{ARR} \tag{5.1}$$

A equação 5.2 define como calculamos o número de solicitações em andamento, ou seja, $RequestsInProcess$. Para isso, somamos a quantidade de tokens em cada lugar que representa uma requisição em processo, $Esp(Place)$, onde $Esp(Place) = (\sum_{i=1}^{n} P(m(Place)) = i)\, timesi)$ e $Place = \{FC, P\_EslavesInProcessA, P\_SlavesInProcessB, P\_SlavesInProcessC\}$.

$$\begin{aligned} \mathbf{RequestsInProcess} = Esp(P\_FrontEndInProcess)+ \\ Esp(P\_SlavesInProcessA)+ \\ Esp(P\_SlavesInProcessB)+ \\ Esp(P\_SlavesInProcessC) \end{aligned} \tag{5.2}$$

O uso percentual de recursos de cada máquina, n, é dado por $U(n)$ (veja Equações 5.3, 5.4 e 5.5). O uso de recursos é calculado pelo número de tokens na fila que estão consumindo recursos do servidor dividido pela capacidade total de recursos daquele servidor. Em outras palavras, o uso de recursos é a quantidade de nós escravos que estão sendo usados. Dada a utilização por servidor, também podemos calcular o uso de todos os recursos disponíveis no sistema usando a Equação 5.6, em que $n\ _servers$ é o número total de servidores no sistema.

$$\mathbf{U(A)} = \frac{Esp(P\_SlavesInProcessA)}{SCA} \tag{5.3}$$

$$\mathbf{U(B)} = \frac{Esp(P\_SlavesInProcessB)}{SCB} \tag{5.4}$$

$$\mathbf{U(C)} = \frac{Esp(P\_SlavesInProcessC)}{SCC} \tag{5.5}$$

$$\mathbf{U(all)} = \frac{U(A) + U(B) + U(C)}{n\_servers} \tag{5.6}$$

5.2 Estudos de caso

A principal contribuição do nosso modelo proposto é a versatilidade e flexibilidade para avaliar diferentes cenários. Assim, para mostrar a aplicabilidade de nosso modelo proposto, apresentamos duas análises numéricas que exemplificam como nosso modelo pode ser utilizado de forma prática por um administrador de sistemas. Em nossos cenários, a arquitetura do MEC é composta de três servidores, e as máquinas disponíveis para avaliação possuem 8, 16 ou 32 núcleos.

Em (PREMSANKAR; TALEB, 2018), os autores avaliaram uma arquitetura MEC com um único dispositivo móvel como um cliente e *containeres* executando os serviços. Os autores avaliaram um jogo 3D chamado *Neverball*, onde o jogador deve inclinar o chão para controlar a bola, a fim de coletar moedas e chegar a um ponto de saída antes que o tempo se esgote.

Nós consideramos os parâmetros do sistema usados em (PREMSANKAR; TALEB, 2018) como parâmetros de entrada para nosso modelo. Portanto, nosso estudo desenvolve o trabalho em (PREMSANKAR; TALEB, 2018), realizando análises numéricas para avaliar os cenários considerando múltiplos parâmetros.

Consideramos um dos seus cenários com uma resolução de jogo de 800 *times*600 pixels. O valor do parâmetro adotado correspondente ao atraso de processamento de uma solicitação é de 10.7ms. Todos os parâmetros utilizados são apresentados na Tabela 2.

Tabela 2: Valores Atribuídos as Transições Temporizadas.

Transições Temporizadas	**Tempo (ms)**
T_A	8.3
T_B	5.2
T_C	2.5
P_A	10.7
P_B	10.7
P_C	10.7
ND	2.3

5.2.1 Estudo de Caso 1

No estudo de caso 1, planejamos uma arquitetura composta de três servidores MEC; e para isso, o objetivo é minimizar o MRT e alocar eficientemente tarefas em máquinas que não apresentam alta taxa de consumo de recursos. O objetivo é minimizar os recursos desdobrados, ao mesmo tempo em que atende a demanda solicitada, a fim de diminuir os investimentos financeiros.

Assumimos que o MRT desejado é de até 40ms, e a taxa de uso desejada deve estar entre 40% e 50% para evitar sobrecarga e ociosidade nos servidores. Nesse caso, a taxa de chegada do pedido é de 0,05 solicitações por milissegundos. A lista de todas as 27 configurações possíveis das máquinas é apresentada na Tabela 3.

A Figura 7 apresenta os resultados deste estudo de caso. A configuração 1 apresenta uma configuração com MRT igual a 995,82ms, muito acima das outras, por isso omitimos essa configuração para fornecer uma melhor visualização. As configurações que satisfazem os requisitos (MRT e taxas de utilização de recursos) são #27, #18, #24, #15, #9, #6, #26, #21, #12, #7, #23, #3, #14 e #8.

A melhor configuração para este caso é #27, que é composta dos melhores servidores disponíveis (todos com 32 núcleos), e a segunda melhor configuração (ServidorA =

Tabela 3: Configurações Possíveis para os Estudos de Caso 1 e 2.

Configuração	**Servidor A**	**Servidor B**	**Servidor C**
#1	8	8	8
#2	8	8	16
#3	8	8	32
#4	8	16	8
#5	8	16	16
#6	8	16	32
#7	8	32	8
#8	8	32	16
#9	8	32	32
#10	16	8	8
#11	16	8	16
#12	16	8	32
#13	16	16	8
#14	16	16	16
#15	16	16	32
#16	16	32	8
#17	16	32	16
#18	16	32	32
#19	32	8	8
#20	32	8	16
#21	32	8	32
#22	32	16	8
#23	32	16	16
#24	32	16	32
#25	32	32	8
#26	32	32	16
#27	32	32	32

16, ServidorB = 32, ServidorC = 32) é focada em dar mais poder de computação para servidores que estão longe do usuário.

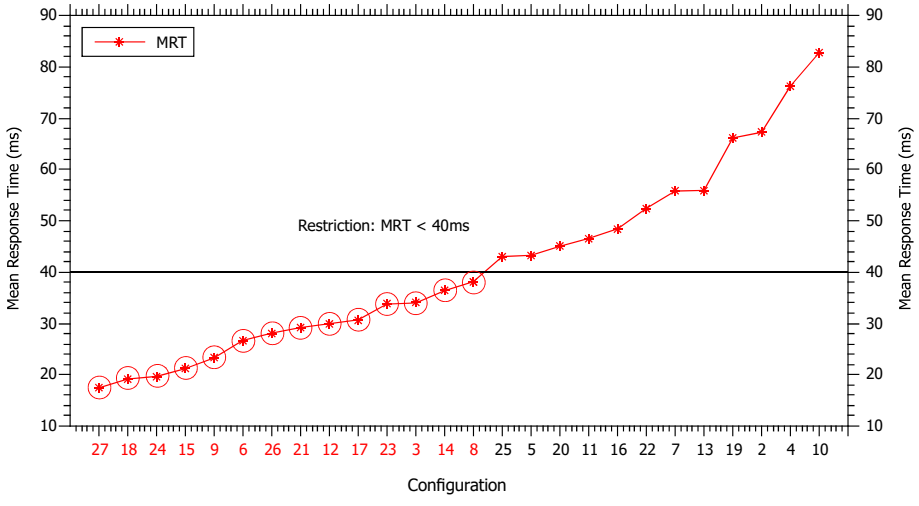

Figura 7: Resultados do Estudo de Caso 1 Considerando MRT $\leq$ 40ms.

A Figura 8 apresenta a utilização dos recursos das configurações das máquinas. Agora, as configurações que se ajustam ao requisito são #25, #14, #8, #3, #16, #20, #22 e #5. Nesse caso, a melhor solução (ServidorA = 32, ServidorB = 32, ServidorC = 8) é baseada em máquinas com mais poder de computação mais próximo dos usuários.

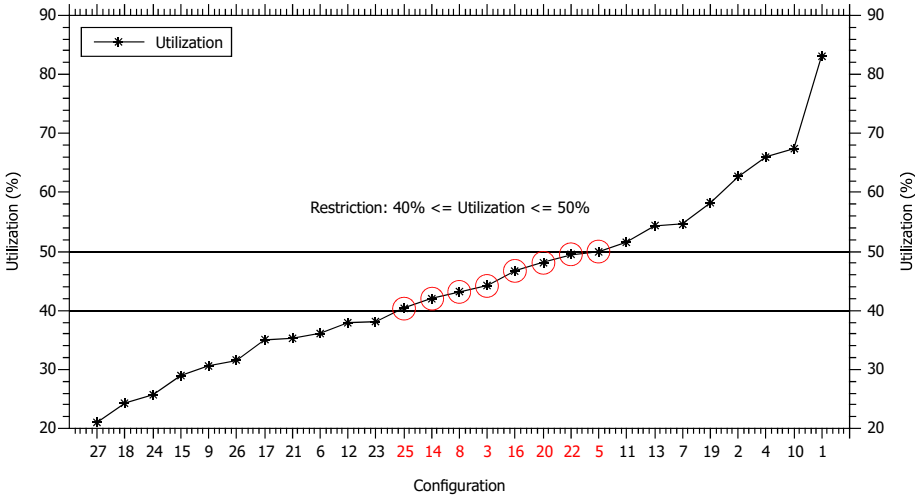

Figura 8: Estudo de Caso 1, Considerando 40% ≤ Utilização de Recursos ≤ 50%.

Então, analisando os resultados de MRT e utilização de recursos, pode-se notar que as configurações #14 (ServidorA = 16, ServidorB = 16, ServidorC = 16), #8 (ServidorA = 8, ServidorB = 32, ServidorC = 16), e #3 (ServidorA = 8, ServidorB = 8, ServidorC = 32) se encaixam nos dois requisitos, conforme apresentado na Tabela 4. Na configuração #14, a solução equilibra o poder computacional igualmente, mas na configuração #8 e #3, as soluções são heterogêneas, alocando mais poder computacional em servidores mais distantes do usuário.

Tabela 4: Configurações Propostas para o Estudo de Caso 1.

Configuração	**MRT (ms)**	**Utilização Média (%)**
#3	33,98	44,22
#8	38,14	43,13
#14	36,37	42,05

5.2.2 Estudo de Caso 2

Neste estudo de caso, consideramos hipoteticamente que as três máquinas já foram escolhidas (8, 16 e 32 núcleos), o objetivo do estudo de caso 2 é minimizar o MRT e a utilização de recursos. A taxa de chegada do pedido é de 0,02 req/ms e a Tabela 5 apresenta todas as configurações possíveis para este estudo de caso.

As Figuras 9 e 10 apresentam o MRT e a utilização de recursos, respectivamente. Ambas as métricas apresentam a mesma ordem, mostrando que a configuração #3 (ServidorA = 8, ServidorB = 16, ServidorC = 32) produz o melhor MRT (21.08ms) e utilização de recursos (14.41%). Por outro lado, a configuração 1 (ServidorA = 32, ServidorB = 16, ServidorC = 8) obteve os piores resultados, tendo MRT igual a 44,5ms e utilização de recursos igual a 24,53%.

Tabela 5: Configurações Possíveis para o Estudo de Caso 2.

Configuração	**Servidor A**	**Servidor B**	**Servidor C**
#1	32	16	8
#2	32	8	16
#3	8	16	32
#4	8	32	16
#5	16	8	32
#6	16	32	8

Nesse caso, podemos afirmar que, para minimizar o MRT e a utilização de recursos, é necessário ter um servidor com o maior poder computacional (32 núcleos) próximo ao Front End e o servidor com a menor capacidade distante do Front End. .

Configurações #5 (ServidorA = 16, ServidorB = 8, ServidorC = 32) e #4 (ServidorA = 8, ServidorB = 32, ServidorC = 16) têm diferentes configurações de servidor, mas apresentam resultados semelhantes: configuration #5 apresentado MRT = 24,23ms e utilização de recursos = 16,13%; e a configuração #4 apresentou MRT = 25.77ms e utilização de recursos = 16.99%. No entanto, como a configuração #5 possui os melhores recursos no ServidorC (melhor latência e capacidade), essa configuração seria preferível à configuração #4.

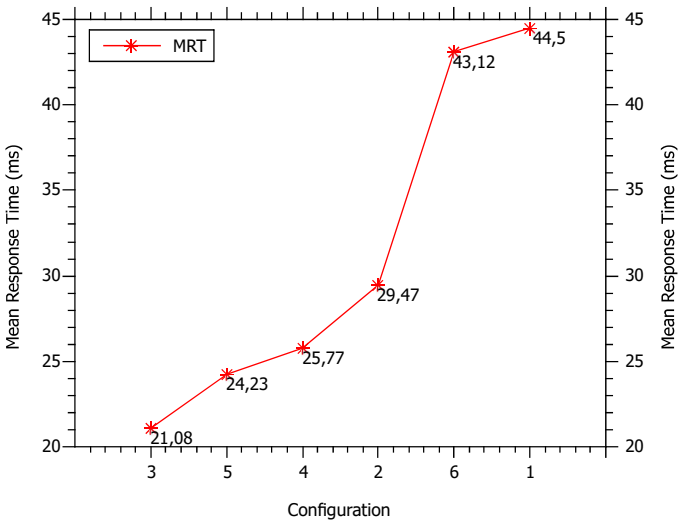

Figura 9: MRT para as 6 Configurações

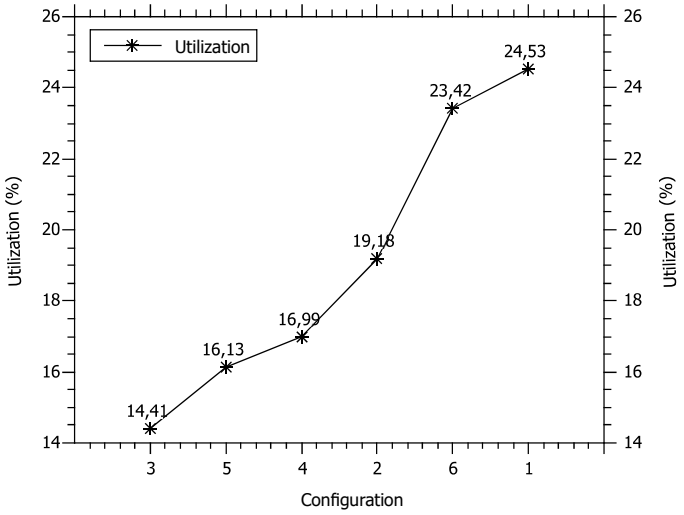

Figura 10: Utilização para as 6 Configurações

6 Modelo SPN Extendido

Quando desenvolvedores de aplicativos e provedores de serviços desejam planejar e projetar um ambiente MEC, eles devem ser capazes de estimar o tempo médio de resposta do sistema. As Funções de Distribuição Cumulativa (CDFs) podem indicar esse momento pela máxima probabilidade de absorção.

As CDFs estão associadas a uma distribuição de probabilidade específica e podem ser obtidas através de avaliações transitórias, gerando probabilidades com tempo tendendo a t. Em outras palavras, pode-se calcular a probabilidade **absorption (AP)** entre $[0, t)$, através de avaliações transitórias, quando $F(t)$ se aproxima de 1. Portanto, é possível estimar a probabilidade de concluir a execução do aplicativo antes de um tempo específico $[P(T < t)]$.

Além disso, os CDFs também podem indicar a probabilidade de um aplicativo ser processado e concluído dentro de um determinado intervalo de tempo $[P(t1 < T < t2) = P(T < t2) - P(T \leq t1)]$.

Os CDFs podem ser obtidos resolvendo modelos SPN adaptando o modelo a um estado absorvente (SILVA et al., 2018). A Figura 11 mostra a adaptação que fizemos em nosso modelo SPN apresentado anteriormente para calcular os CDFs. Basicamente, foram feitas três alterações: *(a)* um local de estado absorvente (denominado Finish) foi adicionado na parte direita do modelo, indicando que quando as solicitações chegam a este local, tais solicitações não mudam de estado; *(b)* no bloco de Admissão, o ciclo de feedback ($ND\ rightarrowP\ _{A}rrival$) foi retirado, indicando que novas solicitações não serão geradas de forma inconfundível; e *(c)* existe um novo parâmetro chamado BATCH (no local P_Arrival) que representa o número de jobs (tokens) que serão processados.

O CDF calcula a probabilidade desses trabalhos concluir o processamento do aplicativo em um determinado momento e em um determinado intervalo de tempo.

6.1 Estudos de Caso

Esta seção apresenta dois estudos de caso para mostrar a aplicabilidade da métrica CDF. Para ambos os estudos, usamos um nível de confiança de 95% e uma taxa de erro relativa máxima de 10%. Para cada teste, geramos 300 amostras aplicando a simulação transiente.

6.1.1 Estudo de Caso 1

A Tabela 6 apresenta os parâmetros usados na simulação deste estudo de caso. Observe que definimos $AD = 8.0ms$ e configuramos a capacidade em todos os servidores para 32,

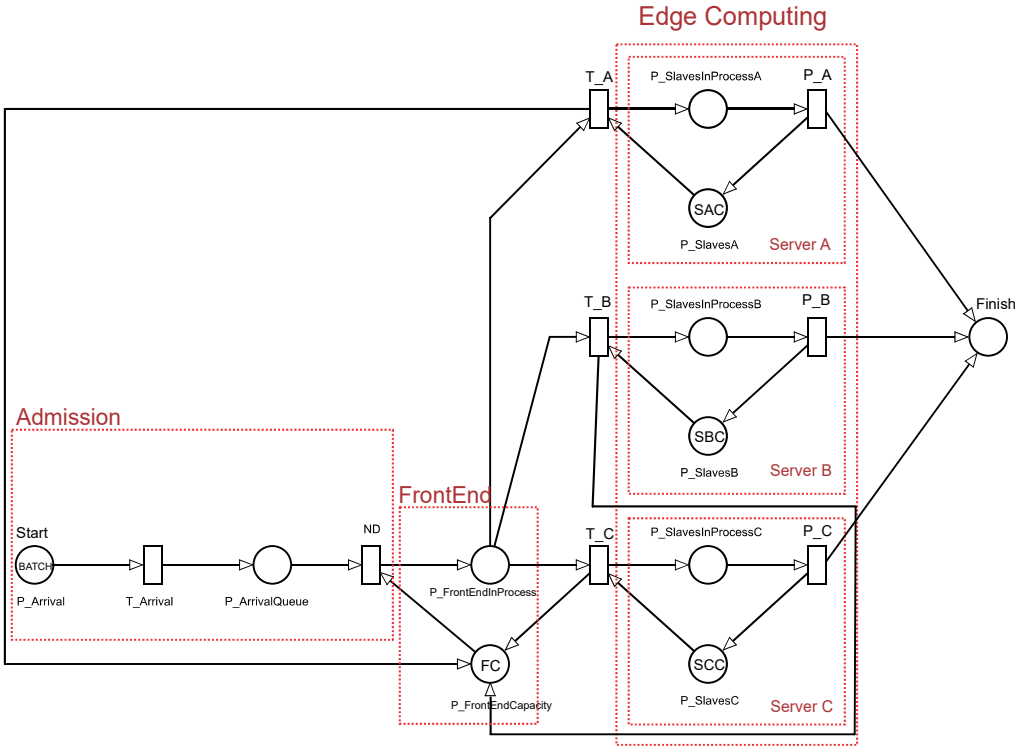

Figura 11: SPN Usado para Calcular o CDF

tentando executar um BATCH de 100 solicitações. Apenas a capacidade do *FrontEnd* (FC) foi alterada. De fato, variamos a capacidade do FrontEnd em três valores: 50, 100 e 150. As probabilidades foram calculadas de $t = 0ms$ a $t = 1000ms$ para as três execuções.

Tabela 6: Valores dos Parâmetros para o Estudo de Caso 1.

Execução	**AD**	**FC**	**SAC**	**SBC**	**SCC**	**BATCH**
Execução 1	8.0	50	32	32	32	100
Execução 2	8.0	100	32	32	32	100
Execução 3	8.0	150	32	32	32	100

6.1.1.1 Probabilidade de Absorção (AP)

A Figura 12 representa a CDF deste estudo de caso. A partir dos resultados, podemos ver que quanto maior a capacidade FrontEnd, menor será o tempo necessário para que a probabilidade de absorção atinja 100%. O tempo necessário para atingir AP = 100% depende da capacidade do FrontEnd, quanto maior a capacidade, menor o tempo necessário.

Quando a capacidade FrontEnd é 50, o CDF variou de AP = 0% em 680ms e levou 303ms para alcançar AP = 100%, enquanto execuções com capacidades de 100 e 150 levaram apenas 190ms e 130ms, respectivamente. Considerando AP = 50%, quando a capacidade FrontEnd é 50, 100 e 150, o tempo associado foi de aproximadamente 881ms,

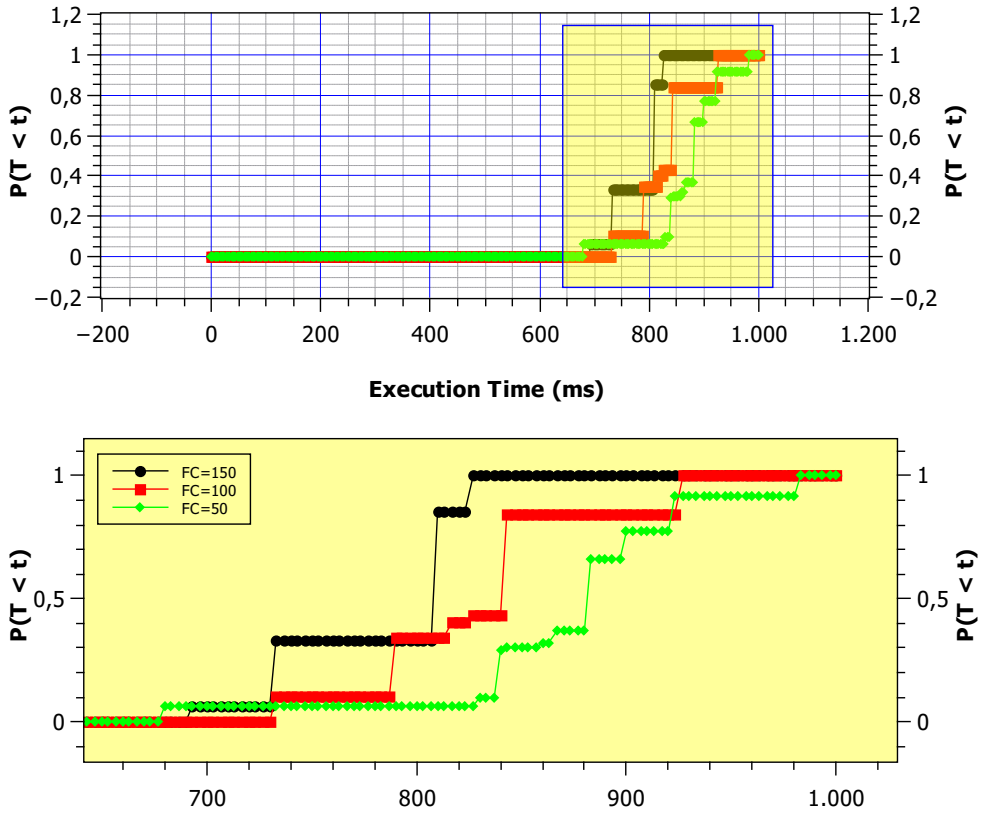

Figura 12: CDF para BATCH=100 e FC = {50,100,150}

841ms e 808ms, respectivamente. Observe que, conforme esperado, a capacidade menor apresentou um tempo maior.

Ao considerar um tempo específico, sua probabilidade de absorção é a seguinte. Em 800ms, FC = 50 está próximo de 0%, enquanto FC = 100 e FC = 150 apresentam AP = 32%. Neste caso, como os resultados são tão próximos, sugere-se adotar a opção FC = 100, a fim de diminuir os custos de investimento financeiro.

6.1.1.2 Probabilidade de Absorção por Intervalo (API)

Para calcular a probabilidade de absorção por intervalos (API), basicamente deve-se considerar dois pontos da curva CDF e subtrair as probabilidades correspondentes. Como exemplo, na Figura 12, se considerarmos HR = 100 e os intervalos 840ms (50%) e 940ms (100%), a API para 100 solicitações nesse intervalo será de 100%−50% = 50%. A Figura 13 apresenta as APIs levando em consideração dois intervalos de tempo, $700ms < T < 800ms$ e $800ms < T < 900ms$.

No primeiro intervalo ($700ms < T < 800ms$), as APIs eram aproximadamente 0%, 34% e 27%, para $FC = 50$, $FC = 100$ e $FC = 150$, respectivamente. Nesse intervalo, a execução com $FC = 50$ apresentou API = 0%; isso ocorreu porque, durante todo o intervalo de tempo, a API foi de 6,45% e, como nenhum aumento ocorreu, a API não

era maior que 0%. Por outro lado, as APIs quando $HR = 100$ (34%) e $HR = 150$ (27%) eram significativas. A API de $FC = 100$ é maior que $FC = 150$ porque, embora a 800ms, eles estão no mesmo nível, no ponto 700ms, a API de $FC = 150$ é ligeiramente superior a $FC = 100$.

No segundo intervalo de tempo ($800ms < T < 900ms$), as APIs são mais altas porque tendem a atingir 100%, assim como as curvas são mais acentuadas nesse intervalo. As APIs quando $FC = 50$, $FC = 100$ e $FC = 150$ eram 70%, 50% e 65%, respectivamente. A probabilidade quando $FC = 100$ é menor que os outros porque apresenta um aumento mais lento. Enquanto $FC = 100$ varia de 30% a 80%, $FC = 50$ varia de 10% a 80%.

Portanto, se alguém quiser que a execução do aplicativo de 100 solicitações termine entre 700ms a 800ms, a simulação sugere a adoção de $FC = 100$, ou seja, uma capacidade FrontEnd de 100 (por exemplo: 100 *containeres*), porque se adotar $FC = 50$, a API é 0%.

Considerando o segundo intervalo, se um requerer que o tempo médio de execução fique entre 800ms e 900ms, a configuração $FC = 50$ retornará a maior probabilidade de conclusão. Este é um resultado muito interessante, porque ao adotar $FC = 50$ há uma economia de recursos, considerando as três possibilidades para a capacidade FrontEnd.

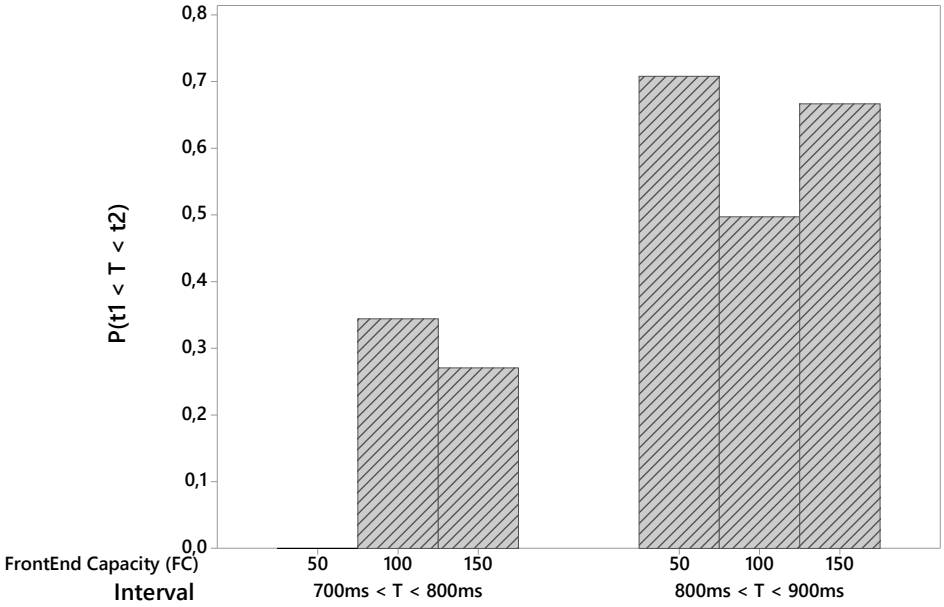

Figura 13: Probabilidade de Absorção com Base em Dois Intervalos de Tempo para o Estudo de Caso 1

6.1.2 Estudo de Caso 2

Tabela 7 apresenta os parâmetros usados no estudo de caso 2. Observe que definimos $AD = 2.0ms$ e definimos a capacidade de todos os servidores para 64, para executar um BATCH de 1000 solicitações. Aumentamos o valor de BATCH de 100 para 1000 para ter uma carga de trabalho maior em comparação com o estudo de caso anterior. Nós variamos a capacidade do FrontEnd (marcação FC) em três valores: 250, 500 e 1000.

Tabela 7: Valores dos Parâmetros para o Estudo de Caso 2.

Execução	AD	FC	SAC	SBC	SCC	BATCH
Execução 1	2.0	250	64	64	64	1000
Execução 2	2.0	500	64	64	64	1000
Execução 3	2.0	1000	64	64	64	1000

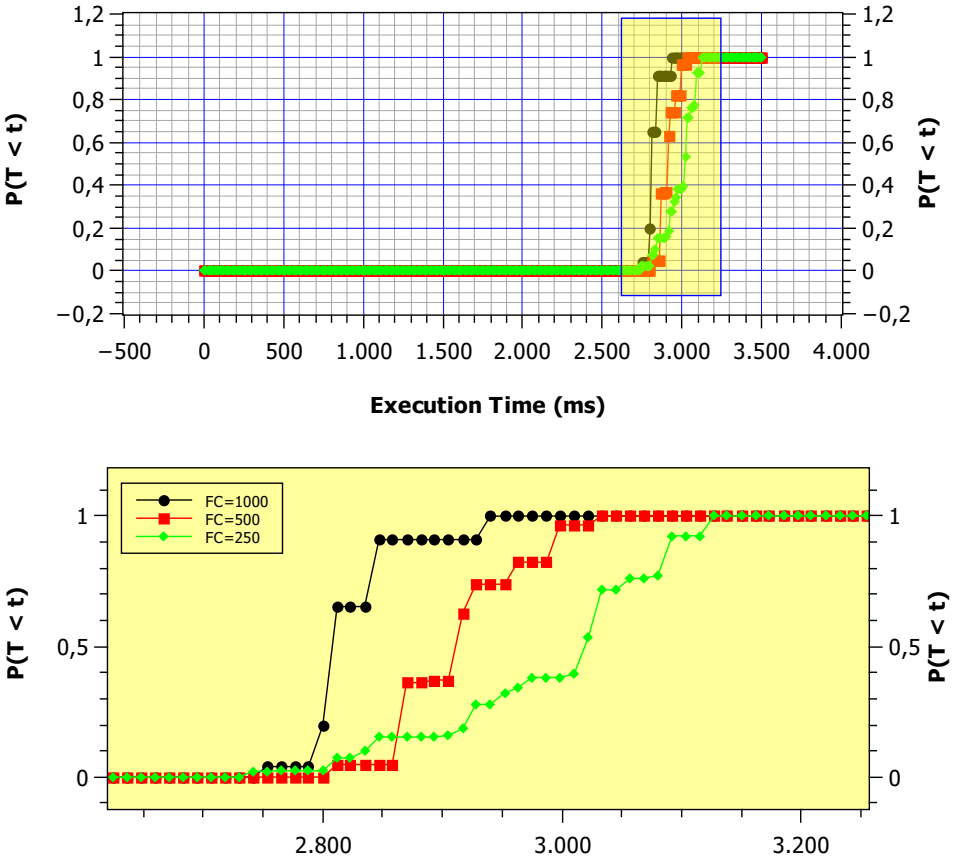

Figura 14: CDF para BATCH=1000 e FC = {250,500,1000}

6.1.2.1 Probabilidade de Absorção (AP)

A Figura 14 exibe o CDF correspondente do estudo de caso 2. Os pontos de acesso foram calculados de $t = 0ms$ a $t = 3500ms$ para todos os experimentos. Como esperado, quanto maior a capacidade do FrontEnd, menos tempo é necessário para o AP atingir 100%. Considerando AP = 50%, os tempos médios de absorção associados às capacidades $FC = 250$, $FC = 500$ e $FC = 1000$ são aproximadamente 3020ms, 2915ms e 2810ms, respectivamente.

Em geral, $FC = 1000$ apresenta os melhores resultados quando comparado com os outros. O sistema atinge AP = 100% a 3127ms, 3033ms e 2940ms para as capacidades FrontEnd 250, 500 e 1000, respectivamente. Note que para 50% ou 100%, as diferenças nos tempos de absorção entre diferentes capacidades são baixas ($cong100ms$). No entanto, em sistemas críticos, 100 ms podem representar um tempo de execução inaceitável. Curiosamente, em 2850ms, o AP para $FC = 250$ (15%) é maior que $FC = 500$ (5%).

Portanto, se aceitarmos o tempo máximo de 2850 ms, é mais vantajoso ter $FC = 250$, o que significa uma economia significativa de recursos.

6.1.2.2 Probabilidade de Absorção por Intervalo (API)

A Figura 15 mostra as APIs que obedecem a dois intervalos de tempo: $(2800ms < T < 2900ms)$ e $(2900ms < T < 3000)$. No primeiro intervalo, há um padrão ascendente, em que quanto maior a capacidade FrontEnd, maiores são as APIs para 1000 solicitações.

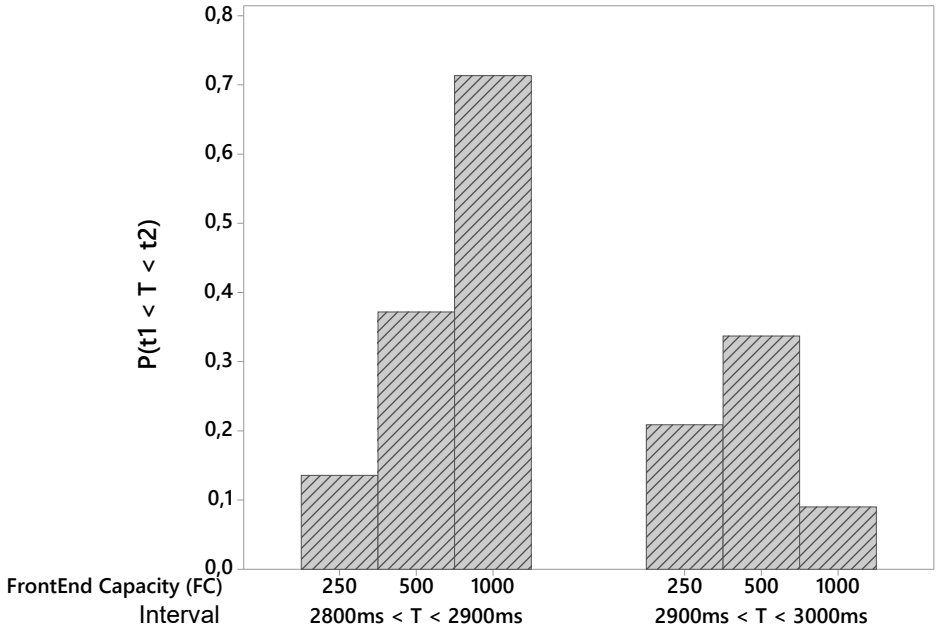

Figura 15: Probabilidade de Absorção com Base em Dois Intervalos de Tempo para o Estudo de Caso 2

Quando $FC = 250$, $FC = 500$ e $FC = 1000$, as APIs eram 12%, 37% e 70%, respectivamente. A API é bem diferente para esse primeiro intervalo, com uma variação de aproximadamente $\cong 28\%$ entre as três capacidades FrontEnd. Vale a pena mencionar que uma API = 70% é bastante significativa, mas alocar 1000 recursos pode ser muito caro.

No segundo intervalo, $(2900ms < T < 3000)$, há um padrão menos intuitivo, com $FC = 500$ (35%) superando os outros. Quando $FC = 500$, a API praticamente não mudou nos dois intervalos, significando que de 2800ms a 3000ms há uma certa constância (veja a Figura 15).

No segundo intervalo, quando $FC = 1000$, apresenta uma API menor que no primeiro intervalo, cerca de 9%. Interessante que mesmo com uma alta capacidade, a API é menor que as outras duas opções. Isso é verdade porque, a 2800ms, a API de $FC = 1000$ já é 90% e, em 2850ms, já atingiu 100%. Portanto, quando a API tende à probabilidade máxima, a API tende a zero.

7 Validação do Modelo SPN

Esta seção apresenta a validação do modelo SPN proposto. O objetivo desta validação foi comparar o MRT calculado pelo modelo e o MRT coletado por experimentos em um cenário real. A Figura 16 apresenta um esboço do experimento prático realizado. Assim como na arquitetura proposta neste trabalho, na validação simulamos a existência de três torres com um servidor cada. A capacidade que adotamos para todos os servidores foi de 4 núcleos, onde cada um deles roda um *container* diferente. Cada requisição é processada por apenas um *container*, portanto, se chegar 4 requisições em uma torre, a mesma irá distribuí-las entre todos os *containers*. Para realizar a validação, foi desenvolvido um sistema sintético que simula o envio de requisições obedecendo uma distribuição exponencial.

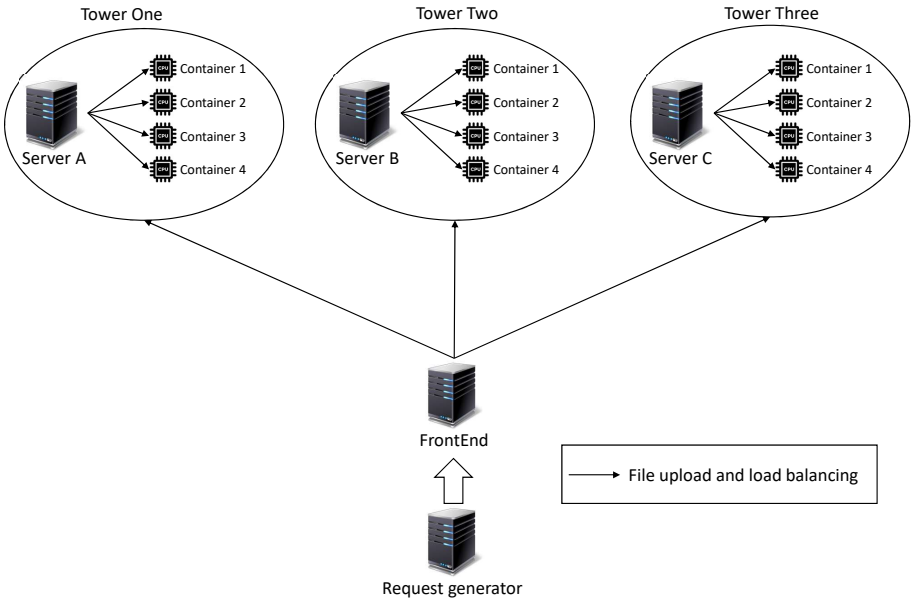

Figura 16: Esboço Prático da Experiência

A Figura 17 apresenta um diagrama de sequência referente ao experimento, com os passos para coletar os dados. As chamadas destacadas em azul se referem às funções principais referentes às requisições. As demais chamadas representam a coleta dos *timestamps* a um servidor local que atua como relógio global. Os tempos são coletados no início e fim de cada tarefa realizada, com o objetivo de saber a duração das etapas. Quando o arquivo de entrada é enviado, a máquina requisita ao servidor do relógio global o *timestamp* e salva em um log, quando o arquivo é recebido em outra máquina, esta realiza a mesma ação. Com esses dois *timestamps* obtemos o tempo necessário para envio de uma requisição. Os tempos obtidos com esse algoritmo foram utilizados para alimentar as transições do modelo SPN estendido, apresentado na seção 6. A partir desses valores, as transições

T_A, T_B e *T_C* foram preenchidas com os tempos de envio do FrontEnd para as torres A, B e C respectivamente. Nas transições *P_A, P_B* e *P_C*, adicionamos os tempos de processamento para cada servidor. A Tabela 8 apresenta a configuração dos computadores utilizados para realizar a validação juntamente com as suas respectivas configurações e funções.

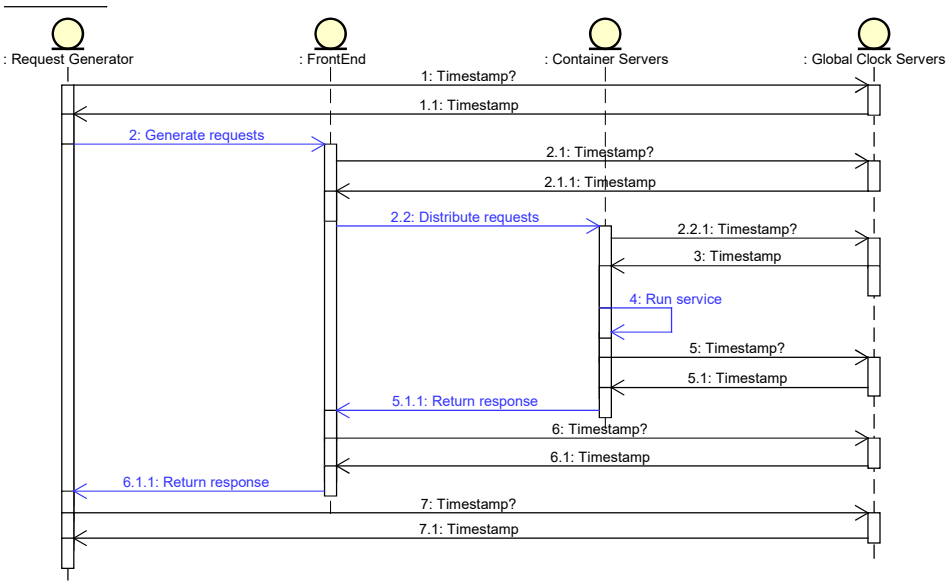

Figura 17: Diagrama de Sequência do Experimento Prático

Tabela 8: Máquinas Utilizadas no Experimento Prático

Configuração do Computador	Perfil	Função
Intel Core i3 1.7Ghz 8GB	Gerador de requisições	Gera solicitações para serem processadas nos servidores
Intel Core i3 3.7Ghz 4GB	FrontEnd	Recebe solicitações e encaminhe para servidores
Intel Core i3 3.7Ghz 4GB	Servidor A	Recebe as solicitações e as processa em seus núcleos
Intel Core i3 3.7Ghz 8GB	Servidor B	Recebe as solicitações e as processa em seus núcleos
Intel Core i3 3.7Ghz 8GB	Servidor C	Recebe as solicitações e as processa em seus núcleos

Cada requisição utilizada no experimento prático consiste em um arquivo contendo uma matriz de tamanho 10000x10000 preenchida com valores aleatórios, onde o mesmo é enviado para um dos servidores para ser ordenado e retornado para o gerador de requisições. O gerador de requisições simula uma quantidade x de clientes que enviam as solicitações ao mesmo tempo, o FrontEnd recebe todos de forma simultânea e os escalona entre as torres, essas por sua vez as escalona entre os servidores existentes. Com o objetivo de tornar o resultado mais confiável, para cada configuração dos servidores repetimos 15 vezes a execução do algoritmo. A Tabela 9 apresenta o MRT e as configurações das máquinas para as execuções realizadas, enquanto a Tabela 10 apresenta os MRTs obtidos nas execuções do modelo.

Note que quanto mais requisições forem enviadas, maior será o MRT. O algoritmo apresentou-se estável, pois os resultados não apresentaram valores muito distantes entre

Tabela 9: Tempos Obtidos no Experimento Prático

Id da Execução	MRT (ms)		
	3 Requisições	6 Requisições	9 Requisições
1	1380	2892	4395
2	1105	3302	3621
3	1376	2803	3830
4	1221	2139	4394
5	1013	2336	4819
6	1162	2081	4526
7	1098	2378	4412
8	1107	2725	3937
9	1015	2274	4567
10	1063	2263	4434
11	1008	2561	3928
12	995	2178	4370
13	1090	2496	4685
14	1068	2068	4418
15	1046	2071	4633

Tabela 10: Tempos Obtidos no Modelo SPN

Número de Requisições	MRT (ms)
3	1103
6	2504
9	4457.4

si. O Teste T de uma amostra [1] é usado para fazer inferências sobre a média da população, com base nos dados de uma amostra aleatória. Todas amostras utilizadas foram normais. Adotamos o Teste T de uma amostra para comparar o MRT gerado pelo modelo com a média do MRT obtido nas execuções. Para verificar se o Teste T é significante utilizamos o valor p, sendo ele uma probabilidade que mede a evidência contra a hipótese nula. Um valor de p menor fornece uma evidência mais forte contra a hipótese nula, sendo ela o MRT obtido na execução do modelo SPN, pois é o valor que queremos comparar. Geralmente, um nível de significância de 0,05 funciona bem, pois indica que o risco de se concluir que existe uma diferença, quando, na verdade, não existe nenhuma diferença real, é de 5%. A Tabela 11 apresenta os resultados obtidos com esse teste.

Tabela 11: Resultados Obtidos com o Teste T de Uma Amostra

Número de Requisições	Quantidade da Amostra	Média (ms)	Desvio Padrão	Valor P
3	15	1116.5	122.1	0.676
6	15	2437.8	360	0.488
9	15	4398	415	0.59

Observe que o desvio padrão aumenta à medida em que há uma maior quantidade de requisições, isso ocorre porque além do algoritmo ter mais requisições para processar, ainda terá um trabalho maior para escaloná-las. Observe também que em todos os casos, o valor p é maior que 0.05 e, portanto, não podemos refutar a hipótese nula em todos os casos com 95% de confiança. Os resultados gerados pelo modelo são equivalentes estatisticamente ao experimento. As Figuras 18, 19 e 20 apresentam os gráficos boxplot referentes aos valores p de cada cenário.

[1] Teste T de uma amostra https://tinyurl.com/yanthw4e

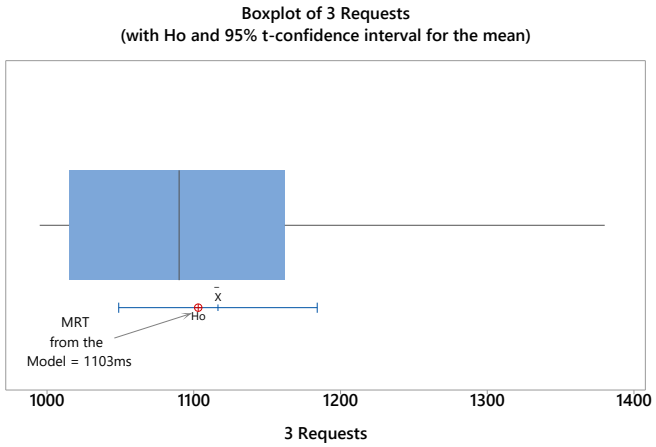

Figura 18: Boxplot das 3 Solicitações

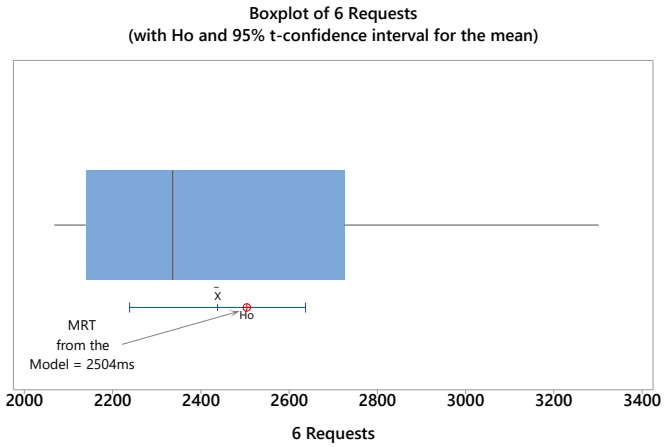

Figura 19: Boxplot das 6 Solicitações

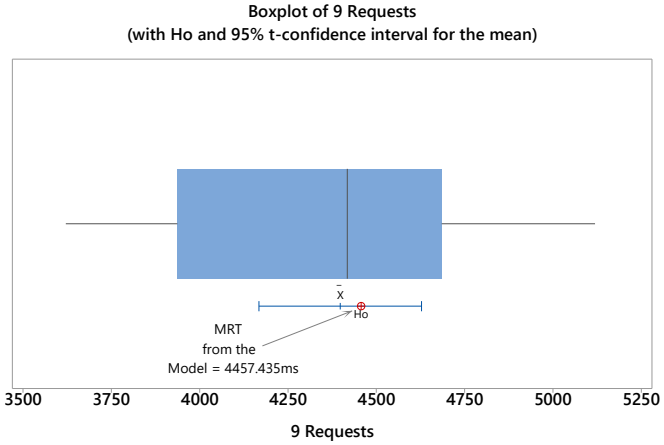

Figura 20: Boxplot das 9 Solicitações

Pode-se concluir que há indícios que o modelo é confiável e preciso dada a similaridade dos resultados obtidos com o experimento real. Vale observar também que o valor do modelo está dentro da margem de erro do experimento. O modelo reflete o ambiente real

e é útil para o planejamento de arquiteturas MEC.

8 Conclusões e Trabalhos Futuros

Este trabalho propôs um modelo SPN para representar e avaliar o desempenho de uma arquitetura MEC composta por servidores n. Nosso modelo permite estimar o tempo médio de resposta (MRT) e o nível de utilização de recursos na borda da rede. Um dos aspectos mais importantes do nosso modelo é a sua flexibilidade de avaliação devido ao alto número de parâmetros que podem ser configurados pelo usuário (15 parâmetros), tornando os testes mais efetivos e decisivos. Para demonstrar a aplicabilidade do modelo proposto, foram realizadas análises numéricas com dados reais coletados de um trabalho de referência. Através das análises numéricas, pudemos observar o comportamento do MRT e a utilização dos recursos, demonstrando a utilidade do modelo na escolha da melhor maneira de implementar uma arquitetura MEC.

Como trabalho futuro, planejamos estender o modelo de SPN para incluir métricas de disponibilidade e medir energia. Dado o modelo estendido, planejamos realizar novas análises numéricas usando cenários heterogêneos.

9 Publicações

Brena Santos, Daniel Carvalho, Iure Fé, Francisco Airton Silva. Avaliação de Desempenho de Computação Móvel na Borda Usando Redes de Petri Estocásticas. **Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance_CSBC)**, Belém. Pará. 2019.

Referências

ABBAS, N. et al. Mobile edge computing: A survey. *IEEE Internet of Things Journal*, IEEE, v. 5, n. 1, p. 450–465, 2017. Citado na página 18.

BADRI, H. et al. Multi-stage stochastic programming for service placement in edge computing systems: poster. In: ACM. *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. [S.l.], 2017. p. 28. Citado na página 21.

BECK, M. T. et al. Mobile edge computing: A taxonomy. In: CITESEER. *Proc. of the Sixth International Conference on Advances in Future Internet*. [S.l.], 2014. p. 48–55. Citado na página 12.

BORGIA, E. et al. Mobile edge clouds for information-centric iot services. In: IEEE. *2016 IEEE symposium on computers and communication (ISCC)*. [S.l.], 2016. p. 422–428. Citado na página 12.

CAU, E. et al. Efficient exploitation of mobile edge computing for virtualized 5g in epc architectures. In: IEEE. *2016 4th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud)*. [S.l.], 2016. p. 100–109. Citado na página 12.

CHEN, X. et al. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, IEEE, v. 24, n. 5, p. 2795–2808, 2016. Citado 2 vezes nas páginas 18 e 20.

CORCORAN, P.; DATTA, S. K. Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network. *IEEE Consumer Electronics Magazine*, IEEE, v. 5, n. 4, p. 73–74, 2016. Citado na página 18.

DESROCHERS, A.; AL-JAAR, R.; SOCIETY, I. C. S. *Applications of petri nets in manufacturing systems: modeling, control, and performance analysis*. [S.l.]: IEEE Press, 1995. ISBN 9780879422950. Citado na página 13.

DINH, H. T. et al. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, Wiley Online Library, v. 13, n. 18, p. 1587–1611, 2013. Citado na página 12.

GUPTA, L.; JAIN, R.; CHAN, H. A. Mobile edge computing–an important ingredient of 5g networks. *IEEE Software Defined Networks Newsletter*, 2016. Citado na página 18.

HU, Y. C. et al. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, v. 11, n. 11, p. 1–16, 2015. Citado na página 17.

JARARWEH, Y. et al. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In: IEEE. *2016 23rd International conference on telecommunications (ICT)*. [S.l.], 2016. p. 1–5. Citado 3 vezes nas páginas 12, 17 e 20.

JARARWEH, Y. et al. Sdmec: Software defined system for mobile edge computing. In: IEEE. *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. [S.l.], 2016. p. 88–93. Citado na página 12.

JUNG, H. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper*. [S.l.], 2016. Disponível em: <http://www.cisco.com/c/en/us-/solutions/collateral/service-provider/visual-networking-indexvni/mobile-white-paper-c11-520862.html>. Citado na página 12.

KE, M. Y. Z.; S, Z. Q. L.; L, P. X. L. Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. v. 4, p. 5896–5907, 2016. Citado na página 20.

KITANOV, S.; MONTEIRO, E.; JANEVSKI, T. 5g and the fog—survey of related technologies and research directions. In: IEEE. *2016 18th Mediterranean Electrotechnical Conference (MELECON)*. [S.l.], 2016. p. 1–6. Citado na página 12.

LI, J. W. G.; WU, J. S. J. Data processing delay optimization in mobile edge computing. v. 2018, 2018. Citado 2 vezes nas páginas 18 e 20.

LITTLE, J. D. A proof for the queuing formula: L= λ w. *Operations research*, INFORMS, v. 9, n. 3, p. 383–387, 1961. Citado na página 26.

LIU, J. et al. Delay-optimal computation task scheduling for mobile-edge computing systems. In: IEEE. *2016 IEEE International Symposium on Information Theory (ISIT)*. [S.l.], 2016. p. 1451–1455. Citado na página 21.

MAO, Y.; ZHANG, J.; LETAIEF, K. B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 34, n. 12, p. 3590–3605, 2016. Citado na página 20.

MAROTTA, M. A. et al. Managing mobile cloud computing considering objective and subjective perspectives. *Computer Networks*, Elsevier, v. 93, p. 531–542, 2015. Citado na página 12.

MOBILE edge computing: Um levantamento sobre arquitetura e computação offloading. *IEEE Communications Surveys and Tutorials*, IEEE, v. 19, n. 3. Citado na página 17.

OLIVEIRA, D. et al. Advanced stochastic petri net modeling with the mercury scripting language. In: ACM. *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. [S.l.], 2017. p. 192–197. Citado na página 15.

ORSINI, G.; BADE, D.; LAMERSDORF, W. Computing at the mobile edge: Designing elastic android applications for computation offloading. In: IEEE. *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*. [S.l.], 2015. p. 112–119. Citado na página 12.

PINHEIRO, T. et al. Performance and data traffic analysis of mobile cloud environments. In: IEEE. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2018. p. 4100–4105. Citado na página 13.

PINHEIRO, T. F. da S. et al. Performance prediction for supporting mobile applications' offloading. *The Journal of Supercomputing*, Springer, v. 74, n. 8, p. 4060–4103, 2018. Citado na página 13.

PREMSANKAR, M. d. F. G.; TALEB, T. Edge computing for the internet of things: A case study. v. 5, p. 1275–1284, 2018. Citado 4 vezes nas páginas 18, 20, 21 e 28.

RAHMANI, A. M. et al. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, Elsevier, v. 78, p. 641–658, 2018. Citado na página 22.

SATRIA, D.; PARK, D.; JO, M. Recovery for overloaded mobile edge computing. *Future Generation Computer Systems*, Elsevier, v. 70, p. 138–147, 2017. Citado na página 17.

SATRIA, D.; PARK, D.; JO, M. Recovery for overloaded mobile edge computing. *Future Generation Comp. Syst.*, v. 70, p. 138–147, 2017. Citado na página 18.

SILVA, F. A. et al. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, IEEE, v. 17, n. 5, p. 1134–1147, 2018. Citado 2 vezes nas páginas 13 e 33.

Sucipto, K. et al. Keep your nice friends close, but your rich friends closer — computation offloading using nfc. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2017. p. 1–9. Citado na página 12.

TONG, L.; LI, Y.; GAO, W. A hierarchical edge cloud architecture for mobile computing. In: IEEE. *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. [S.l.], 2016. p. 1–9. Citado na página 20.

TRAN, T. X. et al. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *arXiv preprint arXiv:1612.03184*, 2016. Citado na página 18.

Tran, T. X. et al. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, v. 55, n. 4, p. 54–61, April 2017. ISSN 0163-6804. Citado na página 12.

TRINH, C.; YAO, L. Energy-aware mobile edge computing for low-latency visual data processing. p. 128–133, 2017. Citado na página 20.

ZHANG, X. et al. Improving cloud gaming experience through mobile edge computing. *IEEE Wireless Communications*, IEEE, 2019. Citado na página 22.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, **Daniel de Carvalho Borges**, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação **Alocação de Servidores em Computação na Borda Móvel Usando Redes de Petri Estocásticas** de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Digite o te

Picos-PI 19 de maio de 2021.

Daniel de Carvalho Borges

Assinatura

Assinatura