Pós-Graduação em Ciência da Computação

Bruno Roberto Silva

Seleção de rede e de recursos computacionais para o *offloading* em ambiente de nuvem móvel

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

RECIFE

2018

Bruno Roberto Silva

Seleção de rede e de recursos computacionais para o *offloading* em ambiente de nuvem móvel

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Kelvin Lopes Dias

RECIFE
2018

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

S586s Silva, Bruno Roberto
Seleção de rede e de recursos computacionais para o *offloading* em ambiente de nuvem móvel / Bruno Roberto Silva. – 2018.
97 f.: il., fig., tab.

Orientador: Kelvin Lopes Dias.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2018.
Inclui referências.

1. Redes de computadores. 2. Computação em nuvem. I. Dias, Kelvin Lopes (orientador). II. Título.

004.6 CDD (23. ed.) UFPE- MEI 2018-080

Dissertação de Mestrado apresentada por **Bruno Roberto Silva** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Seleção de rede e de recursos computacionais para o *offloading* em ambiente de nuvem móvel**" **Orientador: Kelvin Lopes Dias** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Dr. José Augusto Suruagy Monteiro
Centro de Informática / UFPE

Prof. Dr. Christian Esteve Rothenberg
Departamento de Engenharia de Computação e Automação Industrial / UNICAMP

Prof. Dr. Kelvin Lopes Dias (Orientador)
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 08 de Março de 2018.

Prof. Aluizio Fausto Ribeiro Araújo
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedico esta dissertação ao Rei dos Reis, Jesus Cristo.
Maranata!

AGRADECIMENTOS

Sou grato primeiramente a Deus, por me dar o dom da vida e a saúde para concluir este trabalho.

Ao meu orientador, professor Kelvin Lopes Dias pela oportunidade, ajuda e apoio que possibilitou a realização do sonho de concluir este mestrado.

Um agradecimento especial ao meu amigo Warley Junior, por ter compartilhado seu conhecimento e pelo incentivo durante todo desenvolvimento desta pesquisa.

Agradeço à minha família, minha mãe Maria Roberto, minha irmã Vânia Freitas e meu cunhado Antônio Nascimento, que sempre me deram apoio, muito amor e carinho em tudo que fiz. Ao meu pai José Silva (*in memoriam*), pelo seu exemplo de pai, que apesar de não estar mais presente, sempre estará no meu coração.

Aos amigos e companheiros do grupo de pesquisa IANG: Atrícia Sabino, Alexsanderson Vieira, Edigleison Barbosa, Francisco Junior e Rhodney Simões. Também aos amigos Eduardo Maia e Albertinin Mourato. Obrigado a todos pelos ótimos momentos e experiências que levarei por toda vida.

Ao meu amigo Adriano Henrique, por todo seu incentivo, principalmente no início do mestrado.

Muito obrigado pela ajuda de todos, pois sem esse apoio não seria capaz de chegar onde cheguei.

RESUMO

A computação em nuvem móvel (MCC) permite que os *smartphones*, com limitações em termos de processamento, armazenamento e de tempo de vida de bateria, se comparados aos servidores de rede, executem aplicações que demandam cada vez mais recursos computacionais. Essas limitações são aliviadas utilizando-se, principalmente, a técnica de *offloading* computacional, que permite enviar e receber dados processados remotamente, permitindo assim, reduzir o consumo de recursos no dispositivo móvel. O *offloading* pode ser executado também em *cloudlets*, que disponibilizam recursos de computação em ambientes virtualizados via rede local Wi-Fi, com vazão elevada e atrasos reduzidos se comparados aos acessos via redes celulares tradicionais e a nuvens públicas. Contudo, mesmo utilizando *cloudlets*, a experiência de *offloading* do usuário pode ser afetada devido às características da mobilidade e degradação do sinal, além de mudanças na carga da rede e de processamento na VM (*Virtual Machine*). Esta dissertação propõe um sistema de seleção de rede sem fio e VM na *cloudlet* para a execução do *offloading* computacional. O algoritmo de decisão considera os requisitos de qualidade de serviço (QoS) da aplicação, nível de sinal, vazão das redes candidatas, tempo de resposta, nível de utilização da CPU e memória da nuvem. A abordagem utiliza duas estratégias: a primeira é um sistema *fuzzy* que utiliza o motor de inferência para decisões de execução de *handoff*; a segunda, calcula o custo de *offloading* para seleção de VM, com pesos dos critérios gerados pelo método AHP (*Analytic Hierarchy Process*). Como estratégia de conectividade MCC, a proposta utiliza o paradigma das redes definidas por software (SDN) para o encaminhamento de pacotes entre o dispositivo e a *cloudlet*, via protocolo *OpenFlow*, evitando sinalizações extras na rede e reduzindo o tempo de reconexão. Além de redirecionar os fluxos de acordo com o novo ponto de acesso e VM selecionados, a aplicação SDN também é responsável pela coleta de informações e por processar as etapas do algoritmo de decisão no controlador da rede. Para avaliar o sistema proposto nesta dissertação, foi preparado um *testbed* *OpenFlow*/Wi-Fi com o controlador Ryu. Duas aplicações foram consideradas: a primeira realiza reconhecimento facial e foi avaliada com base na métrica tempo de execução. A segunda aplicação, de processamento em tempo real, considera o número de *frames* renderizardos por segundo, como métrica. Com base nos cenários avaliados, os resultados indicam um ganho no desempenho de *offloading* quando utilizada a solução proposta de até 48,33% para a primeira aplicação e 27,52% para a segunda.

Palavras-chaves: Computação em Nuvem Móvel. *Offloading* computacional. *Cloudlet*. sistema *Fuzzy*. Redes Definidas por *Software*. *OpenFlow*.

ABSTRACT

Mobile cloud computing (MCC) enables smartphones to run resource intensive applications on powerful servers located in remote cloud computing environment through computational offloading technique. Recently, computing resources in virtualized environments on Wi-Fi network named Cloudlets have also been used for offloading purposes. Cloudlets provide higher throughputs and reduced delays when compared to traditional cellular access to remote public clouds. However, even using cloudlets, the user's offloading experience may be degraded by mobility and signal fading. Besides that, changes in network load and virtual machine (VM) processing of the cloud infrastructure may also impact the quality of service (QoS) of applications during the offloading process. This dissertation proposes a system for selecting the appropriate access point and VM belonging to a cloudlet in order to execute computational offloading. The decision algorithm considers the application's QoS requirements, signal level of access points, throughput of candidate networks, response time, CPU utilization level, and VM memory. To this end, the proposal adopts two strategies: a fuzzy system which uses an inference engine for handff decision, and a cost function for VM selection, whose weights are defined by the AHP (Analytic Hierarchy Process) method. Software-Defined Networking (SDN) paradigm is used as the connectivity management approach for packet forwarding between the device and the cloud, avoiding extra signaling on the network and reducing reconnection time. Besides redirecting flows according to the new selected access point and VM, the SDN application is also responsible for collecting information and processing the decision algorithm stages in the network controller. To evaluate the proposal, an OpenFlow/Wi-Fi testbed was devised with the Ryu controller. Two applications were considered: face recognition which was evaluated based on the execution time metric, and a real-time processing application considering the number of rendering frames per second as the metric. Based on the evaluated scenarios, the results indicate a gain in offloading performance when the proposed solution was compared to baseline schemes with values achieving up to 48.33% for the first application, and 27.52% for the second one, respectively.

Key-words: Mobile Cloud Computing. Computational offloading. Cloudlet. Fuzzy system. Software Defined Networks. OpenFlow.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

3G *Third Generation.*

4G *Fourth Generation.*

ABC *Always Best Connected.*

AHP *Analytic Hierarchy Process.*

AP *Access Point.*

API *Application Programming Interface.*

CC *Connecting Cloudlet.*

CR *Consistency Ratio.*

CRS *Cloud Ranking Service.*

ENDA *Embracing Network Inconsistency for Dynamic Application.*

FIS *Fuzzy Inference System.*

GPRS *General Packet Radio Service.*

GPS *Global Positioning System.*

GUI *Graphical User Interface.*

ICMP *Internet Control Message Protocol.*

ICN *Information-Centric Networking.*

IP *Internet Protocol.*

IPv6 *Internet Protocol version 6.*

ISP *Internet Service Provider.*

JVM *Java Virtual Machine.*

LAN *Local Area Network.*

LTE *Long Term Evolution.*

M-MIP *Multi-homed Mobile IP.*

MAC *Media Access Control.*

MCC *Mobile Cloud Computing.*

MCDM *Multiple-Criteria Decision-Making.*

MEC *Mobile Edge Computing.*

MHS *Mobile Healthcare System.*

MOSys *Mobile Offloading System.*

MpOS *Multiplatform Offloading System.*

MSN *Mobile Social Networks.*

NAT *Network Address Translation.*

NDP *Neighbor Discovery Protocol.*

NETCONF *Network Configuration Protocol.*

NFC *Near Field Communication.*

OF-Config *OpenFlow Management e Configuration Protocol.*

OFLMA *OpenFlow Local Mobility Anchor.*

OFMAG *OpenFlow Mobility Access Gateway.*

ONF *Open Networking Foundation.*

PFS *Proxy Finder Server.*

PMIPv6 *Proxy Mobile IPv6.*

QoS *Quality of Service.*

RNL *Relative Network Load.*

RSSI *Received Signal Strength Indicator.*

RTT *Round Trip Time.*

SC *Serving Cloudlet.*

SDN *Software-Defined Networking.*

SSID *Service Set Identifier.*

TOPSIS *Technique for Order of Preference by Similarity to Ideal Solution.*

UMTS *Universal Mobile Telecommunication System.*

VANETS *Vehicular Ad-hoc Networks.*

VM *Virtual Machine.*

SUMÁRIO

1 INTRODUÇÃO

Este Capítulo descreve a motivação, introduzindo conceitos na área de *Mobile Cloud Computing* (MCC), tecnologias relacionadas e desafios que concernem à experiência dos dispositivos móveis nesses ambientes. Em seguida, são apresentados o objetivo geral e os objetivos específicos, bem como a solução proposta e, por fim, a estrutura da pesquisa desenvolvida nesta dissertação.

1.1 Motivação

O rápido crescimento do número de dispositivos móveis como *smartphones*, está revolucionando muitos aspectos em nossas vidas. Projeções indicam que a quantidade destes dispositivos conectados até 2021 será de 11,6 bilhões - superando a previsão da população mundial deste período, estimada em 7,8 bilhões (CISCO, 2017). Este fenômeno se deve principalmente aos benefícios que a Computação Móvel oferece como, por exemplo, a liberdade que os usuários necessitam de acessar informações através das redes sem fio em qualquer lugar e a qualquer momento (SAMAD; LOKE; REED, 2015).

A popularização desta nova tendência dos usuários, permitiu que os recursos dos dispositivos móveis (por exemplo, capacidade de processamento, quantidade de memória e compatibilidade entre tecnologias sem fio) fossem sendo cada vez mais aprimorados. Este fato, proporcionou a criação de uma gama de aplicações móveis cada vez mais robustas. Graças ao desenvolvimento do paradigma MCC, foi possível continuar a suprir estas demandas, o que permite aos dispositivos móveis executarem aplicações com requisitos de processamento cada vez mais altos (BAHTOVSKI; GUSEV, 2014).

A principal característica da MCC é possibilitar a inserção dos benefícios da Computação em Nuvem como, por exemplo, a alta capacidade de processamento e armazenamento, para dentro do ecossistema móvel. Esses méritos também trouxeram grandes avanços na maneira como as aplicações móveis acessam e processam seus serviços, onde nesse contexto, começaram a realizar a migração de seus dados para serem processados na nuvem. Esta técnica é conhecida como *offloading* computacional e oferece inúmeras vantagens para aplicações de diversas áreas (por exemplo, processamento de imagens, realidade virtual e aumentada, *healthcare* e jogos), visando tornar o tempo de resposta das solicitações do usuário mais rápidas e maximizando a vida útil da bateria do dispositivo móvel (KUMAR; LU, 2010). Na prática, aplicações como, por exemplo, de *healthcare*, podem utilizar a conectividade sem fio do *smartphone* para enviar informações de sinais vitais humanos (capturados através de sensores vestíveis), para serem processados na nuvem de modo contínuo, enquanto o usuário está se movendo em um carro.

Como um dos principais aspectos dos dispositivos móveis é a mobilidade, a execução de *handoffs* (mudança entre pontos de acesso sem fio) é uma questão crítica dentro de ambientes MCC devido à escassez de recursos, energia finita e baixa conectividade sem fio (LI et al., 2015). Quando um usuário se conecta a um *Access Point* (AP) para realizar o *offloading*, sua experiência na execução da aplicação pode ser afetada, seja pela perda de conectividade ou pela degradação do nível de sinal. Como é comum que o usuário se encontre durante o dia em vários ambientes com inúmeras redes disponíveis para a execução do *offloading*, devem ser definidas estratégias para determinar quais delas possuem os requisitos adequados. Dessa forma, tomadas de decisão que não considerem informações importantes relacionadas à rede, bem como ao ambiente de processamento e armazenamento, podem afetar a qualidade do acesso ao sistema da nuvem e causar erros de conexão.

Segundo Ong e Khan (2008), a decisão de *handoff* é a chave para garantir que os dispositivos móveis permaneçam conectados às melhores redes sem fio disponíveis, adotando assim o conceito denominado de *Always Best Connected* (ABC) (GUSTAFSSON; JONSSON, 2003). Há muitos algoritmos de *handoff* que utilizam apenas o nível de sinal para esta decisão, causando muitas vezes o efeito *ping-pong*. Outras soluções usam parâmetros mais eficazes como, por exemplo, a largura de banda disponível na rede sem fio, tempo de conexão de rede, custo monetário e a preferência do usuário móvel (YAN; ŞEKERCIOğLU; NARAYANAN, 2010). No entanto, essas técnicas tradicionais não tratam aspectos específicos relacionados à mobilidade em MCC.

Como os requisitos de execução das aplicações do ambiente MCC não são restritos apenas aos recursos da rede sem fio, é preciso considerar outros parâmetros como o estado da nuvem da qual o usuário está obtendo serviços de processamento e/ou armazenamento e as características do aplicativo móvel em execução. Em relação aos parâmetro da nuvem, leva-se em conta o estado da *Virtual Machine* (VM) que o dispositivo está conectado responsável por atender suas demandas. Manter-se conectado a uma VM com, por exemplo, um alto nível de utilização da CPU, pode causar a degradação do desempenho da aplicação devido ao aumento do tempo de resposta, inviabilizando assim a execução remota. Dessa forma, para a execução do *offloading* é preciso que o dispositivo esteja sempre conectado a uma VM ideal para mitigar estes impactos (ZHOU et al., 2015). Além disso, cada tipo de aplicação possui requisitos de *Quality of Service* (QoS) distintos, sendo necessário ter ciência de cada um deles para obter sucesso na seleção de rede e recursos de nuvem. Enquanto algumas delas são mais sensíveis a variações da rede sem fio como largura de banda insuficiente, causando alta porcentagem de perda de pacotes, outras são mais impactadas pelo estado da nuvem como indisponibilidade do serviço, causando atrasos no processamento dos dados transmitidos.

Algumas técnicas já existentes procuram atender a estes requisitos que, além dos tradicionais parâmetros da rede sem fio para decisão de *offloading*, consideram também

recursos de nuvem variados como *cloudlets*, nuvens públicas e ad-hoc. Além disso, também existem técnicas de conectividade em MCC que principalmente pelo aspecto da heterogeneidade das redes, focam em prover o *offloading* transparente, visando evitar interrupções durante a execução das aplicações (GANI et al., 2014). No entanto, mesmo com as pesquisas já realizadas, a falta de flexibilidade, soluções programáveis e visão holística dos vários elementos envolvidos nas operações de *offloading*, reforçam a necessidade do desenvolvimento de soluções mais efetivas (AHMED et al., 2015).

Recentemente, com o surgimento do paradigma *Software-Defined Networking* (SDN) (KREUTZ et al., 2015), os problemas no âmbito MCC, citados anteriormente, podem ser atacados com soluções inovadoras, flexíveis e programáveis. Diferente das redes tradicionais, com SDN é possível ter uma visão holística através de um controlador centralizado, tornando os dispositivos (por exemplo, *switches* e roteadores) programáveis, oferecendo assim, maior flexibilidade à infraestrutura da MCC. Diante destas características, é possível usufruir de uma estratégia de conectividade mais eficaz do usuário com a nuvem, através do protocolo *OpenFlow* que permite o gerenciamento do encaminhamento de pacotes, sem a necessidade de realizar alterações do lado do dispositivo. Como benefício do uso desta funcionalidade em MCC, destaca-se o redirecionamento do resultado de *offloading* durante a mobilidade do usuário, possibilitando que o mesmo usufrua dos serviços de nuvem com maior qualidade (JUNIOR et al., 2017).

Alinhadas com as estratégias de conectividade, mecanismos de tomada de decisão para seleção de rede e recursos de nuvem são essenciais para serem utilizadas no ambiente MCC. Conforme já enfatizado, para proporcionar uma melhor experiência do usuário com suas aplicações durante as execuções de *offloading*, um conjunto de variáveis devem ser levadas em consideração para que se obtenham decisões mais eficazes. Para isso, a técnica *Multiple-Criteria Decision-Making* (MCDM) fornece soluções que atendem a estas necessidades, pois conseguem suportar critérios distintos que permitem selecionar a melhor rede e recursos de nuvem. Dentre os mecanismos existentes, aqui se destaca a Lógica *Fuzzy* que possui abordagem e conceitos intuitivos, além de possuir uma estrutura flexível que permite definir funções de pertinência de acordo com cada critério utilizado na decisão, com base na experiência do especialista (PERERA et al., 2014).

Com base nestes aspectos, fica evidente a necessidade de desenvolver técnicas e mecanismos que preservem a potencialidade da MCC. Este trabalho propõe e avalia um sistema de decisão, que visa prover ao usuário o acesso aos melhores recursos de rede e nuvem disponíveis para execução do *offloading* computacional. Muitos trabalhos utilizam esquemas baseados em *proxy* ou utilizam múltiplos critérios somente para decisão de *handoff*, já outros também consideram o estado da nuvem utilizando estratégias de ranqueamento para seleção de recursos remotos (JUNIOR; SILVA; DIAS, 2018). O sistema proposto tem por objetivo principal ser ciente da aplicação, de forma que a seleção da rede sem fio e VM na *cloudlet* seja realizada de acordo com suas exigências de QoS. Este trabalho se diferen-

cia dos outros da literatura, por considerar o tipo da aplicação e tamanho dos dados de *offloading* como critérios de decisão, variando as métricas utilizadas para prover decisões mais adaptáveis. Como a integração com outras tecnologias, paradigmas e mecanismos de tomada de decisão estão cada vez mais presentes na infraestrutura MCC, estas podem auxiliar no acesso aos serviços distribuídos, dentre os mais diversos cenários existentes no ambiente de nuvem móvel.

1.2 Objetivo Geral

O objetivo desta dissertação é propor um sistema de seleção de redes sem fio e VMs na *cloudlet* com base nos requisitos de QoS da aplicação móvel, visando fornecer ao usuário o acesso aos melhores recursos disponíveis e o menor tempo de execução de *offloading* possível.

1.3 Objetivos Específicos

- Desenvolver um sistema para tomada de decisão de *handoff* com base no tipo da aplicação móvel.
- Elaborar uma função de custo de *offloading* para utilizar como estratégia de seleção de VMs na *cloudlet*, variando os critérios de acordo com o tipo da aplicação móvel.
- Propor e implementar um gerenciador de fluxo com objetivo de gerenciar as etapas de decisão do sistema proposto.
- Definir uma estratégia de conectividade que permita o encaminhamento de pacotes sem degradar a experiência do usuário.

1.4 Estrutura da Dissertação

Esta dissertação contém 6 Capítulos organizados da seguinte forma:

O Capítulo 2 apresenta os conceitos básicos do paradigma da Computação em Nuvem Móvel, a técnica de *offloading* computacional e a arquitetura geral do ambiente MCC. Fundamentos básicos sobre o paradigma das redes definidas por *software*, do protocolo *OpenFlow* e os controladores de rede existentes. Por fim, é apresentada a lógica *fuzzy*, incluindo o conceito sobre a teoria dos conjuntos nebulosos, as características das funções de pertinência e componentes deste sistema de tomada de decisão.

O Capítulo 3 apresenta e discute os trabalhos relacionados que inspiraram o tema proposto por esta dissertação.

O Capítulo 4 descreve a proposta desta dissertação, os elementos do sistema de decisão, detalhes da sinalização da solução e a apresentação do pseudocódigo do algoritmo de decisão.

O Capítulo 5 descreve as avaliações realizadas, detalhando o ambiente de teste e cenários definidos em cada experimento e os resultados obtidos com a utilização do sistema proposto.

Posteriormente, o Capítulo 6 contém as conclusões finais desta dissertação, contribuições e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, as tecnologias que deram base ao desenvolvimento da proposta desta dissertação são apresentadas. Está dividido em três subseções. A Seção 2.1 aborda o paradigma MCC e seus componentes essenciais para caracterizar o ambiente da proposta. A Seção 2.2 aborda o paradigma SDN que é utilizado neste trabalho no desenvolvimento de parte do sistema de decisão. Na Seção 2.3 é apresentada a Lógica *Fuzzy* no intuito de introduzir o conhecimento básico para uso desta técnica que é utilizada para tomada de decisão de *handoff* deste trabalho.

2.1 Computação em Nuvem Móvel

A Computação em Nuvem, superou limitações no modelo da computação tradicional como a baixa capacidade de processamento e armazenamento de computadores *desktops*. Esta evolução, deu origem ao paradigma denominado *Mobile Cloud Computing* (MCC), permitindo avanços no modo de como os dispositivos móveis acessam e processam seus serviços. A seguir, serão abordados os principais conceitos e técnicas existentes em MCC que foram utilizados neste trabalho.

2.1.1 Fundamentos básicos

Devido à crescente demanda dos usuário móveis de acessar informações em qualquer lugar e a qualquer momento, foram desenvolvidas novas tecnologias, possibilitando o desenvolvimento de aplicações mais sofisticadas. No entanto, os dispositivos móveis possuem limitações como poder de processamento, tempo de vida da bateria e capacidade de memória. A MCC combina as qualidades da Computação Móvel e Computação em Nuvem, permitindo potencializar os benefícios de cada um deles para atenuar as limitações de recursos dos dispositivos móveis (KHAN et al., 2014).

A Computação em Nuvem fornece um *pool* de recursos virtualizados através de um *hypervisor* que mascara a complexidade do acesso aos recursos físicos, permitindo prover estes recursos de forma elástica (ARMBRUST et al., 2010). A Computação Móvel que passa por crescentes desafios de demanda de recursos pelos serviços móveis, visa permitir o acesso aos serviços computacionais em movimento (SATYANARAYANAN, 1996). Como resultado, o MCC se apresenta como alternativa para permitir o acesso da computação móvel a recursos potencialmente ilimitados, estendendo a visão computacional sob demanda da Computação em Nuvem.

Autores como Dinh et al. (2013), descrevem MCC como um novo paradigma para aplicações móveis onde o processamento e armazenamento dos dados são transferidos através

da conexão sem fio, do dispositivo móvel para um ambiente centralizado de alto poder computacional, localizado na nuvem. Dessa maneira, para a execução de certas aplicações, os dispositivos móveis não necessitam de uma configuração de *hardware* poderosa, pois toda complexidade e módulos de processamento podem ser acessados na nuvem.

Devido às vantagens únicas trazidas por esta sinergia, foi possível que uma ampla gama de aplicações móveis fossem desenvolvidas nas mais diversas áreas, por exemplo, processamento de imagem, compartilhamento de *Global Positioning System* (GPS), aplicativos de dados de sensores, *crowd computing*, transmissão de vídeo, jogos online, comércio eletrônico, reconhecimento facial, realidade aumentada, realidade virtual e redes sociais, tornando os dispositivos móveis uma parte importante da vida cotidiana (FERNANDO; LOKE; RAHAYU, 2013).

A MCC também inspirou o modelo das *cloudlets*, fornecendo em menor escala, vários serviços semelhantes à nuvem, sobre uma infraestrutura de *workstations/clusters* e uma camada de virtualização na mesma rede local sem fio do usuário. Seu objetivo é oferecer uma melhor qualidade de serviço através das redes WiFi por serem menos congestionadas (ou seja, têm velocidades mais altas e menor latência) do que as redes móveis (por exemplo, *Third Generation* (3G) e *Fourth Generation* (4G)) (SATYANARAYANAN et al., 2009).

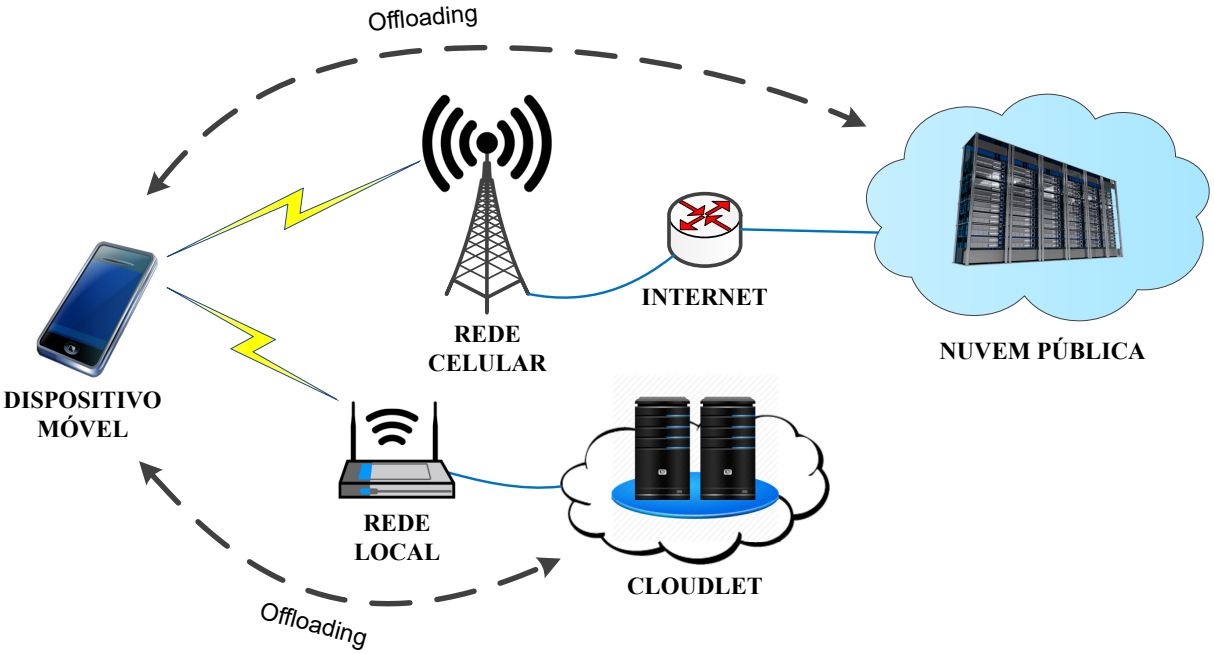
2.1.2 *Offloading* Computacional

Ao contrário da MCC que é um paradigma recente, o conceito de *offloading* não é novo na literatura. Segundo Ma et al. (2012), pesquisadores desde a década de 70 já propuseram conceitos similares como, por exemplo, o balanceamento de carga em sistemas distribuídos. Ainda nos dias atuais, esta linha de pesquisa existe de várias formas, como no trabalho de Ha et al. (2013), que com o objetivo de aumentar as capacidades computacionais e de armazenamento de dispositivos móveis através da distribuição de tarefas, apresenta o conceito de *cyber foraging*.

Em MCC, se convencionou usar o termo *offloading* computacional para a técnica que migra um intenso volume de dados do dispositivo móvel, para serem processados na nuvem ou em servidores ricos de recursos computacionais (ENZAI; TANG, 2014). De acordo com Verbelen et al. (2012), o *offloading* computacional pode ser executado em ambientes remotos como máquinas virtuais na nuvem, onde o poder de computação é oferecido sob demanda (por exemplo, nuvens públicas), bem como máquinas que estejam na mesma *Local Area Network* (LAN) dos dispositivos móveis (por exemplo, *cloudlets*). A Figura 1 apresenta exemplos de ambientes de execução do *offloading* computacional.

Vale ressaltar que o *offloading* computacional é diferente do tradicional paradigma cliente-servidor onde ocorre a migração para um servidor remoto, pois o conceito de nuvem implica em serviços, armazenamento de dados e outros recursos que são hospedados remotamente. O *offloading* computacional também é diferente do modelo *Grid Computing* que além de ter fins de balanceamento, tem como principal diferença que a migração nor-

Figura 1 – Exemplos de ambientes de execução do *offloading* computacional



Fonte: Elaborada pelo Autor

malmente ocorre de um computador para outro dentro do mesmo ambiente de computação (a grade) e não fora (KUMAR et al., 2013).

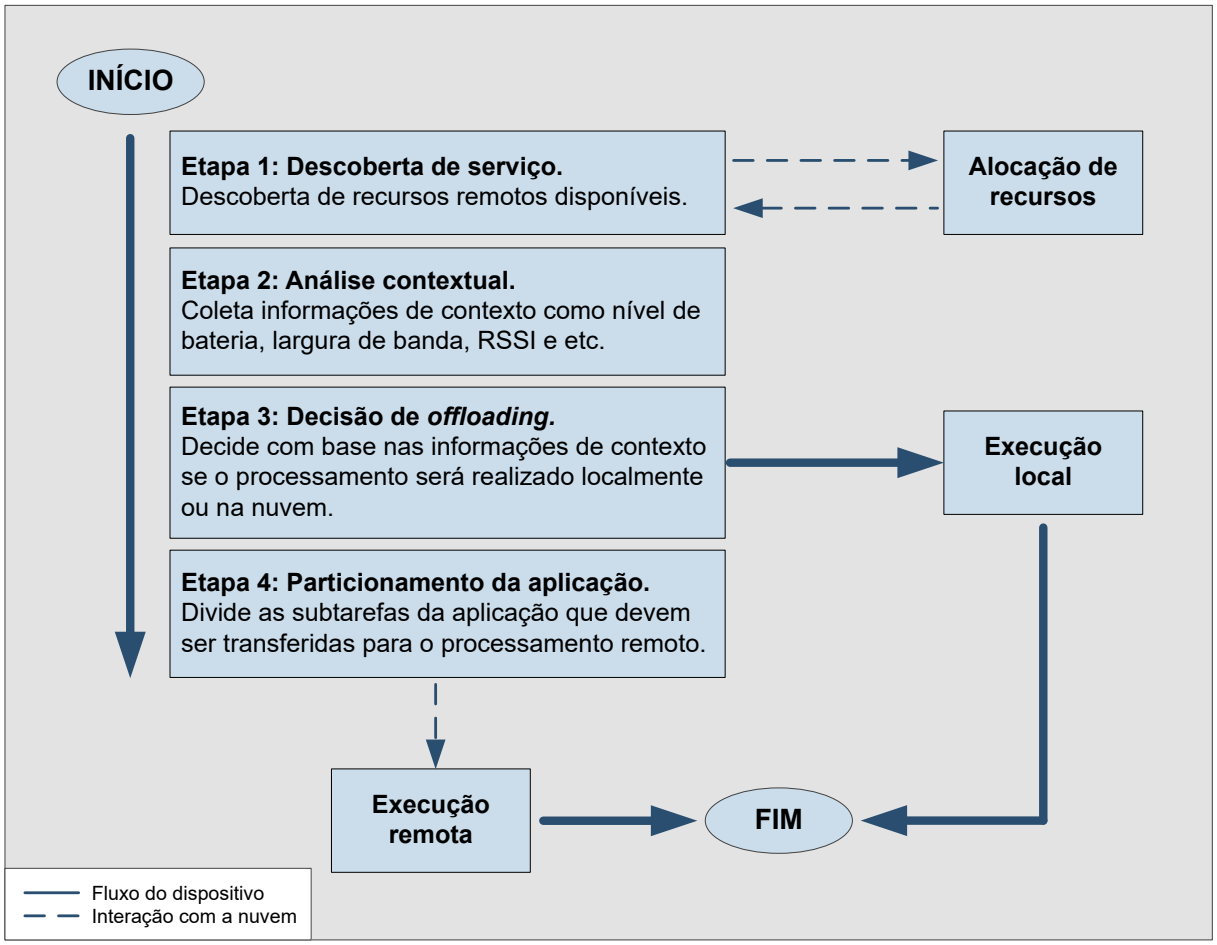
Dentre os benefícios no uso da técnica do *offloading* computacional nos dispositivos móveis, destaca-se: o ganho de desempenho das aplicações em execução (JUNIOR; LOPES, 2015); o aumento do tempo de bateria devido à redução do consumo energético (KUMAR; LU, 2010); diminuição da carga de processamento sobre um dispositivo sobrecarregado, executando o *offloading* de partes de uma aplicação (ou a aplicação inteira) para uma máquina remota (YANG et al., 2016).

De acordo com Li et al. (2015), o processo de *offloading* computacional é dividido basicamente em 4 etapas, detalhadas a seguir. A Figura 2, apresenta o fluxo básico das atividades contidas nestas etapas, que são realizadas pelas aplicações e nuvem nos ambientes MCC.

(1) O primeiro passo acontece quando aplicação é iniciada no dispositivo móvel, através da execução da descoberta de serviço. Este serviço, é responsável pela busca de recursos de nuvem remota (por exemplo, nuvem pública ou *cloudlet*), através das redes sem fio disponíveis (por exemplo, WiFi ou 3G). Caso haja sucesso na busca, a nuvem aloca os recursos necessários para atender a demanda solicitada e responde favoravelmente ao dispositivo móvel.

(2) Em seguida, o dispositivo móvel executa uma análise contextual que coleta informações como o nível atual da capacidade da bateria, localização do usuário, largura de banda (*bandwidth*) e nível de sinal da rede sem fio conectada atualmente. Estas

Figura 2 – O processo de *offloading* computacional



Fonte: Adaptada de Li et al. (2015)

informações são utilizadas como critérios em um algoritmo de tomada de decisão de *offloading*. Dessa maneira, a depender do contexto, poderá ser identificado que uma execução local será mais eficiente do que uma execução remota.

③ Nesta etapa, é definido se o *offloading* solicitado pelo usuário através da aplicação móvel será executado localmente pelo dispositivo ou pelo recurso de nuvem remoto. Caso local, significa que a resposta do algoritmo de decisão ou a opção do usuário não foram favoráveis à execução remota. Dessa forma, o dispositivo segue com a execução do processamento local. Se remoto, a aplicação inicia a atividade de particionamento de tarefas.

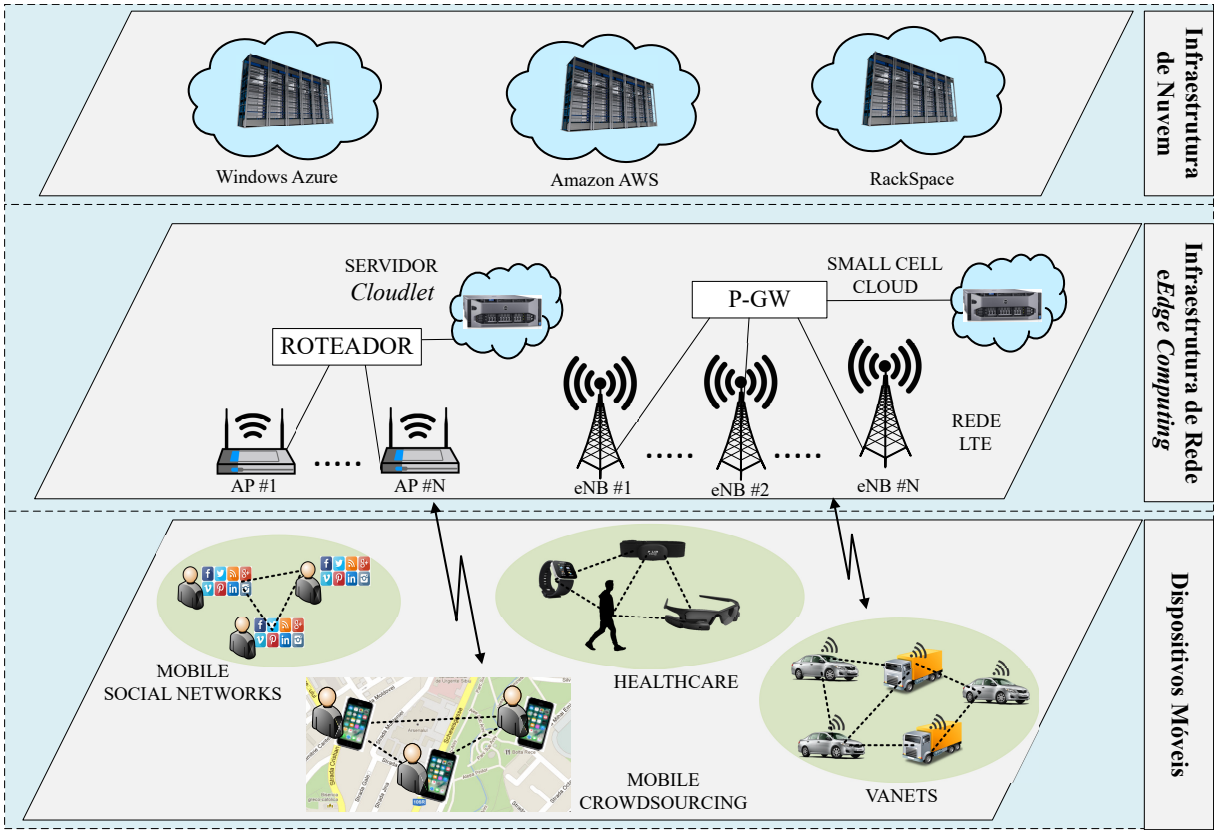
④ Esta última etapa, só acontece caso a decisão de *offloading* seja favorável. Em tempo de execução, a aplicação deve ser particionada com base em subtarefas definidas de modo prévio por um programador. O objetivo deste procedimento é identificar as subtarefas que devem ser transferidas para o recurso de nuvem remota que foi identificado anteriormente. Dessa forma, a nuvem realiza o processamento destas subtarefas e devolve os resultados para a aplicação em execução no dispositivo móvel.

Devido à variabilidade de cenários que compõem os ambientes em MCC, o desempenho da técnica de *offloading* computacional será afetada por alguns aspectos como: (1) eficiência do acesso sem fio, causando interferências severas se múltiplos dispositivos próximos decidirem executar o *offloading* na mesma rede simultaneamente; (2) disponibilidade dos serviços de nuvem, aumentando o tempo de resposta ao usuário devido ao congestionamento de acesso a recursos; (3) requisitos da aplicação, que podem ser impactadas de maneiras variadas a depender de seus requisitos de QoS (JUNIOR; LOPES, 2015).

2.1.3 Arquitetura MCC

A arquitetura MCC de três camadas é apresentada na Figura 3 e inclui a identificação de cada componente, cenários e infraestrutura para o funcionamento de todo o ambiente. A arquitetura genérica contém os dispositivos móveis na camada inferior, que é composto por usuários, dispositivos e redes *ad-hoc/ opportunistic networks*; uma camada intermediária com redes de acesso e o núcleo de operadoras móveis, bem como recursos de nuvem privada (por exemplo, *cloudlets*) e *Edge Computing*; e uma camada superior onde o *Internet Service Provider* (ISP) e as infraestruturas de nuvem públicas estão presentes.

Figura 3 – Arquitetura MCC



Fonte: Elaborada pelo Autor

Na camada dos dispositivos móveis, é onde os dispositivos e aplicações consomem os recursos de nuvem, mantendo suas rotinas de mobilidade e conectividade. As aplicações de *Mobile Social Networks* (MSN) consomem dados gerados pelo dispositivos móveis (por exemplo, GPS, câmera, acelerômetro), fornecendo conteúdo adicional que atenda às necessidades sociais dos usuários. O usuário pode criar uma publicação contendo informações sobre a localização de uma pessoa, com fotos e comentários sobre o nível de satisfação em usar algum produto ou serviço. Esta publicação pode ser compartilhada através da nuvem, proporcionando interação com outros usuários e contribuindo para a avaliação do estabelecimento (HU et al., 2015).

O *Mobile Crowdsourcing* é um tipo de serviço de comércio eletrônico, onde os usuários móveis criam uma nuvem móvel para vender recursos e serviços de nuvem (por exemplo, coleta de dados e processamento) para consumidores de serviços através das redes móveis ou se comunicando com *cloudlets* e usuários vizinhos através de tecnologias como *Bluetooth/Near Field Communication* (NFC)/WiFi. Este recurso de serviço emergente, pode permitir que usuários móveis assumam tarefas terceirizadas (CHATZIMILIOUDIS et al., 2012).

Em outro exemplo, as aplicações de *healthcare* exigem quantidades cada vez maiores de recursos computacionais e de comunicação. O *Mobile Healthcare System* (MHS) oferece maior conveniência e agilidade na assistência médica através de infraestruturas de nuvem móvel. Por exemplo, um esquema de *cloudlet* pode ser usado para analisar dados e registros de pacientes, extrair recomendações e estimar condições de saúde humana, enquanto o paciente está se movendo pela cidade em direção ao hospital (JEMAL et al., 2015).

Com as *Vehicular Ad-hoc Networks* (VANETS), o uso das aplicações veiculares estão aumentando tremendamente com o crescente interesse em incorporar uma nova classe de serviços interativos usando interfaces de bordo (ASHOK; STEENKISTE; BAI, 2016).

A camada de infraestrutura de rede e *Edge Computing*, contém *cloudlets* em áreas de alta densidade (como cafés e *shoppings*), onde o usuário se conecta ao via WiFi para o *offloading* (SATYANARAYANAN et al., 2009). Na mesma camada, uma *Small Cell Cloud* opera dentro da rede da operadora móvel, onde um conjunto de pequenas células oferecem a funcionalidade de *cloudlet*, denominada *Mobile Edge Computing* (MEC) (ETSI, 2014).

Por fim, a camada de infraestrutura de nuvem fornece recursos de computação virtualizados sob a forma de um serviço (computação, armazenamento, etc.) implementado em um servidor de nuvem remota (no fornecedor de serviços de nuvem), onde os consumidores de serviços são dispositivos móveis (por exemplo, *smartphones*, *tablets*), VANETS, MSNs, etc. Os dispositivos móveis se conectam à Internet por uma conexão geralmente de maior custo, como *Long Term Evolution* (LTE), *Universal Mobile Telecommunication System* (UMTS) e *General Packet Radio Service* (GPRS). Embora a conexão móvel seja cara, ela contém muitas vantagens, como cobertura maior, suporte de entrega, disponibilidade, privacidade de localização e criptografia nativa (ABDO; DEMERJIAN, 2017).

2.2 Redes Definidas por Software

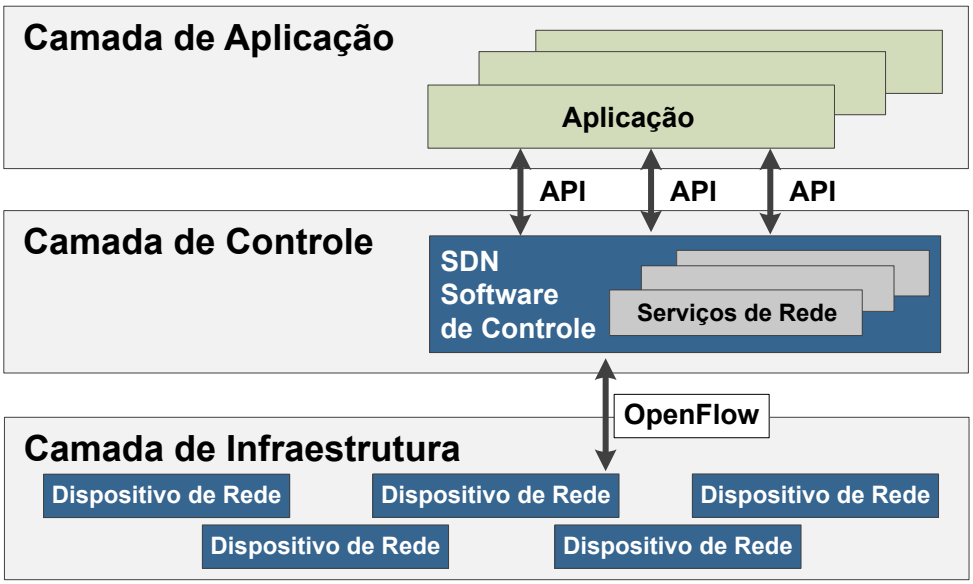
O SDN é um paradigma que possui como principal característica a separação do plano de controle de um dispositivo de rede do seu plano de dados, tornando possível controlar, monitorar e gerenciar uma rede a partir de um nó centralizado (o controlador SDN). Nesta seção, são apresentados os principais fundamentos e como este paradigma funciona, com o objetivo de apresentar a teoria principal do que foi usado nesta dissertação.

2.2.1 Fundamentos básicos

O conceito de *Software-Defined Networking* (SDN) propõe superar certas limitações existentes no paradigma das redes tradicionais. Como os equipamentos de infraestrutura da rede são projetados com *hardware* e *software* específicos, os administradores são limitados a funcionalidades pré-definidas, o que não permite a customização através de aplicações e o desenvolvimento de protocolos. Com SDN, ocorre a separação do plano de controle e plano de dados, sendo possível centralizar o gerenciamento, e através de software, controlar os dispositivos da rede (KREUTZ et al., 2015).

Em SDN, um controlador centralizado possui a visão geral de toda a rede, permitindo controlar os dispositivos (por exemplo, *switches*) através do encaminhamento de pacotes. As tabelas de fluxo existentes nestes dispositivos, são usadas para controlar os pacotes (por exemplo, encaminhamento para uma porta específica, alteração de cabeçalhos e descarte), de acordo com as políticas administradas pelo controlador. Dessa forma, dispositivos SDN podem atuar com a função de roteador, *switch*, *Network Address Translation* (NAT), *firewall* ou similares, dependendo das regras de manipulação de pacotes (ONF, 2017a).

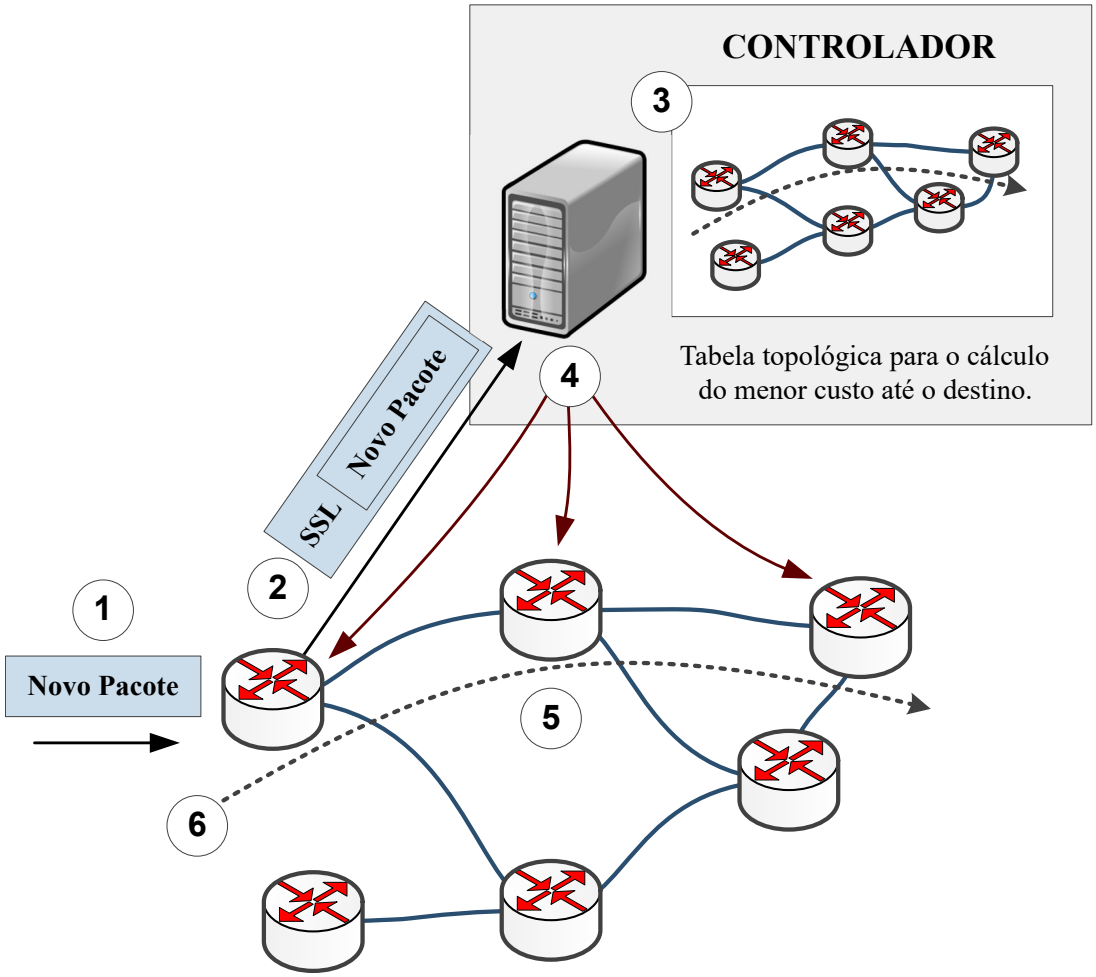
Figura 4 – Arquitetura SDN



Fonte: ONF (2017a)

A Figura 4 apresenta a arquitetura SDN que consiste em três camadas principais: Aplicação, Controle (Plano de controle) e Infraestrutura (Plano de dados). A camada superior é onde se encontram as aplicações que se comunicam com o controlador através de *Application Programming Interface* (API). Este desacoplamento com o plano de dados, simplifica a operação e o gerenciamento de rede, mantendo a padronização com o controlador. A camada de controle manipula os dispositivos de encaminhamento através do controlador, para atingir o objetivo específico da aplicação em execução. O controlador usa a interface *southbound* para se conectar ao plano de dados. A camada de infraestrutura manipula os pacotes dos diversos dispositivos através do protocolo *OpenFlow*, com base nas configurações recebidas pelo controlador (FARHADY; LEE; NAKAO, 2015).

Figura 5 – Modo de funcionamento das redes SDN



Fonte: Elaborada pelo Autor

A seguir, está descrita o funcionamento de modo mais detalhado de um ambiente SDN, conforme o exemplo apresentado pela Figura 5.

(1) Quando uma nova máquina entra na rede e envia seu primeiro pacote em direção ao seu destino, este é recebido pelo dispositivo de acesso (por exemplo, *switch*,

roteador) mais próximo. Como ainda não existem regras de encaminhamento pré-configuradas neste dispositivo, é necessário que lhe seja indicado uma regra de fluxo para encaminhar o pacote de maneira correta.

(2) Nesta etapa, o dispositivo de rede realiza uma consulta ao controlador através de um meio seguro, para que sejam definidas nas tabelas de fluxo, as regras de encaminhamento necessárias para prover a comunicação solicitada pelo novo usuário.

(3) Recebida esta consulta, o controlador calcula através da aplicação em execução que foi desenvolvida pelo administrador da rede, utiliza sua visão holística para calcular qual o menor custo (por exemplo, quantidade de saltos, maior vazão) entre o ponto de origem, até o destino solicitado.

(4) Com o resultado dos cálculos feitos pela aplicação, o controlador através do protocolo *OpenFlow*, define em quais dispositivos de rede será preciso adicionar as regras de encaminhamento necessárias para atender o fluxo solicitado.

(5) Sendo concluída a etapa de adição das regras, a nova máquina realizará com sucesso a conexão solicitada em todo o trajeto de ida e volta, através do encaminhamento e roteamento dos pacotes.

(6) Caso sejam realizadas sucessivas solicitações de fluxo com as mesmas configurações de encaminhamento já pré-configuradas no plano de dados, não será mais necessário realizar uma nova consulta ao controlador e os pacotes já serão transferidos automaticamente.

Devido às características apresentadas, acredita-se que a SDN seja uma nova tecnologia de rede que simplifique a operação e gerenciamento das redes atuais, permitindo inovações e novos projetos de rede. Devido aos benefícios em potencial da SDN nas atuais arquiteturas da internet e do futuro, como *Information-Centric Networking* (ICN), ganhou atenção considerável dos pesquisadores (AHLGREN et al., 2012).

2.2.2 Protocolo *OpenFlow*

O protocolo *OpenFlow* é o padrão mais popular utilizado nas redes SDN. Foi desenvolvido pela universidade de *Stanford* (MCKEOWN et al., 2008) e posteriormente foi padronizado pela *Open Networking Foundation* (ONF)[1], um órgão composto por um grupo de operadoras de rede, provedores de serviços e fornecedores. Isso permitiu que o protocolo fosse disponibilizado de forma aberta e bem definida, possibilitando a sua adoção em larga escala por pesquisadores e fabricantes de dispositivos de rede.

[1] ONF: https://www.opennetworking.org/

A primeira versão do protocolo *OpenFlow* foi a 1.0, lançada em março de 2008. O desenvolvimento das versões seguintes foi padronizado e gerenciado pela ONF. A versão 1.1 que foi lançada em 28 de fevereiro de 2011, teve como principal novidade a inserção de múltiplas tabelas de fluxo. Em fevereiro de 2012, foi disponibilizada a versão 1.2 do *OpenFlow*, que permite a comunicação simultânea entre múltiplos controladores. A versão 1.3 foi lançada em abril de 2012, e suporta o protocolo *Internet Protocol version 6* (IPv6) de movo nativo. Em agosto de 2013, foi lançada a versão 1.4 que dá suporte a interfaces de fibras ópticas. Sua versão mais recente, a 1.5 que traz o recurso de *Egress Tables*, foi lançada em dezembro de 2014. A versão 1.6 já está disponível desde setembro de 2016, contudo, apenas acessível aos membros da ONF (ONF, 2017b). O Quadro 1 apresenta um comparativo entre as especificações e principais características das versões lançadas até o momento.

Quadro 1 – Comparação de especificações *OpenFlow*

Funcionalidades principais	**Versões do *OpenFlow* suportadas**					
	v1.0	**v1.1**	**v1.2**	**v1.3**	**v1.4**	**v1.5**
Flow Tables	Única	Várias	Várias	Várias	Várias	Várias
MPLS	✗	✓	✓	✓	✓	✓
IPv6	✗	✗	✓	✓	✓	✓
Múltiplos controladores	✗	✗	✓	✓	✓	✓
Group Table	✗	✗	✗	✓	✓	✓
Meter	✗	✗	✗	✓	✓	✓
Interfaces de fibra óptica	✗	✗	✗	✗	✓	✓
Egress Tables	✗	✗	✗	✗	✗	✓

Fonte: Adaptado de JÚNIOR (2016)

A compreensão das particularidades de cada versão do protocolo *OpenFlow* é necessária devido a três aspectos: (1) o desenvolvedor da aplicação SDN precisa escolher qual versão atenderá melhor sua necessidade de implementação; (2) cada controlador possui características específicas como linguagem de programação e versão do *OpenFlow* suportada; (3) é preciso verificar se a versão escolhida é compatível com todos os equipamentos existentes na infraestrutura de rede. Em resumo, é preciso que seja feita uma análise de todos estes três aspectos de modo que não haja incompatibilidade entre estes.

2.2.3 Controladores *OpenFlow*

Para o desenvolvimento de aplicações em cenários SDN, é preciso compreender as funcionalidades dos controladores de rede, considerados o "cérebro" deste tipo de infraestrutura. Eles oferecem um ambiente baseado no paradigma denominado Programação Orientada a Eventos (*Event-Driven Programming*), onde é possível desenvolver e definir

ações para o controle de fluxo dos pacotes, com base no acesso aos eventos gerados por uma interface de rede que siga o padrão *OpenFlow*. Com esse tipo de ambiente, é mais simples implementar políticas de segurança baseadas em níveis de abstração maiores, que por exemplo, endereços *Ethernet* ou *Internet Protocol* (IP) (KHONDOKER et al., 2014).

Da mesma maneira, SDN possibilita a implementação de controladores de rede que suportem estratégias de monitoração e acompanhamento de tráfego mais sofisticadas, ou ainda soluções que ofereçam novas abstrações para os usuários da rede. Por exemplo, cada usuário de um *datacenter* pode ter a visão de que todas as suas máquinas estão ligadas a um *switch* único e privado, independente dos demais.

Quadro 2 – Comparativo de controladores SDN

	POX	**Ryu**	**Trema**	**Floodlight**	**ODL**
Compatibilidade	Mininet / OVS	Mininet / OVS	Mininet / OVS	Mininet / OVS	Mininet / OVS
GUI	Sim	Sim (beta)	Não	Web UI	Sim
REST API	Não	Sim	Não	Sim	Sim
Eficiência	Média	Média	Alta	Média	Média
Open Source	Sim	Sim	Sim	Sim	Sim
Documentação	Pobre	Média	Média	Boa	Média
Linguagem	Python	Python	C/Ruby	Java	Java
Modularidade	Média	Média	Média	Alta	Média
Plataforma	Linux, Mac e Windows	Linux	Linux	Linux, Mac e Windows	Linux
TLS	Sim	Sim	Sim	Sim	Sim
OpenFlow	v 1.0	v 1.0 - 3.0	v 1.0	v 1.0 - 3.0	v 1.0 - 3.0
OpenStack	Não	Sim	Sim	Sim	Sim

Fonte: Elaborada pelo Autor

Atualmente, existem muitos controladores disponíveis como, o POX[2], Ryu[3], Trema[4], Flooglight[5] e OpenDaylight[6]. Cada um deles se diferenciam por característica particulares, como por exemplo, variedades de interfaces de interação, suporte a *Graphical User Interface* (GUI), linguagem de programação e versões do *OpenFlow* suportadas e integração com o OpenStack[7]. O Quadro 2, apresenta os controladores citados com um pequeno resumo de suas principais características.

[2] POX: https://github.com/noxrepo/pox/
[3] Ryu: https://osrg.github.io/ryu/
[4] Trema: https://trema.github.io/trema/
[5] Flooglight: https://www.projectfloodlight.org/floodlight/
[6] OpenDaylight: https://www.opendaylight.org/
[7] OpenStack: https://www.openstack.org/

2.2.4 Controlador Ryu

Apesar de terem sido apresentados e comparados vários controladores no Quadro 2, o Ryu será melhor detalhado devido a este ser o adotado nesta proposta. Dentre os motivos que levaram a esta escolha, destaca-se a vasta documentação gratuita disponível através da Internet, os baixos requisitos de *hardware* necessários para seu funcionamento e baixa curva de aprendizagem.

O termo Ryu significa "fluxo" (*flow*) em Japonês, e é um controlador SDN de código aberto, escrito totalmente em Python e disponibilizado gratuitamente pela licença Apache 2.0. (COMMUNITY, 2017). Este controlador fornece componentes de *software* com uma API bem definida, facilitando a criação de novas aplicações de gerenciamento e controle de rede. Este tipo de abordagem, permite que os desenvolvedores possam modificar de forma fácil e rápida os componentes existentes ou implementar novos para garantir que a rede possa atender às mudanças de suas aplicações.

O controlador Ryu também oferece a implementação de APIs REST[8] que torna possível obter todo tipo de informação sobre os *switches*, além de instalar manualmente novos fluxos, grupos e consultar estatísticas. Este recurso é interessante, pois torna possível a integração destas funcionalidades através de aplicações de outras linguagens de programação. Por exemplo, utilizando a linguagem JAVA, é possível implementar uma interface consumidora destas informações com o uso da classe `URLConnection`[9].

Dentre os protocolos suportados estão o *OpenFlow* (até a versão 1.5), *Network Configuration Protocol* (NETCONF) e o *OpenFlow Management e Configuration Protocol* (OF-Config). O Ryu foi testado e certificado para trabalhar com uma série de *switches OpenFlow*, incluindo Open vSwitch e ofertas da Centec, Hewlett Packard, IBM e NEC (SDXCENTRAL, 2017). O *OpenStack*, que é um gerenciador de sistemas de nuvem, oferece suporte a implantações do Ryu como o controlador de rede (OSRG, 2017).

2.3 Lógica *Fuzzy*

A Lógica *Fuzzy*, também denominada lógica nebulosa ou difusa, é uma teoria que incorpora a experiência, a intuição, o conhecimento especialista e a natureza imprecisa do processo decisório humano através de um conjunto de regras ou heurísticas simples. Para o entendimento do procedimento adotado neste trabalho, faz-se uma introdução aos principais conceitos que envolvem a Lógica *Fuzzy*.

[8] API REST Ryu: http://ryu.readthedocs.io/en/latest/app/ofctl_rest.html
[9] URLConnection: https://docs.oracle.com/javase/7/docs/api/java/net/URLConnection.html

2.3.1 Teoria dos Conjuntos *Fuzzy*

O fundamento da Lógica *Fuzzy* é a Teoria dos Conjuntos *Fuzzy*, conceito este desenvolvido originalmente por Zadeh (1965), que observou a impossibilidade de modelar através da Lógica Tradicional, certos sistemas que apresentavam limites imprecisos ou vagos.

A Teoria Clássica dos Conjuntos (*crisp*) fundamenta a Lógica Tradicional, que foi formulada primeiro por Aristóteles através da Lei do Meio Excluído (*Law of the Excluded Middle*). Esta lei, define que um determinado elemento de um universo em discurso (domínio), apenas pode pertencer ou não ao referido conjunto, adotando apenas os valores falso (0) ou verdadeiro (1) (KOSKO; ISAKA, 1993). Devido aos resultados serem restritos a estas duas sentenças, a Lógica Tradicional também é conhecida como Lógica *booleana*. A Função 2.1 apresenta esta característica dos Conjuntos Clássicos nos elementos (x) de um dado conjunto A em um universo X:

$$f_A(x) = \begin{cases} 1 \, se, \, e \, somente \, se, \, x \in A \\ 0 \, se, \, e \, somente \, se, \, x \notin A \end{cases} \tag{2.1}$$

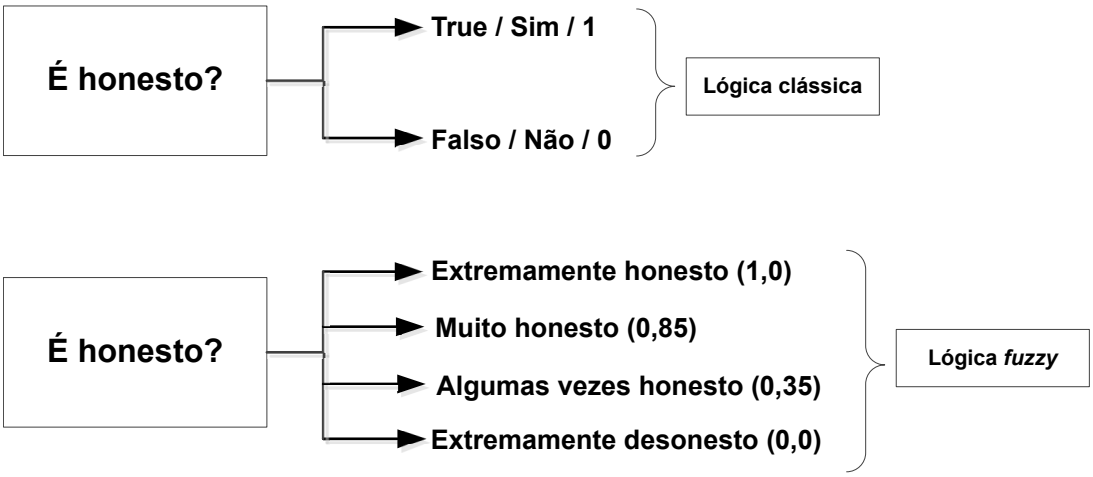
Um Conjunto *Fuzzy* é um conjunto sem um limite claramente definido, ou seja, ele pode conter elementos com apenas um grau parcial de pertinência, que são o conceito-chave de Zadeh. Trata-se de um valor de associação que mede o grau que um objeto atende a propriedades vagas e/ou imprecisas. Esses valores de pertinência que podem assumir qualquer valor entre 0 e 1 (onde 0 indica uma completa exclusão e 1 uma completa associação), são definidos sobre o universo de discurso por uma função de pertinência, que é o Conjunto *Fuzzy*. Ou seja, a Lógica *Fuzzy* foi inicialmente construída a partir dos conceitos já estabelecidos de Lógica Tradicional, sendo uma generalização da Teoria Clássica dos Conjuntos. Zadeh também definiu as operações para os Conjuntos *Fuzzy*, que compreendem todas as ferramentas matemáticas necessárias para aplicá-los (EBERHART, 2007). A Função 2.2 apresenta formalmente as características nebulosas, onde um conjunto A mapeia seus elementos (x) de um universo X para o intervalo [0,1]:

$$\mu_A(x) : X \to [0, 1] \tag{2.2}$$

Para resumir a compreensão entre as lógicas Clássica e *Fuzzy*, a Figura 6 apresenta um diagrama onde no sistema clássico, a "honestidade" pode ser classificada como verdade absoluta (valor 1) ou falsidade absoluta (valor 0). Já no sistema nebuloso, existem outros valores entre 1 e 0 que indicam a "honestidade", onde neste exemplo, chamamos de "nível de honestidade". Em outras palavras, podemos dizer que a Lógica *Fuzzy* não é uma lógica que é "nebulosa" ou que se refere a falta de conhecimento sobre algum parâmetro, mas uma lógica que é usada para descrever de forma mais ampla um determinado critério. Aplicando a nomenclatura técnica adotada por Zadeh aos Conjuntos *Fuzzy*, a variável "honestidade" é

denominada como uma variável linguística, e a classificação de suas sentenças de valores linguísticos (ROSS, 2009).

Figura 6 – Termos da variável honestidade nas Lógicas Clássica e *Fuzzy*



Fonte: Elaborada pelo Autor

Como nos métodos clássicos de MCDM, é comum que os critérios assumidos possuam pesos expressos de forma clara para o ranqueamento das alternativas de modo eficiente, a Lógica *Fuzzy* permite noções imprecisas. Dessa forma, é possível que informações como "alto", "médio" e "baixo" sejam capturadas, podendo assim tomar decisões mais eficientes quando trabalhamos com critérios de contextos variados. Quando comparamos as principais potencialidades da Lógica *Fuzzy* com outros esquemas que também utilizam dados imprecisos (por exemplo, redes neurais), é que sua base de conhecimento que estão no formato de regras, possibilita uma análise de fácil entendimento. Esta abordagem de usos de regras, também torna fácil a manutenção e a atualização da base de conhecimento (BARROS; BASSANEZI, 2006).

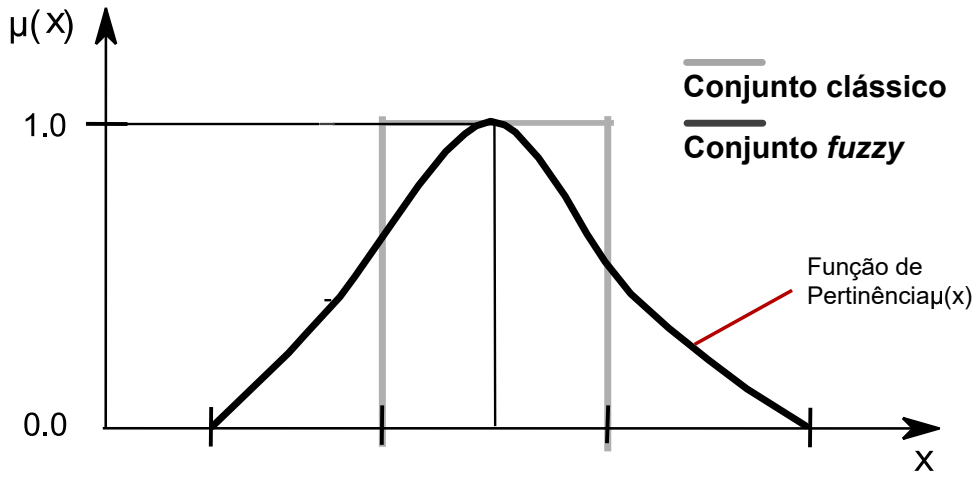
2.3.2 Funções de pertinência

Para ser caracterizada como uma função de pertinência compatível com a distribuição de um Conjunto *Fuzzy*, é preciso que ela permita a representação das variáveis linguísticas dentro de valores entre 0 e 1. Contudo, na escolha destas funções, deve-se levar em conta o contexto em que elas serão utilizadas na representação destas variáveis. Nos sistemas *fuzzy*, as funções de pertinência podem ser escolhidas arbitrariamente de acordo com a experiência do usuário. Ou seja, esta é uma das principais características da Lógica *Fuzzy*, onde é permitido o uso do conhecimento de especialistas (não matemáticos) sobre o processo que se quer estudar para projetar uma determinada solução, que nesse caso, auxilia na escolha do número e no formato destas funções (MENDEL, 1995).

A Figura 7 apresenta um exemplo comparativo entre a representação de um valor para uma variável linguística através de uma função de pertinência. É possível evidenciar em

relação ao sistema *fuzzy*, a limitação do sistema *crisp* na capacidade do uso de um grau de associação entre 0 e 1, indicando que os mesmos podem pertencer parcialmente a um conjunto.

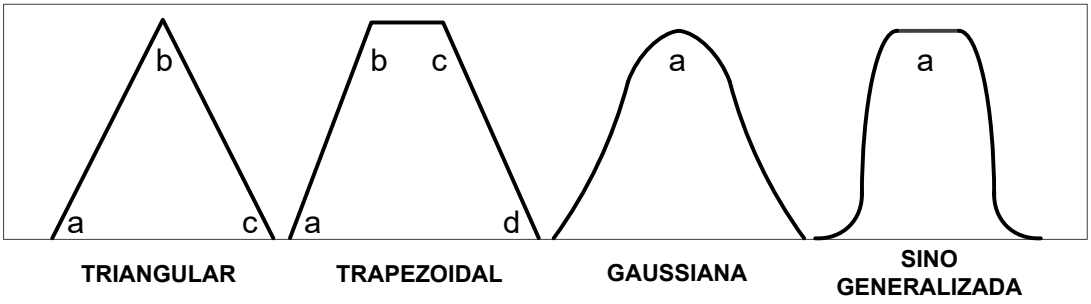
Figura 7 – Representação de um valor em Conjuntos *Crisp* e *Fuzzy*



Fonte: Elaborada pelo Autor

De acordo com Tanscheit (2004), as funções de pertinência podem ser de várias formas, onde alguns exemplos são destacados a seguir: A Triangular é formada por uma tripla (a, b, c), onde "a" e "c" determinam o intervalo que a função assume valores diferentes de zero e "b" é o nível de pertinência máximo. A Trapezoidal é semelhante à Triangular, porém possui como característica um conjunto de quatro valores (a, b, c e d), onde "a" e "d" determinam os valores diferentes de zero, e "b" e "c" o intervalo de pertinência igual a 1. A Gaussiana é caracterizada pela sua média e desvio padrão. Este tipo de função apresenta um decaimento suave e possui valores diferentes de zero para todo domínio da variável estudada. A Sino Generalizada, permite que o valor máximo de pertinência assuma um trecho de valores mais amplo. Tanto na função Gaussiana como na Sino Generalizada, o valor "a" sempre representa o centro da curva. Estas funções são apresentadas pela Figura 8.

Figura 8 – Funções de Pertinência

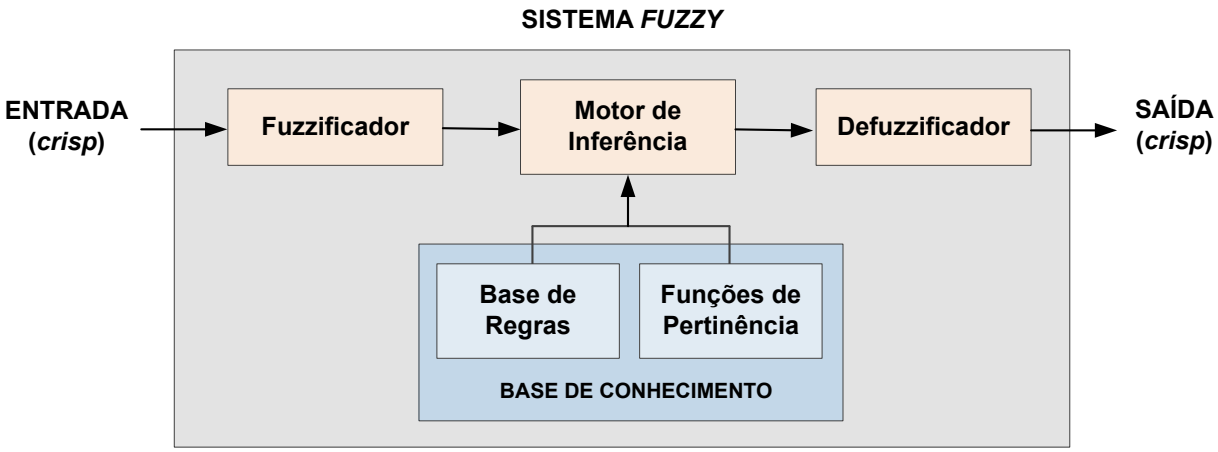


Fonte: Elaborada pelo Autor

2.3.3 Componentes do Sistema Fuzzy

De acordo com Tzafestas e Blekas (1997), um sistema *fuzzy* geralmente é composto pelos seguintes componentes principais: Fuzzificador, Base de conhecimento, Máquina de Inferência e Defuzzificador, conforme apresentados na Figura 9. A seguir estes componentes são explicados com mais detalhes:

Figura 9 – Componentes do sistema *fuzzy*



Fonte: Elaborada pelo Autor

- **Fuzzificador:** Esta primeira etapa do processo da Lógica *Fuzzy*, é responsável por processar e transformar as entradas do sistema no formato *crisp* em uma representação *fuzzy*. Este processo, permite que estas entradas se tornem instâncias de variáveis linguísticas para envio ao Motor de Inferência, formato este necessário para execução de seus procedimentos.

- **Base de Conhecimento:** Este componente é responsável por armazenar o conhecimento do problema com base nas informações fornecidas por especialistas. Nesse caso, junto com as funções de pertinência, encontramos aqui as regras *fuzzy*, caracterizadas pelo formato "SE x E/OU y ENTÃO z", onde as entradas (x e y) são comparadas com as operações "E/OU" para obter a saída (z). Estas sentenças possibilitam o raciocínio do sistema para derivar conclusões a partir destas regras.

- **Motor de Inferência:** É a principal etapa do sistema, conhecido como *Fuzzy Inference System* (FIS). Aqui ocorrem os procedimentos e são inferidas as ações de controle utilizando as regras *fuzzy* armazenadas na Base de Conhecimento, para processar os valores de entrada recebidos após o processo de Fuzzificação. Em seguida, o resultado extraído em formato *fuzzy* é enviado ao Deffuzificador.

- **Deffuzificador:** É a última etapa do processo que interpreta as ações de controle expressas em um conjunto *fuzzy* para um valor solução que possa ser diretamente

aplicado ao sistema controlado. Em outras palavras, nesta transformação obtemos uma saída precisa (número real) que representa os valores inferidos da variável linguística de saída. Para este objetivo, existem alguns métodos disponíveis, como por exemplo, Média dos Máximos e Centro de Área.

2.3.4 Características da Lógica *Fuzzy*

Podemos resumir as seguintes características da Lógica *Fuzzy* que podem justificar seu uso em relação à Lógica Clássica (JANG, 1997):

- Sua abordagem e conceitos são intuitivos e fáceis de entender, reduzindo o tempo de aprendizagem e permitindo o rápido uso do sistema;
- É flexível, permitindo a adição de novas funcionalidades sem a necessidade de recomeçar todo o sistema;
- Trabalha com dados imprecisos, permitindo trabalhar com este tipo de abordagem durante o processo;
- Permite combinar qualquer conjunto de dados de entrada e saída, devido à sua capacidade de modelar funções não-lineares de complexidade arbitrária;
- Pode ser construída utilizando a experiência de especialistas, ou seja, pessoas que têm domínio do sistema;
- Em muitos casos, é combinada com técnicas convencionais de controle;
- Utiliza a linguagem natural, que é a base da comunicação humana.

2.4 Considerações finais

Este capítulo apresentou um resumo dos principais conceitos que fundamentam o desenvolvimento do sistema de decisão que será apresentado nessa proposta. Inicialmente foram apresentados os principais conceitos relativos à computação em nuvem móvel, seguido dos conceitos que envolvem as redes definidas por software e por último foram apresentadas informações sobre a Lógica *Fuzzy*, possibilitando ao leitor conhecer de forma básica os conceitos que fundamentam esse trabalho. O Capítulo 3 apresentará um resumo de alguns trabalhos relacionados aos temas expostos neste Capítulo 2.

3 TRABALHOS RELACIONADOS

Este capítulo descreve trabalhos relacionados encontrados na literatura durante a pesquisa desta dissertação. Os trabalhos foram divididos em duas Seções. A Seção 3.1 aborda artigos sobre estratégias de conectividade entre os dispositivos e os recursos remotos no ambiente MCC. Na Seção 3.2 são discutidos os trabalhos que abordam soluções de seleção de rede e nuvem para o *offloading*.

3.1 Estratégias de conectividade em MCC

O trabalho de Lee et al. (2013), propõe uma arquitetura baseada em servidores *proxy* para melhorar o desempenho do enlace para cada dispositivo móvel no ambiente MCC. Através deste esquema, o dispositivo não precisa manter uma conexão direta com o servidor da nuvem. Um *proxy* na mesma sub-rede, será responsável por intermediar este processo. Os autores também propõem um algoritmo de descoberta de rede de acesso para otimizar o uso da largura de banda. Quando o dispositivo móvel muda de AP, o algoritmo ajuda na conexão à rede ideal para acessar o serviço de nuvem. No entanto, destaca-se como primeiro aspecto de comparação com essa dissertação, a ausência de uma estratégia de *handoff*, isto é, os resultados, porventura, processados na nuvem e não devolvidos enquanto o dispositivo estava conectado ao AP antigo, serão perdidos se um *handoff* ocorrer. Assim, o dispositivo terá que repetir a operação de *offloading* considerando, agora, o novo AP para o qual tenha migrado.

Na solução utilizada em LEE et al. para eleição de um servidor *proxy*, é realizada a verificação periódica de informações referentes ao estado destes *proxies* como, seu estado de atividade (ativo ou inativo), desempenho de *hardware* (por exemplo, CPU e memória) e a quantidade de conexões simultâneas ainda suportadas. Contudo, soluções de conectividade baseadas nesses servidores possuem a desvantagem de gerar possíveis sobrecargas e redução no desempenho do *offloading*, principalmente neste artigo, onde são escolhidos *proxies* que já estão acessando o mesmo tipo de serviço solicitado pelo dispositivo móvel. Outro aspecto em LEE et al. é que para seleção da rede de acesso, o algoritmo extrai informações de camada 3, onde são analisadas mensagens do protocolo *Neighbor Discovery Protocol* (NDP) transmitidas pelos roteadores. Contudo, neste caso, as informações coletadas (por exemplo, contagem de saltos para o servidor *proxy* e grau de hierarquia na topologia) estão mais relacionadas a aspectos de infraestrutura da rede que, apesar de importantes, não consideram o *Received Signal Strength Indicator* (RSSI).

Na mesma linha do trabalho anterior, o esquema baseado em servidores *proxy* proposto por Khalaj e Lutfiyya (2013) distribui as várias solicitações de *offloading* entre estes servidores. O esquema possui um conjunto de *proxys* em locais distintos, projetados como

uma espécie de biblioteca dinâmica de serviços genéricos, que podem ser usados por um conjunto de serviços e aplicativos de clientes. Na arquitetura proposta pelos autores, o dispositivo móvel deve se conectar primeiramente a um *Proxy Finder Server* (PFS) que é responsável por encontrar os serviços de *proxy* apropriados, monitorar limiares e gerenciar operações de *handoff* entre os servidores. Dessa forma, os *proxies* funcionam como nós intermediários entre a nuvem e o dispositivo para devolver os resultados das execuções de *offloading*.

Apesar do trabalho de KHALAJ; LUTFIYYA propor uma solução para o problema da sobrecarga nos servidores *proxy* com o uso do PFS, estes servidores não possuem a mesma capacidade de processamento de uma nuvem. Além disso, com o advento das *cloudlets*, ambiente de nuvem utilizado nesta dissertação, os *proxies* tornaram-se obsoletos no ecossistema MCC, principalmente pelo fato das *cloudlets* disponibilizarem seus serviços no mesmo ambiente sem fio do usuário, provendo um tempo de resposta e latência menores. Outro diferencial dessa dissertação, é a utilização de um controlador SDN para gerenciar as conexões através de regras de encaminhamento de pacotes, não havendo a necessidade de um agente intermediário entre o dispositivo móvel e a nuvem.

A pesquisa de Ravi e Peddoju (2014) provê uma estratégia com o objetivo de lidar com a mobilidade do usuário e eficiência energética do dispositivo móvel em ambientes MCC. Para isso, os autores propõem algoritmos de decisão de *offloading* e de *handoff*, para determinar em que situações o processamento deve ser feito localmente ou remotamente, e quando o dispositivo deve alternar entre duas ou mais *cloudlets*. A arquitetura proposta nesse artigo contém módulos que são distribuídos entre os ambientes de nuvem e o dispositivo móvel do usuário. A nuvem é responsável por guardar informações dos dispositivos conectados e pelo processamento das requisições. O dispositivo móvel contém o *proxy* que o conecta à *cloudlet*, o módulo verificador da conexão e, por fim, dois algoritmos de decisão. O primeiro é baseado em lógica *fuzzy* para tomada de decisão de *offloading* e o segundo para mudança de rede (*handoff*).

Ao contrário da proposta de RAVI; PEDDOJU, o presente trabalho utiliza a lógica *fuzzy* para decisão de *handoff*, pois assume-se que o usuário sempre precisa realizar o processamento remoto. O sistema proposto nesta dissertação, utiliza funções de pertinência com métricas de QoS distintas para cada tipo de aplicação, para analisar todas as redes disponíveis com objetivo de obter decisões mais eficientes. A estratégia utilizada em RAVI; PEDDOJU para esta mesma decisão, não considera estas particularidades, se limitando em utilizar pesos entre os parâmetros de RSSI, taxa de transferência e o número de interações do dispositivo móvel com a *cloudlet*. Ainda em RAVI; PEDDOJU, quando o dispositivo realiza o *handoff*, necessariamente se conecta a uma nova *cloudlet* em um ambiente de *cloudlets* interconectadas. No ambiente proposto dessa dissertação, o sistema de decisão verifica de modo independente em qual recurso conectado (rede ou nuvem) não serão atendidos os requisitos aceitáveis para o *offloading*. Dessa forma, o dispositivo não precisa

se desconectar de uma VM que já esteja fornecendo serviços adequados devido ao disparo de *handoff*, sendo garantida a conectividade com base nas regras SDN, evitando-se assim transferências desnecessárias entre recursos.

O trabalho proposto por Junior et al. (2017) apresenta o *Mobile Offloading System* (MOSys). Um sistema de gerenciamento de mobilidade em MCC que permite operações de *offloading* sem que haja interrupções na entrega de resultados durante as execuções de *handoffs* dos dispositivos entre APs, alcançando o objetivo de prover a transparência de *offloading*. O sistema se beneficia do paradigma SDN como parte do desenvolvimento do MOSys, onde os pesquisadores se inspiraram no protocolo *Proxy Mobile IPv6* (PMIPv6) para criar entidades chamadas *OpenFlow Mobility Access Gateway* (OFMAG) (que representam os roteadores sem fio) e *OpenFlow Local Mobility Anchor* (OFLMA) (que representam os *switches*) nos dispositivos de rede. A *cloudlet* presente na arquitetura também utiliza técnicas de *cache* remoto para reduzir o tempo de resposta de *offloading* das aplicações.

Em relação à estratégia de conectividade, a proposta desta dissertação assemelha-se ao ambiente proposto por JUNIOR et al., no aspecto de utilizar o protocolo *OpenFlow* para gerenciar regras de encaminhamento de pacotes entre o dispositivo móvel e a *cloudlet*. No entanto, em JUNIOR et al. não existe um algoritmo que decida qual o melhor momento do dispositivo móvel executar o *handoff* para outra rede, além de também não existir seleção de recursos de nuvem. A gestão de mobilidade feita pelo MOSys ocorre quando o dispositivo móvel se desloca entre a área de cobertura de diferentes OFMAGs, sendo influenciada principalmente pela localização atual do usuário. Esta dissertação procura preencher esta lacuna, permitindo que o dispositivo móvel realize execuções de *handoff* para uma rede alvo, que possa atender aos requisitos da aplicação em execução. Nessa dissertação, as regras SDN são inseridas para reencaminhar os resultados para o AP conectado, onde também é possível selecionar em caso de degradação do sistema da nuvem, um nova VM na *cloudlet* para o *offloading* dos dados.

A proposta apresentada nesta dissertação difere dos trabalhos acima mencionados devido a vários aspectos. Os benefícios das estratégias baseadas em *proxy* são diretamente relacionados à capacidade deste tipo de servidor. Nesse caso, como geralmente estes recursos são disponibilizados através de hospedeiros com recursos de processamento restritos, isso causará redução no desempenho de *offloading*. Como os recursos da nuvem são cada vez mais poderosos, a extinção destes servidores do ecossistema MCC motivou soluções que gerenciem a conectividade nestes ambientes. Este trabalho procura atender a esse aspecto, utilizando critérios como as condições das redes sem fio disponíveis e o estado da *cloudlet* para decisões de conectividade para seleção de AP e VM, respectivamente. Além disso, é utilizado o encaminhamento de pacotes com regras SDN que não necessitam de módulos para prover conectividade no dispositivo móvel e a *cloudlet*, na entrega dos resultados de *offloading*.

3.2 Seleção de rede e nuvem em MCC

No trabalho de Li et al. (2013) os autores propõem o *Embracing Network Inconsistency for Dynamic Application* (ENDA), uma arquitetura de três camadas com o objetivo de otimizar as decisões de *handoff* para o *offloading* em tempo real e de modo adaptável. O ENDA utiliza a predição da mobilidade do usuário, junto com informações da qualidade da rede e a carga do servidor de nuvem na tomada de decisões. O sistema utiliza módulos e um conjunto de *profilers* (monitores dinâmico de recursos) na nuvem para calcular esta predição e, através de bancos de dados, recuperar informações históricas de mobilidade para melhorar esta acurácia. O ENDA também realiza a seleção de APs, conforme vai definindo a possível rota do usuário, realizando uma filtragem entre as redes sem fio candidatas disponíveis.

No entanto, quando o sistema desenvolvido por LI et al. decide disparar o *handoff* e um novo AP é selecionado, o dispositivo móvel precisa reexecutar as operações de *offloading* dos dados para a nuvem. Esta situação não acontece na proposta dessa dissertação, pois o sistema de decisão trabalha junto com o encaminhamento de pacotes que redireciona os dados processados pela *cloudlet* para o novo AP selecionado pelo dispositivo móvel. Outro aspecto é que o ENDA é focado na predição da mobilidade do usuário para decisão de *handoff*. Esta dissertação utiliza métricas das redes sem fio disponíveis no ambiente do dispositivo (por exemplo, a vazão de todas as outras redes), permitindo definir de maneira eficaz o próximo AP ideal para conexão. Apesar de em LI et al. os autores considerarem dois tipos de recursos essenciais para a seleção em ambientes MCC (rede e nuvem), o ENDA não contempla os requisitos de QoS da aplicação, aspecto adotado por este trabalho.

Em Mitra et al. (2015) os autores apresentam um sistema de mobilidade chamado M^2C^2 que suporta mecanismos de *multihoming* e de seleção de rede ou nuvem para melhorar a eficiência das aplicações. O M^2C^2 incorpora várias entidades de rede e nuvem, como nuvens locais e públicas, um agente central que contém o *Cloud Ranking Service* (CRS) (ranqueamento de recursos de nuvem), as redes WiFi, 3G e os APs. O dispositivo móvel se conecta ao ambiente de nuvem através destas entidades e para o gerenciamento de mobilidade, o M^2C^2 utiliza o protocolo *Multi-homed Mobile IP* (M-MIP) que permite a execução de *handoffs* com baixa latência e perda mínima de pacotes. Como o M-MIP suporta *multihoming*, o dispositivo se conecta a várias redes sem fio simultaneamente antes de disparar o processo de *handoff*.

Para seleção de rede em MITRA et al., o dispositivo precisa examinar periodicamente mensagens de sondagem para todas as redes disponíveis com o objetivo de calcular o *Relative Network Load* (RNL), métrica utilizada para estimar a carga das redes no ambiente. Esta característica possível através do M-MIP, possui a desvantagem do alto consumo energético, além de provocar *overhead* na rede devido à manutenção de múltiplas conexões e o envio de mensagens simultâneas para todas as redes conectadas. No trabalho desta dissertação, este aspecto é inexistente, pois o RSSI é coletado através do próprio

sistema do dispositivo via API, e a vazão das outras redes é obtida através de cada dispositivo conectado em cada uma delas, sendo estes os parâmetros utilizados para seleção de rede. Para seleção de nuvem, o M^2C^2 utiliza mecanismos de sondagem e ranqueamento de recursos remotos (CRS), coletando por exemplo, o nível de utilização da CPU e memória, através de um agente centralizado na rede em intervalos de tempo regulares para calcular suas classificações via MCDM. Neste aspecto, o trabalho proposto se assemelha ao de MITRA et al., pois nessa dissertação, também são realizadas coletas em um nó centralizado, no entanto, através do controlador SDN. Contudo, as métricas capturadas que são utilizadas para o cálculo de custo de *offloading*, variam de acordo com o tipo da aplicação em execução, conforme seus requisitos de QoS mais característicos.

Ravi e Peddoju (2015) abordam o problema de escolher o melhor recurso de nuvem entre múltiplas opções, tais como *cloudlets*, *cloudlet* ad-hoc e nuvem pública como alternativas de operações de *offloading*. O aspecto principal do artigo é reduzir o consumo energético do dispositivo móvel, ao mesmo tempo em que se mantém a disponibilidade do serviço para os usuários. Para isso, os autores propõem um algoritmo que utiliza o *Technique for Order of Preference by Similarity to Ideal Solution* (TOPSIS) para escolher em qual destes recursos, os dados devem ser processados remotamente. Como nessas técnicas de MCDM, a decisão pode ser afetada diretamente de acordo com o modo que foram definidos os pesos, foi utilizada nesta dissertação o *Analytic Hierarchy Process* (AHP), que permite verificar a consistência dos julgamentos realizados nos critérios.

Ainda em RAVI; PEDDOJU, é utilizado um algoritmo para decidir sobre as execuções de *handoff*, onde são calculados o tempo de conexão restante e o consumo energético durante a comunicação com o recurso de nuvem, informações estas que alimentam um sistema *fuzzy*. O *handoff* é disparado nos casos em que o consumo energético for alto ou quando o tempo de conexão com o recurso remoto for baixo. O diferencial do trabalho dessa dissertação é que, além de utilizar o estado da *cloudlet* para seleção de VMs, os graus de pertinência das métricas utilizadas (RSSI e vazão) no sistema *fuzzy* são definidos de modo distinto para cada tipo de aplicação e tamanho dos dados, utilizando a experiência do desenvolvedor do sistema, contribuindo para decisões mais customizadas e adequadas.

A solução de Saad, Kassar e Sethom (2016) propõe uma arquitetura de três camadas contendo o dispositivo móvel, as *cloudlets* e a nuvem pública, respectivamente. Por meio da interação entre estas camadas, os autores visam otimizar o *offloading* e a tomada de decisão para manter o dispositivo sempre conectado à melhor rede e nuvem. O modelo sugere que o dispositivo se conecte a duas *cloudlets*, cada uma atribuindo um papel diferente. A primeira é a *Connecting Cloudlet* (CC) que oferece boas condições de rede e a segunda a *Serving Cloudlet* (SC) capaz de hospedar o serviço e executar o processamento da aplicação móvel. Os autores distinguem estas funções para tratar as causas da degradação das operações de *offloading* de forma independente. A seleção da CC e SC se baseia numa abordagem que considera o tipo de aplicativo e a capacidade das *cloudlets*.

O sistema proposto nesta dissertação se assemelha a este trabalho em discussão no aspecto de tratar as decisões de maneira independente, pois o dispositivo pode permanecer conectado na mesma rede e apenas selecionar uma nova VM na *cloudlet*, ou vice-versa. No entanto, a seleção de rede em SAAD; KASSAR; SETHOM, denominada CC, é feita com base nos parâmetros de largura de banda e latência, não contemplando o RSSI que é fator determinante para proporcionar uma melhor conexão durante a mobilidade do usuário. Esta métrica só entra em ação, caso seja identificada alguma degradação de desempenho do *offloading*, sendo possível que o usuário já se conecte a uma rede com baixo nível de sinal que não seja ideal para a aplicação do usuário. Outro diferencial dessa dissertação é que na seleção de rede, o tamanho dos dados da aplicação são considerados, evitando execuções de *handoffs* desnecessárias para um determinado volume de dados. Na seleção de nuvem (SC), o sistema dos autores calcula para cada *cloudlet* um fator de adequação para hospedar o aplicativo com base em limiares do perfil do aplicativo (por exemplo, largura de banda e o nível da bateria), nessa dissertação, as informações do estado das VMs na *cloudlet* utilizadas na seleção, variam entre o nível da CPU, da memória e *Round Trip Time* (RTT), onde o tipo de aplicação (por exemplo, reconhecimento facial) influencia diretamente nas métricas utilizadas para a tomada de decisão de *handoff*. Além disso, não está claro que o ambiente com várias *cloudlets* como propõe SAAD; KASSAR; SETHOM é viável, pois devido a limitações do simulador CloudSim, apenas disponibiliza um modelo de rede básico com um gerador de tráfego de carga de trabalho limitado, segundo Bahwaireth et al. (2016).

Os trabalhos acima inspiraram a proposta desta dissertação, mas diferem dela de várias formas. Nem todas as estratégias de decisão propostas pelos artigos suportam operações de *offloading* sem a necessidade do reenvio dos dados em função da mobilidade do usuário entre pontos de acesso. Para suprir esta necessidade, o uso do protocolo M-MIP foi proposto em um dos trabalhos, contudo nesta dissertação é utilizado o paradigma SDN que também compõe o sistema de decisão, redirecionando os fluxos de acordo com o novo AP ou VM selecionada, o que evita ações adicionais por parte do dispositivo. Alguns artigos utilizam o modo tradicional de decisão onde os algoritmos são executados no próprio dispositivo móvel, o que implica aumentar o consumo de energia e, assim, reduzir o tempo de vida da bateria do dispositivo. Neste trabalho, as atividades do algoritmo são delegadas ao controlador da rede, que também é responsável por parte da coleta das métricas utilizadas na seleção de rede e VM. Em relação aos critérios utilizados no algoritmo, esta proposta é mais completa, pois também são utilizados o tamanho dos dados da aplicação como critério de decisão, abordagem não contemplada pelos trabalhos relacionados discutidos. Em comparação ao sistema de tomada de decisão proposto, soluções tradicionais de MCDM são utilizadas nos trabalhos de RAVI; PEDDOJU e MITRA et al.. Esta dissertação utiliza a lógica *fuzzy* por ficarem mais claros os limiares adotados para cada aplicação, possibilitando de modo flexível, inserir o conhecimento de especialistas no ambiente projetado e, assim, permitir decisões mais customizadas.

Quadro 3 – Comparação dos trabalhos relacionados

Proposta	**Objetivo Principal**	**Critérios para Seleção**			**Estratégia de Conectividade**	**Mecanismo de Decisão**	**Ambiente de Avaliação**
		Rede	Nuvem	App			
Lee et al. (2013)	Melhorar o desempenho do enlace móvel	Estado do *proxy*, Protocolo NDP			Servidor *Proxy*	Método MCDM	Simulação (JAVA)
Khalaj e Lutfiyya (2013)	Eleição de servidores *proxy*	RTT, Carga do *proxy*			Servidor *Proxy*	Baseado em funções	*Testbed* (Wi-Fi)
Li et al. (2013)	Seleção entre APs com base em predição de mobilidade	Localização, Vazão, Latência	Carga da nuvem		Wi-Fi	Baseado em funções	Simulação (JAVA)
Ravi e Peddoju (2014)	Prover a mobilidade do usuário entre *cloudlets* e eficiência energética.	RSSI, Vazão			Módulos entre o dispositivo e as *cloudlets*	Baseado em funções	*Testbed* (Wi-Fi)
Ravi e Peddoju (2015)	Seleção entre múltiplos recursos de nuvem e economia de energia	RSSI, Energia, Tempo de conexão			Módulos entre o dispositivo e as *cloudlets*	TOPSIS + Lógica *Fuzzy*	*Testbed* (Wi-Fi)
Mitra et al. (2015)	Prover *multihoming* e seleção de rede e nuvem	RSSI, RNL	CPU, Memória		Protocolo M-MIP	Método MCDM	*Testbed* (Wi-Fi)
Saad, Kassar e Sethom (2016)	Seleção de *cloudlets* intermediárias	RSSI, Vazão, Latência	CPU	Tipo	Módulos entre o dispositivo e as *cloudlets*	Baseado em funções	Simulação (CloudSim)
Junior et al. (2017)	Prover transparência de *offloading* e economia energética	Cobertura do AP			Baseado em regras *OpenFlow*	Localização do OFMAG	*Testbed* (*OpenFlow*/Wi-Fi)
Esta dissertação	Seleção de melhor rede e VM, ciente da aplicação móvel e encaminhamento dos resultados de *offloading* entre APs	RSSI, Vazão	CPU, Memória, RTT	Tipo, Dados	Baseado em regras *OpenFlow*	Lógica *Fuzzy* + Cálculo de Custo (AHP)	*Testbed* (*OpenFlow*/Wi-Fi)

3.3 Considerações finais

Este capítulo discutiu trabalhos relacionados ao mecanismo proposto nesta dissertação. Estes trabalhos foram divididos em dois subgrupos de acordo com as contribuições principais de cada um deles, onde foi observada a necessidade de implementação de mecanismos de conectividade em MCC que proporcionem menor *overhead* à rede, bem como sistemas de seleção da melhor rede ou nuvem considerando critérios relevantes para o ambiente móvel e principalmente características da aplicação. O Capítulo 4 apresenta o sistema proposto, incluindo a arquitetura, contribuições e o funcionamento de cada uma das fases do algoritmo de decisão.

4 ARQUITETURA PROPOSTA

Este Capítulo apresenta o sistema proposto nesta dissertação, que tem por objetivo selecionar a melhor rede sem fio disponível e VM na *cloudlet* com base nos requisitos de QoS do tipo da aplicação móvel, visando não degradar a experiência de *offloading* do usuário. A proposta também elabora uma estratégia que evita reenvio e reprocessamento dos dados na *cloudlet* através do redirecionamento do resultado de *offloading*, anterior ao *handoff*, para o AP selecionado.

As subseções deste Capítulo estão organizadas da seguinte forma. A Seção 4.1 apresenta uma visão geral da proposta contendo os elementos que compõem a arquitetura do sistema. Os detalhes do sistema de decisão, explanações sobre cada uma de suas etapas são apresentados na Seção 4.2. A sinalização envolvida na solução é detalhada na Seção 4.3. Por fim, a Seção 4.4 descreve o funcionamento do algoritmo de decisão.

4.1 Arquitetura em Camadas

A arquitetura proposta possui três camadas principais, onde em cada uma delas, um conjunto de requisitos devem ser suportados para atingir os objetivos desejados deste trabalho e caracterizar o ambiente MCC (apresentado na Seção 2.4 dessa dissertação). Na sequência, segue a descrição destes requisitos:

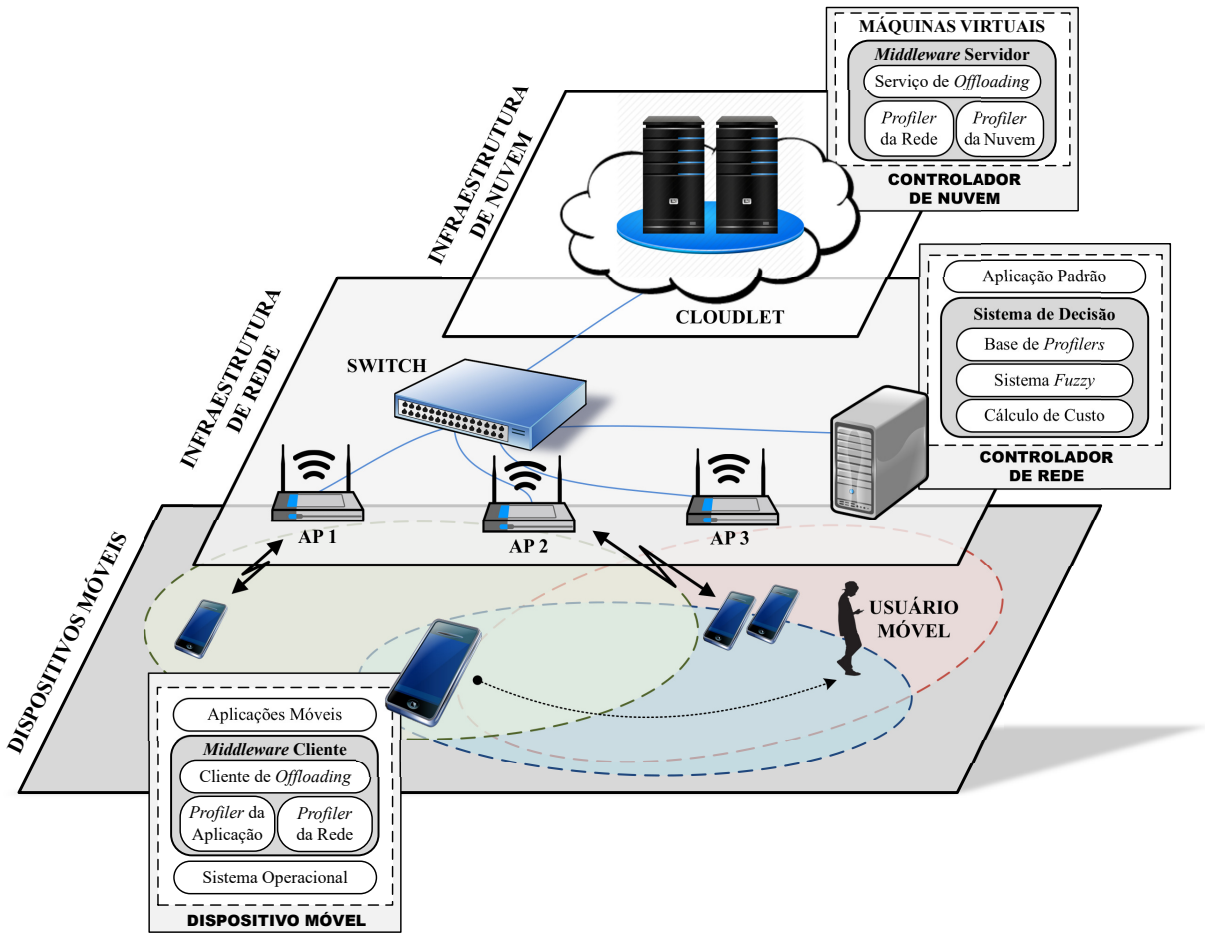
- A camada inferior é denominada de dispositivos móveis, provendo os meios necessários para o funcionamento das aplicações neste ambiente. Para isso, deve suportar plataformas de *software* e *hardware* através de sistemas móveis que possam prover a execução de aplicações e conectividade sem fio. Também necessita suportar mecanismos de *offloading* que, através do sistema operacional móvel, gerenciem serviços específicos para a execução das operações com o recurso de nuvem. Como último requisito, estão os sistemas de monitoramento dinâmico que coletam informações de contexto da rede e dos dados das aplicações, para uso na tomada de decisão.

- A camada intermediária compõe toda infraestrutura de rede responsável por prover a conectividade sem fio dos dispositivos móveis com o recurso de nuvem. Aqui, os requisitos são: controle sobre a rede, prover a conectividade sem fio para a camada de dispositivos móveis e intermediar as conexões com o recurso de nuvem. Além disso, é necessário suportar o gerenciamento dinâmico dos fluxos (encaminhamento de pacotes) durante as execuções de *handoff*, evitando interromper ciclos de *offloading*, o que obrigaria o reenvio dos mesmos dados para a nuvem. Como requisito final, é necessário suportar mecanismos de decisão para processar as demandas do dispositivo, selecionando assim, a rede sem fio e recursos de nuvem ideais.

- Por fim, a camada superior corresponde à infraestrutura de nuvem que contém os recursos computacionais remotos. Os requisitos para o projeto desta camada consistem em: capacidade de gerenciamento de máquinas virtuais (VMs) através de ambientes virtualizados e disponibilização aos usuários móveis destes recursos, contendo serviços de *offloading* disponíveis. Como requisito final, a exemplo da camada de dispositivos móveis, estão os sistemas de monitoramento dinâmico que devem coletar informações do estado das VMs disponíveis, para também serem utilizadas na tomada de decisão.

Os componentes definidos na implementação da arquitetura proposta, visam atender a este conjunto de requisitos que foram apresentados. Estes componentes distribuídos em cada uma das camadas, são apresentados na Figura 10 e, a seguir, de modo mais detalhado, a descrição de cada um deles.

Figura 10 – Arquitetura em Camadas



Fonte: Elaborada pelo Autor

Na camada dos dispositivos móveis, os dispositivos terão as aplicações instaladas, que são executadas utilizando o *Multiplatform Offloading System* (MpOS)[1]. Este sistema fornece um *middleware* cliente que, por padrão, executa um conjunto de serviços (por exemplo, descoberta e conexão com a *cloudlet*) responsáveis por gerenciar as execuções de *offloading* entre o dispositivo e a *cloudlet*. Neste *middleware* também funcionam dois sistemas de *profilers* (da aplicação e da rede), que coletam informações do aplicativo em execução e das redes sem fio disponíveis. Quando o dispositivo solicita ao controlador em qual rede e VM ele deve se conectar para executar o *offloading*, esses dados são enviados como parte dos critérios necessários que são utilizados no sistema de decisão.

A camada de infraestrutura da rede contém APs responsáveis por manter a conexão sem fio com os dispositivos no ambiente móvel, também existe um *switch* responsável por encaminhar estas conexões para a *cloudlet*. Como este ambiente MCC utiliza o paradigma SDN, este conjunto de elementos também funciona via protocolo *OpenFlow* e compõe o plano de dados da solução. O plano de controle é composto por um controlador de rede, que gerencia estes equipamentos através de uma aplicação padrão que insere regras básicas de encaminhamento, e pelo sistema de decisão proposto nesta dissertação. Os elementos que compõem esse sistema são a base de *profilers* que recebe e armazena as métricas enviadas pela *cloudlet* e dispositivos móveis que alimentam o sistema *fuzzy*, responsável pela seleção de rede, e pelo cálculo de custo de *offloading*, que realiza a seleção entre as VMs disponíveis na *cloudlet*.

Finalmente, a camada de infraestrutura de nuvem contém uma *cloudlet* composta de dois servidores gerenciados pelo controlador de nuvem. O primeiro servidor contém o nó de controle, responsável por gerenciar os serviços que permitem a criação de instâncias (VMs) e o segundo, o nó de computação que oferece seus recursos de *hardware* para comportar as VMs em execução. Dentro de cada máquina virtual disponível, funciona o componente que provê o serviço de *offloading* através do *middleware* servidor, responsável por processar as requisições do usuário. Por fim, existem também os sistemas de *profiler* da rede, que trabalham em conjunto com os dispositivos móveis para obter informações da qualidade das redes sem fio, e o *profiler* da nuvem que coleta dados do estado das VMs da *cloudlet*. Estes dados são métricas como nível de processamento e memória, que são enviados ao controlador SDN para complementar as informações recebidas pelo dispositivo do usuário, formando o conjunto de critérios necessários para operação do sistema de decisão.

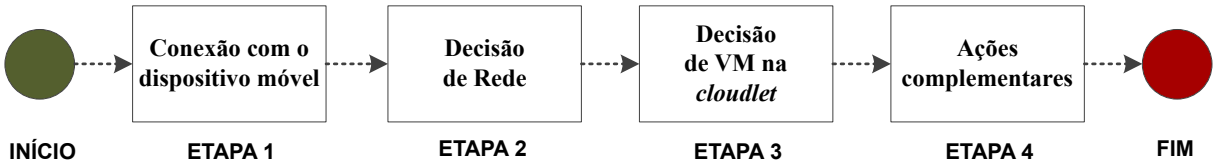
4.2 Visão Geral das Etapas da Proposta

Conforme já foi apresentado, o sistema de decisão proposto nesta dissertação atua no controlador da rede. Esta característica tem o objetivo de usufruir ao máximo da visão global da rede proporcionada pelo paradigma SDN, para coletar as métricas de

[1] MpOS: https://github.com/ufc-great/mpos

decisão necessárias e utilizar o poder computacional de um *hardware* mais robusto para o processamento das atividades do algoritmo de decisão. Dessa forma, o desempenho do sistema não está restrito aos recursos limitados no dispositivo móvel do usuário, como é comum nas soluções de outros trabalhos.

Figura 11 – Etapas da Proposta



Fonte: Elaborada pelo Autor

Outra característica do sistema proposto é que devido ao ambiente intradomínio, o tratamento das decisões é realizado de modo independente. Isto é, o dispositivo móvel do usuário pode executar *handoff* e manter-se conectado à mesma VM da rede anterior, ou selecionar outra VM e permanecer conectado à mesma rede, de acordo com o resultado do sistema de decisão. Para alcançar estes objetivos, foram definidas etapas lógicas para o funcionamento do algoritmo conforme apresentado na Figura 11. A primeira etapa realiza a conexão com o dispositivo móvel e recebe as métricas coletadas por ele. As duas etapas seguintes são responsáveis pela fase de decisão, onde a segunda compõe o sistema *fuzzy* para seleção da rede sem fio e a terceira executa o cálculo de custo de *offloading* em cada uma das VMs na *cloudlet*. A quarta etapa executa ações complementares, como por exemplo, as configuração das regras *OpenFlow* nos elementos da rede para que o ambiente MCC funcione com o novo cenário definido pelas etapas de decisão.

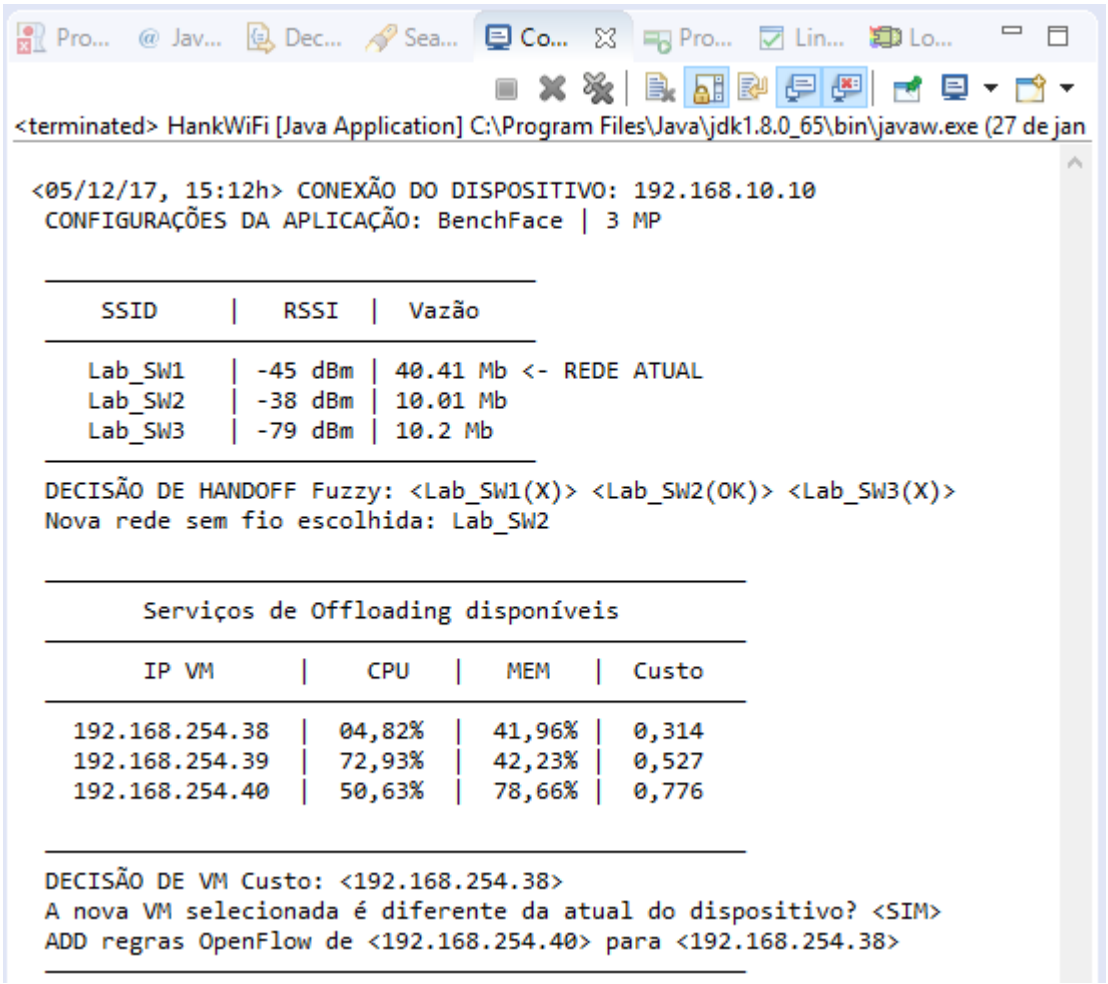
Destaca-se como aspecto principal da proposta, que as decisões realizadas sempre são feitas com base nas características da aplicação móvel em execução no dispositivo. Como cada tipo de aplicação em MCC possui requisitos de QoS distintos, este aspecto do sistema auxilia na tomada de decisões mais eficazes, como por exemplo, identificando que para um tamanho de dados escolhido, a rede atual ainda é capaz de entregar resultados satisfatórios nas execuções de *offloading*, evitando o efeito *ping-pong* entre redes sem fio. Na prática, as etapas de decisão utilizam limiares e métricas relacionados aos critérios considerados como os mais sensíveis de cada tipo de aplicação. De acordo com estes perfis previamente definidos, o algoritmo seleciona dinamicamente estas métricas e limiares para serem utilizados nas etapas de decisão do sistema *fuzzy* e cálculo de custo.

Por fim, destaca-se também a transparência de acesso que funciona na decisão de VM na *cloudlet*. Todo o fluxo de dados de *offloading* entre o dispositivo e a *cloudlet* é gerenciado pelas regras *OpenFlow*. Dessa forma, quando uma nova VM é escolhida pelo sistema, o dispositivo do usuário não fica ciente desta mudança, caso contrário, seria necessário que o *middleware* cliente fosse reconfigurado para a conexão com o novo recurso de nuvem. Com este aspecto da transparência, o fluxo de dados é direcionado à nova VM definida,

sem a necessidade de configurações adicionais no dispositivo móvel pois as novas regras que são adicionadas gerenciam os cabeçalhos de destino e origem dos pacotes.

A Figura 12 apresenta a tela de *log* do sistema de decisão, onde todas as informações da execução do algoritmo são listadas. Pode-se verificar inicialmente as informações do IP do dispositivo móvel, características da aplicação em execução e dados do *Service Set Identifier* (SSID), RSSI e vazão das redes atuais disponíveis, destacando a rede atualmente conectada. Em seguida, o resultado da decisão *fuzzy* que seleciona uma nova rede sem fio e a decisão de VM na *cloudlet* que define a nova VM para as execuções de *offloading*, apresentando as métricas recebidas da *cloudlet*. Nas próximas subseções, são apresentados com mais detalhes o sistema de *profilers* do sistema de decisão, o funcionamento das etapas de decisão da rede e de VM na *cloudlet*, a sinalização estabelecida em ambas as etapas de decisão e por fim, detalhes da implementação do algoritmo utilizado.

Figura 12 – Tela do Sistema de Decisão

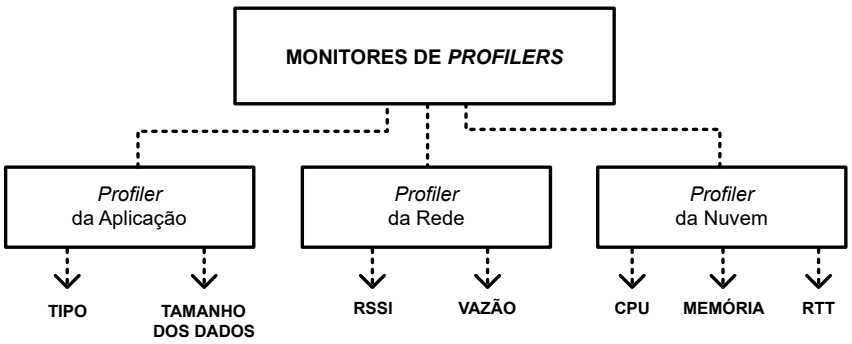


Fonte: Elaborada pelo Autor

4.2.1 Monitores de *Profilers*

Para obter as informações de contexto necessárias para alimentar o sistema de decisão de rede e VM na *cloudlet*, esta proposta utiliza monitores de *profilers*, conforme representado na Figura 13. Estes são responsáveis pela coleta das medições em relação às características do ambiente e estão divididos da seguinte forma: 1) aplicação móvel em execução; 2) interfaces sem fio disponíveis no dispositivo móvel; 3) VMs em execução na *cloudlet* que estejam disponíveis. A seguir, é fornecida uma descrição de cada um destes itens e detalhes de implementação:

Figura 13 – Monitores de *Profilers*



Fonte: Elaborada pelo Autor

- ***Profiler* da Aplicação:** Para que critérios da aplicação sejam considerados na decisão, este componente armazena duas características: **[1] Tipo da aplicação:** o tipo "nuvem", foi a categoria dada às aplicações cujo processamento remoto é mais intensivo, sendo mais afetadas pelo estado da *cloudlet*. Já o tipo "rede", as que enviam e recebem dados durante todo o tempo de execução, sendo mais afetadas por variações deste recurso. **[2] Tamanho dos dados:** a aplicação pode ser executada com diferentes tamanhos de dados, esta informação é coletada para definir os limiares de decisão a serem aplicados a cada um deles.

- ***Profiler* da Rede:** Este componente coleta dois tipos de critérios de cada AP: **[1] Vazão:** calcula a largura de banda atual da rede sem fio à qual o dispositivo móvel está conectado, obtendo o resultado em Mbps. **[2] RSSI:** armazena a potência do sinal sem fio recebido pelo dispositivo em dBm, de todos os APs disponíveis no ambiente através da classe `WifiManager`[2], via API Android.

- ***Profiler* da Nuvem:** Coleta três critérios do estado da *cloudlet*: **[1] CPU** e **[2] Memória:** monitoram respectivamente estatísticas do nível de utilização das vCPUs e de memória RAM das VMs em porcentagem (%), através do uso da API Java SIGAR[3]. **[3] RTT:** obtém o tempo médio do envio e resposta de uma solici-

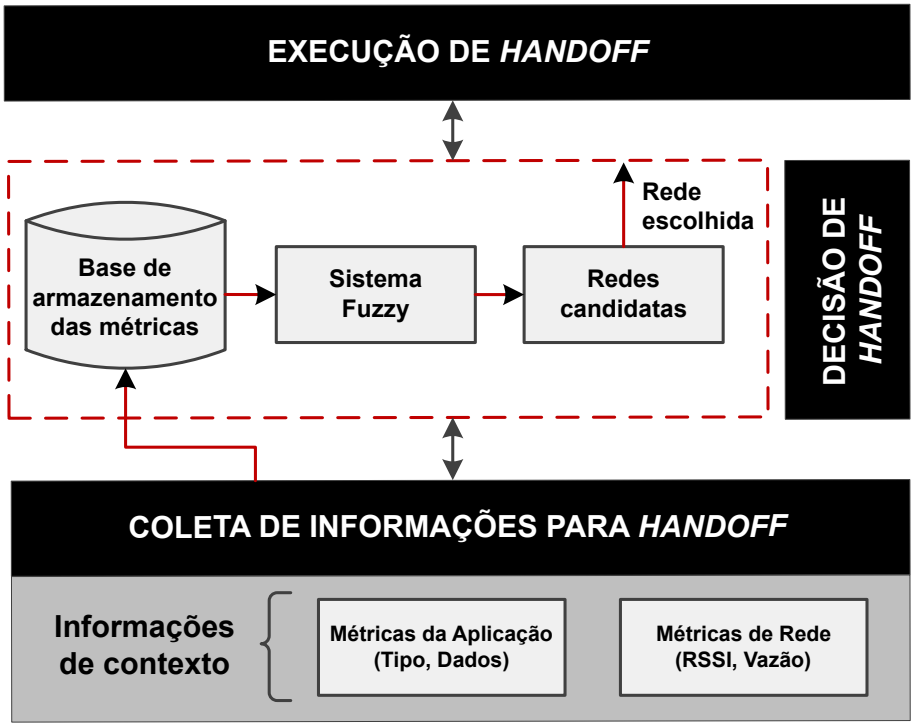
[2] WifiManager: https://developer.android.com/reference/android/net/wifi/WifiManager.html
[3] SIGAR: https://github.com/hyperic/sigar

tação do dispositivo para as VMs em milissegundos, através de mensagens *Internet Control Message Protocol* (ICMP), pela classe `InetAddress`[4], via API Android.

4.2.2 Seleção de rede

A seleção de rede é a segunda etapa da solução que é executada no fluxo do algoritmo. Nesta etapa é definida a rede sem fio na qual o dispositivo deve permanecer ou se conectar, para realizar as execuções de *offloading*. A estratégia de decisão utilizada neste estágio foi o sistema *fuzzy*, onde foram definidos os graus de pertinência para cada critério considerado na seleção, de acordo com o tipo da aplicação e seu tamanho de dados. O tipo das funções de pertinência escolhido foi o trapezoidal, bastante empregado na maioria das aplicações práticas, de acordo com Barua, Mudunuri e Kosheleva (2014). Para aplicar esta técnica em nosso sistema, foi utilizada a biblioteca jFuzzyLogic (CINGOLANI; ALCALA-FDEZ 2012 e 2013) que permite implementar a Lógica *Fuzzy* na linguagem de programação JAVA.

Figura 14 – Fases da decisão de rede para *handoff*



Fonte: Elaborada pelo Autor

Para mais detalhes do funcionamento desta etapa, a Figura 14 apresenta o fluxo das fases da decisão de rede para *handoff* que acontecem no dispositivo móvel e no sistema de decisão. A primeira fase é a coleta de informações, onde o dispositivo móvel captura as métricas utilizadas nesta fase de decisão através dos *profilers*, para alimentar estas informações na base de armazenamento presente no sistema de decisão. As métricas coletadas

[4] InetAddress: https://developer.android.com/reference/java/net/InetAddress.html

são o tipo da aplicação, o tamanho dos dados de *offloading*, e o nível do RSSI e vazão das redes disponíveis. O sistema *fuzzy* recebe os parâmetros destas métricas e carrega as funções de pertinência de acordo com as características da aplicação em execução. O resultado do sistema *fuzzy* indica quais redes sem fio são consideradas candidatas, ou seja, as redes que segundo as regras *fuzzy* apresentam condições favoráveis para uma experiência eficaz de *offloading*. Caso mais de uma rede seja considerada candidata, é escolhida a que tiver o menor número de usuários conectados naquele momento. Por fim, a fase de execução acontece no dispositivo móvel, que recebe o SSID escolhido pelo sistema de decisão para, em seguida, disparar o *handoff*.

4.2.3 Seleção de VM na *cloudlet*

A seleção de VM na *cloudlet* é a terceira etapa a ser executada no sistema de decisão. A estratégia utilizada foi uma fórmula para calcular o custo de *offloading* para cada VM em execução na *cloudlet*, onde são utilizados critérios com pesos associados a cada um, permitindo balancear suas características. O resultado desse cálculo permite a classificação das VMs para identificar qual possui o menor custo e que, consequentemente, será a escolhida para receber as solicitações de *offloading* do dispositivo móvel do usuário. Para prover os meios deste novo acesso, novas regras *OpenFlow* de maior prioridade do que as já inseridas pela aplicação padrão, são adicionadas para redirecionar o tráfego para o novo recurso de nuvem selecionado. Dessa forma, o usuário não tem conhecimento do local que suas tarefas estão sendo processadas, proporcionando assim a transparência de acesso. A seguir, são apresentados detalhes de implementação desta etapa.

$$Ct = (pA \times nA) + (pB \times nB) + (pC \times nC) \tag{4.1}$$

A fórmula de custo (Ct) é apresentada na Equação 4.1 e é composta por três métricas normalizadas (n) através do método min-max (HAN; PEI; KAMBER, 2011), que são multiplicadas pelos seus respectivos pesos (p). O primeiro critério (A) corresponde ao tempo médio de processamento de uma execução de *offloading*, possibilitando que o resultado do cálculo seja proporcional ao poder de processamento do tipo da VM com quantidades de vCPUs distintas. Diferente das outras métricas que são capturadas de modo dinâmico através dos *profilers*, este valor médio inserido na fórmula é obtido através da experiência, por medições prévias e variam de acordo com cada tamanho de dados utilizados na aplicação móvel, apresentado com mais detalhes na Seção 5.2. O segundo critério (B) corresponde à porcentagem de utilização da CPU calculada para cada VM na *cloudlet* e por fim, o último critério (C) varia de acordo com o parâmetro de QoS mais prioritário para o tipo de aplicação. Conforme a classificação atribuída pelo *profiler* da aplicação,

caso "nuvem" o critério definido na fórmula será o nível de utilização da memória da VM e caso "rede", o tempo de RTT.

Para definir os pesos utilizados para balancear os critérios da fórmula (pA, pB e pC) foi aplicado o AHP (ZANAKIS et al., 1998). Este método permite realizar comparações em pares de todos os critérios existentes na solução a ser utilizada, com base em uma escala padronizada contendo nove níveis, apresentada no Quadro 4.

Quadro 4 – Escala de valores comparativos

Intensidade de importância	**Definição**
1	Mesma importância
3	Importância pequena de um sobre o outro
5	Importância grande ou essencial
7	Importância muito grande ou demonstrada
9	Importância absoluta
2, 4, 6, 8	Valores intermediários entre valores adjacentes

A Figura 15 apresenta a matriz de decisão obtida a partir do uso da técnica AHP. Esta matriz contém os valores da comparação em pares de cada um dos critérios (A, B e C) com base nos julgamentos realizados, que serão aplicados na fórmula de custo de *offloading*.

Figura 15 – Matriz de decisão

$$X = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1 & 2 & 2 \\ \frac{1}{2} & 1 & 2 \\ \frac{1}{2} & \frac{1}{2} & 2 \end{pmatrix} \end{array}$$

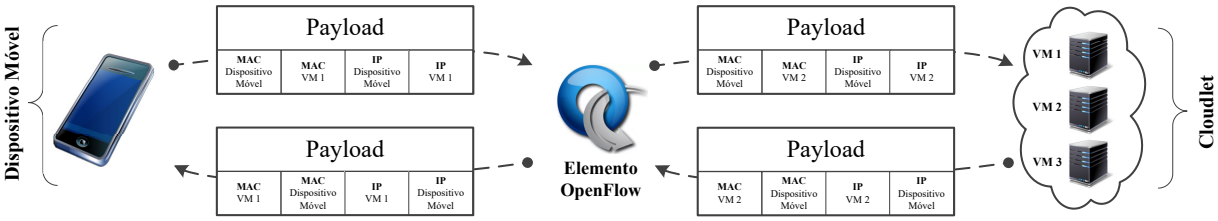
Em seguida, é necessário verificar a consistência dos julgamentos realizados, o AHP gera um índice chamado de *Consistency Ratio* (CR), onde o valor CR deve ser inferior a 0,1 para que as saídas de pesos relativos sejam consideradas válidas. Estes valores obtidos são apresentados no Quadro 5, que contém o peso para cada critério e a taxa de consistência obtida para todos eles.

Quadro 5 – Pesos dos critérios

Critério	**Peso**	**CR**
(A) Tempo de processamento	0,493	0,056
(B) Nível da CPU	0,311	
(C) Nível da Memória / Valor do RTT	0,196	

Ao contrário do processo da seleção de rede onde o resultado da decisão é finalizado pela execução de *handoff*, na etapa da seleção de VM na *cloudlet*, após obter a VM escolhida são executadas ações complementares que correspondem à adição de regras *OpenFlow*. De modo resumido, um pacote de rede contém o *payload* e os campos de destino e origem (camada 2 e 3) que são utilizados pelos elementos da rede para direcionar o fluxo de comunicação. No ambiente apresentado, estes campos são preenchidos de acordo com as informações dos endereços *Media Access Control* (MAC) e IP do dispositivo móvel e das VMs na *cloudlet*, de acordo com o cenário de comunicação definido, a fim de proporcionar as execuções do *offloading*. Nas ações complementares aplicadas nesta etapa, quando um novo recurso de nuvem é escolhido, são modificadas as informações de origem e destino dos pacote através de ações definidas nas regras *OpenFlow*. Com isso, os dados de *offloading* são encaminhados para a nova VM sem a necessidade de ações adicionais por parte do dispositivo móvel do usuário.

Figura 16 – Exemplo de alteração de campos do pacote



Fonte: Elaborada pelo Autor

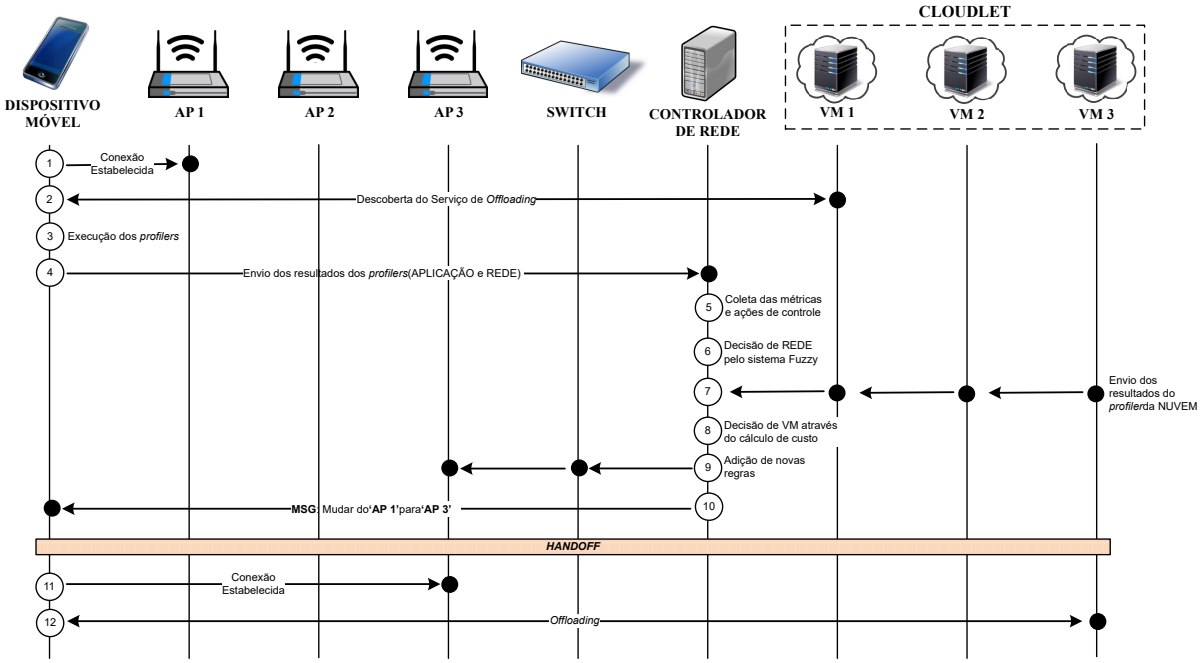
A Figura 16 apresenta um exemplo da aplicação desta estratégia utilizada na etapa de decisão de VM. O cenário contém o dispositivo móvel, a *cloudlet* com três VMs em execução e o elemento *OpenFlow*, que pode ser um AP ou *switch*. Conforme o cenário, na execução de *offloading* os campos de destino do pacote que sai do dispositivo aponta para a VM 1, que foi definida inicialmente pelo serviço do *middleware* cliente. Como neste exemplo o sistema de decisão de VM na *cloudlet* selecionou a VM 2, são adicionadas ao elemento *OpenFlow* na rede através do controlador, as regras contendo as ações necessárias para substituir os campos de origem e destino pelo IP e MAC da VM 2. Com este procedimento, as requisições são direcionadas para o novo recurso de nuvem escolhido, sem o conhecimento do dispositivo móvel. Em seguida, o resultado do processamento da solicitação é devolvido, porém é necessário que os campos de origem dos pacotes da VM 1 sejam reinseridos neste fluxo para que as conexões do dispositivo móvel recebam os dados processados. Para isso, o elemento *OpenFlow* desfaz as ações anteriores, reinserindo as informações originais de IP e MAC da VM 1 nos pacotes, permitindo o recebimento da solicitação pelo usuário.

4.3 Sinalização da Proposta

Esta Seção apresenta o esquema de sinalização que ocorre durante o funcionamento da proposta. Existem basicamente quatro tipos de interações, duas iniciadas pelo dispositivo móvel e as duas restantes pelo sistema de decisão: 1) O dispositivo móvel se comunica com a *cloudlet* para a execução de *offloading*; 2) O dispositivo móvel envia ao sistema de decisão as métricas capturadas e recebe o resultado da seleção de rede; 3) O sistema de decisão consulta a *cloudlet* para coletar as métricas do estado das VMs; 4) O sistema de decisão se comunica com o controlador da rede com objetivo de adicionar as novas regras de encaminhamento nos elementos *OpenFlow*.

A Figura 17 detalha as trocas de mensagens durante o processo da seleção da rede e VM na *cloudlet* e, a seguir, a explicação de um exemplo contendo cada mensagem e evento mostrado durante o processo.

Figura 17 – Sinalização da Proposta



Fonte: Elaborada pelo Autor

(1) Inicialmente, o dispositivo móvel se conecta a uma determinada rede sem fio, que no exemplo apresentado é representada pelo AP 1 e, em seguida, o usuário inicializa a aplicação móvel para a qual que deseja realizar *offloading*.

(2) A descoberta de serviços de *offloading* presente no *middleware* cliente é iniciada com o objetivo de localizar na *cloudlet* alguma VM que esteja apta para receber as solicitações de processamento remotas da aplicação escolhida pelo usuário. Na *cloudlet* presente no ambiente, as três VMs em execução estão preparadas para

receber estas solicitações, porém a escolhida foi a VM 1 por ter sido a primeira a ter sua resposta alcançada pelo *middleware* cliente.

(3) O dispositivo inicia as execuções dos *profilers*. No da aplicação, são verificados tipo e tamanho dos dados do aplicativo móvel para a execução do *offloading*. No da rede, são coletados os SSIDs e RSSIs do AP 1, AP 2 e AP 3. Para que o dispositivo móvel não precise se conectar em todas as redes para obter a vazão média, ele captura apenas a rede à qual está atualmente conectado. As vazões dos APs restantes serão informadas ao sistema de decisão através dos outros clientes conectados em cada uma destas redes.

(4) Todas estas informações e resultados coletados são reunidos. Em seguida, o dispositivo se conecta ao controlador da rede para solicitar a seleção de rede e VM na *cloudlet* pelo sistema de decisão, enviando estes dados obtidos pelos *profilers*.

(5) O sistema de decisão recebe e reúne estes resultados e, inicialmente, atualiza as informações de controle como por exemplo, em qual AP o dispositivo está conectado para que na etapa final do algoritmo, sejam adicionadas regras *OpenFlow* específicas ou o registro da VM configurada para o dispositivo móvel. Em seguida, este conjunto de informações segue para as etapas de decisão da rede e VM na *cloudlet*.

(6) A primeira etapa de decisão é iniciada através da seleção de rede. O sistema *fuzzy* é alimentado com os dados de contexto referentes às redes sem fio capturadas pelos *profilers*. De acordo com a aplicação, ao cliente é atribuída uma nova rede onde seu SSID é armazenado para uso em etapas seguintes.

(7) O sistema de decisão consulta os resultados do *profiler* da nuvem em cada uma das VMs existentes na *cloudlet*. Estes dados são fornecidos para a etapa seguinte de acordo com o tipo de aplicação em execução no dispositivo móvel.

(8) Para a seleção de VM na *cloudlet*, a segunda etapa de decisão é iniciada. O cálculo de custo recebe os dados coletados das VMs, selecionados de acordo com o tipo da aplicação e como resultado é obtida a VM com o menor custo de *offloading*.

(9) Os resultados das decisões de rede e VM na *cloudlet* são reunidos e as ações de responsabilidade das regras *OpenFlow* são realizadas através do controlador de rede. No exemplo, foi selecionado o AP 3 e a instância da *cloudlet* a VM 3. Com base nestas decisões, regras de encaminhamento são inseridas no *switch* para redirecionar o fluxo de dados da antiga rede para a nova rede, e inseridas também no novo AP para reescrever os cabeçalhos de origem e destino permitindo que o dispositivo móvel alcance a nova VM escolhida.

(10) Como mensagem de retorno, o sistema de decisão envia para o dispositivo o SSID da nova rede alvo para execução do *handoff*. O *middleware* cliente recebe esta informação e aciona as ações necessárias para realizar a mudança de rede por API Android.

(11) O dispositivo móvel se conecta no AP da rede indicada no passo anterior, onde as regras de encaminhamento *OpenFlow* foram inseridas pelo sistema de decisão para redirecionar os pacotes para a nova VM.

(12) O *middleware* cliente dispara a execução de *offloading* e recebe os dados de processamento da nova VM selecionada através da nova rede sem fio escolhida pelo sistema de decisão.

4.4 Algoritmo de Decisão

O algoritmo de decisão desta proposta comanda o fluxo de atividades que é seguido pelo sistema de decisão, cujo pseudocódigo é apresentado a seguir. O Quadro 6 apresenta a legenda dos símbolos dos parâmetros e variáveis do algoritmo.

Quadro 6 – Símbolos usados como parâmetros e variáveis para o algoritmo

Símbolo	**Descrição**
DM_{IP}	IP de um determinado dispositivo móvel na rede.
$Dados_{DM}[\,]$	Todos os dados recebidos do dispositivo móvel.
IPr_{DM}	IP atual do dispositivo móvel.
$SSIDr_{WLAN}$	Rede sem fio atual do dispositivo móvel.
$RSSIr_{WLAN}$	Nível de sinal da rede atual do dispositivo móvel.
BWr_{WLAN}	Vazão da rede sem fio atual do dispositivo móvel.
$Lista_{WLANs}[SSID, RSSI]$	Nome e nível de sinal das redes do dispositivo móvel.
IPr_{VM}	IP da VM descoberta através do *middleware*.
APP_{TIPO}	Tipo da aplicação no dispositivo móvel.
APP_{DADOS}	Tipo dos dados da aplicação no dispositivo móvel.
$Lista_{DMs}[IP, IP]$	IP de cada VM definida para cada dispositivo móvel.
$Lista_{BWs}[SSID, BW]$	Nome das redes sem fio e a vazão atual de cada uma.
$ListaFuzzy_{APs}[SSID, boolean]$	Resultados *boolean* de cada uma das redes sem fio.
$ResultadoAtualSSID$	Resultado *boolean* da rede atual do dispositivo móvel.
$RespostaSSID$	Nome da rede escolhida para o *handoff*.
$Lista_{VMs}[IP, CPU, MEM, RTT]$	IP, nível da CPU, da memória e valor RTT das VMs.
$CLOUDLET_{VM}$	VM de uma determinada *cloudlet*.
$ListaCusto_{VMs}[IP, inteiro]$	Valores de custo de cada VM na *cloudlet* por IP.
$IPMenorCusto_{VM}$	IP da VM com menor custo.

A primeira atividade do algoritmo é executar o método `ConectarDM` que inicia um *socket* TCP para aceitar as conexões dos dispositivos móveis (DM_{IP}). Assim que uma conexão é realizada, são recebidos dados que serão usados na tomada de decisão de melhor

Algoritmo 1: Pseudocódigo do Algoritmo de Decisão

```
1  repita
2      ConectarDM();
3      Dados_DM[ ] ← IPr_DM, SSIDr_WLAN, RSSIr_WLAN, BWr_WLAN,
         Listar_WLANs[ ], IPr_VM, APP_TIPO, APP_DADOS do DM_IP;
4      se Lista_DMs[ ] não contém IPr_DM então
5          Lista_DMs[ ] ← IPr_DM, IPr_VM;
6      fim se
7      Lista_BWs[ ] ← SSIDr_WLAN, BWr_WLAN;
8      ListaFuzzy_APs[ ] ← DecisãoFuzzy(APP_TIPO, APP_DADOS, RSSIr_WLAN,
         BWr_WLAN, Lista_WLANs[ ], Lista_BWs[ ]);
9      ResultadoAtualSSID ← ListaFuzzy_APs[ ];
10     se ResultadoAtualSSID = false então
11         RespostaSSID ← BuscarRede(ListaFuzzy_APs[ ]);
12     senão
13         RespostaSSID ← SSIDr_WLAN;
14     fim se
15     selecione APP_TIPO faça
16         caso "Nuvem" faça
17             Lista_VMs[ ] ← IPr_VM, CPUr_VM, MEMr_VM da CLOUDLET_VM;
18             ListaCusto_VMs[ ] ← CalcularCusto(APP_TIPO, APP_DADOS,
                 Lista_VMs[ ]);
19         caso "Rede" faça
20             Lista_VMs[ ] ← IPr_VM, CPUr_VM, RTTr_VM da CLOUDLET_VM;
21             ListaCusto_VMs[ ] ← CalcularCusto(APP_TIPO, APP_DADOS,
                 Lista_VMs[ ]);
22         fim caso
23     fim selecione
24     VmMenorCusto_IP ← MenorCusto(ListaCusto_VMs[ ]);
25     se RespostaSSID = SSIDr_WLAN e IPMenorCusto_VM = IP_VM então
26         EnviarResposta(RespostaSSID) para DM_IP;
27     senão
28         Lista_DMs[ ] ← IPr_DM, IPMenorCusto_VM;
29         AdRegrasRyu(RespostaSSID, IPr_DM, IPMenorCusto_VM);
30         EnviarResposta(RespostaSSID) para DM_IP;
31     fim se
32 até ∞;
```

rede e VM na *cloudlet* para o *offloading*. Estes dados contêm informações e métricas de contexto, a saber: o IP do dispositivo (IPr_{DM}), o nome ($SSIDr_{WLAN}$), o nível de sinal ($RSSIr_{WLAN}$) e a vazão (BWr_{WLAN}) da rede sem fio conectada, a lista do nome e nível de sinal de todas as outras redes sem fio no ambiente do dispositivo ($Lista_{WLANs}[\,]$), o IP da VM descoberta através do *middleware* cliente (IPr_{VM}), e o tipo (APP_{TIPO}) e tamanho dos dados (APP_{DADOS}) da aplicação em execução. No pseudocódigo, estas informações

são armazenadas dentro da lista $Dados_{DM}[\,]$ para centralizar toda a coleta. Em seguida, uma condicional "SE" verifica se já foi feito o mapeamento do IP do dispositivo com uma VM na lista $Lista_{DMs}[\,]$. Esta checagem é necessária, pois as próximas atualizações só deverão ser feitas pelo algoritmo após as etapas de decisão para o controle e inserção das regras *OpenFlow*. Após esta atividade, a lista $Lista_{BWs}[\,]$ é atualizada. Esta lista contém a vazão de todas as redes sem fio obtidas através das medições dos dispositivos no ambiente de execução.

Após a etapa inicial de armazenamento e controle de informações, é iniciada a seleção de rede através do sistema *fuzzy*. Para isso, o método `DecisãoFuzzy` é executado. Este método recebe os parâmetros APP_{TIPO} e APP_{DADOS} para que sejam selecionadas as funções de pertinência referentes a estas características da aplicação, e também recebe as métricas de rede contidas nas variáveis $RSSIr_{WLAN}$, BWr_{WLAN}, $Lista_{WLANs}[\,]$ e $Lista_{BWs}[\,]$. O retorno deste método alimenta a lista $ListaFuzzy_{APs}[\,]$ contendo um valor *boolean* para cada rede verificada. Caso *true*, significa que é uma rede candidata para disparar o *handoff* no dispositivo móvel e caso *false*, que não deve ser considerada na seleção. Em seguida, na linha 9, a variável *ResultadoAtualSSID* recebe o valor retornado pela rede sem fio atual do dispositivo, onde a condicional "SE" verifica o valor recebido. Se recebeu o valor *true*, a variável *RespostaSSID* recebe o $SSIDr_{WLAN}$, que indicará no final das etapas de decisão que o dispositivo permanecerá na rede atual. Caso tenha recebido *false*, o método `BuscarRede` busca na lista $ListaFuzzy_{APs}[\,]$ o nome da rede que recebeu valor favorável para executar o *handoff*, e terá seu SSID armazenado na variável *RespostaSSID*.

Continuando a execução do algoritmo, é iniciada a etapa de seleção de VM na *cloudlet* através do cálculo de custo de *offloading*. Para definir quais critérios das VMs que serão considerados na fórmula, o valor do parâmetro APP_{TIPO} é verificado. Caso tenha recebido o valor "Nuvem", significa que a aplicação em execução no dispositivo móvel é mais sensível a métricas de poder de processamento e será considerado como terceiro critério o nível de utilização de memória. Caso "Rede", demonstra que métricas de tempo de resposta no ambiente sem fio são mais prioritárias, e será considerado o RTT. O primeiro critério será o valor médio do tempo de execução da aplicação obtido através da experiência e o segundo critério o nível de processamento atual da CPU. As informações de métricas das VMs são armazenadas na lista $Lista_{VMs}[\,]$, que em seguida é utilizada como parâmetro junto com APP_{TIPO} e APP_{DADOS} no método `CalcularCusto`. Este método alimenta a lista $ListaCusto_{VMs}[\,]$, contendo o custo de *offloading* para cada VM disponível na *cloudlet* que por fim, é usada como parâmetro do método `MenorCusto` para retornar na variável $VmMenorCusto_{IP}$ o IP da VM com o menor custo calculado.

As últimas atividades do algoritmo são responsáveis por viabilizar o funcionamento da nova configuração definida pelas etapas de decisão. A condicional "SE" verifica se o SSID da rede escolhida é igual ao da atual conectada e se o IP da VM com menor custo é igual ao

da descoberta através do *middleware* cliente. Caso estas duas condições sejam atendidas simultaneamente, significa que não serão adicionadas regras de encaminhamento por parte do sistema de decisão e o método `EnviarResposta` retorna para o dispositivo móvel que ele deve permanecer na rede atual. Caso alguma condição não seja atendida, é necessário inserir novas regras *OpenFlow*. Para isso, a lista $Lista_{DMs}$ é atualizada com o novo IP da VM selecionada para o dispositivo móvel e o método `AdRegrasRyu` recebe os parâmetros *RespostaSSID* para indicar qual AP deve ser inseridas as regras, indicando em suas ações o IPr_{DM} e $VmMenorCusto_{IP}$. Por fim, o método `EnviarResposta` dessa vez irá retornar ao dispositivo o SSID da rede em que deverá ser disparado o *handoff*.

4.5 Considerações finais

Este Capítulo apresentou os principais elementos da proposta apresentadas nesta dissertação, detalhando os componentes que a compõem, bem como de cada um dos componentes do sistema de decisão, o esquema de sinalização e o algoritmo proposto que gerencia as etapas do sistema. No Capítulo 5, a proposta será avaliada através de um ambiente experimental e serão definidos os limiares de decisão e métricas utilizadas.

5 AVALIAÇÃO EXPERIMENTAL

Neste Capítulo são realizadas as avaliações com o objetivo de validar o sistema de decisão proposto nessa dissertação. A Seção 5.1 apresenta a metodologia incluindo informações técnicas dos equipamentos utilizados no *testbed*, aplicações de *benchmark* e descreve as métricas utilizadas. A Seção 5.2 apresenta os limiares de decisão e a Seção 5.3 os resultados dos experimentos contendo especificações de como foram realizados os testes.

5.1 Metodologia

Para a avaliação da proposta desta dissertação, foi utilizado o ambiente real de um Laboratório do Centro de Informática da UFPE, que se trata de um ambiente de seis por nove metros quadrados. Este ambiente foi utilizado para realizar dois experimentos, que são descritos na Seção 5.4. Além disso, um conjunto de equipamentos foram utilizados, além de duas aplicações distintas e suas respectivas métricas, maiores detalhes são apresentados a seguir.

5.1.1 Especificação dos equipamentos

Um conjunto de equipamentos foi utilizados para montar o *testbed* com objetivo de caracterizar o ambiente MCC. O Quadro 7 contém um resumo de informações técnicas dos elementos utilizados.

O ambiente possui três pontos de acesso sem fio, que são roteadores com *firmware* padrão substituídos pelo OpenWrt[1]. Este novo sistema permite que estes equipamentos sejam compatíveis com o protocolo *OpenFlow*, através da instalação do Open vSwitch[2] para possibilitar a integração com o ambiente SDN. É importante destacar que esta instalação foi feita no nível do *kernel*, permitindo que os APs obtenham maior desempenho em ambiente de produção como por exemplo, suportando maior vazão, conforme os resultados dos testes realizados por JÚNIOR (2016).

A *cloudlet* que disponibiliza os serviços de nuvem é composta de dois servidores onde funcionam os nós de controle e computação, responsáveis pelo funcionamento do gerenciador de nuvem OpenStack. O controlador de rede contém o *framework* Ryu que gerencia todo o encaminhamento de pacotes na rede *OpenFlow*, e também é onde atua o sistema de decisão proposto nesta dissertação. O sistema operacional instalado no servidor de nuvem e controlador da rede é a distribuição Linux Ubuntu. O ambiente também possui

[1] OpenWrt - https://openwrt.org/
[2] Open vSwitch - http://openvswitch.org/

Quadro 7 – Configuração de *hardware* e *software* do *testbed*

Ambiente de Execução	**Configurações de *hardware***	**Configurações de *software***
Cloudlet	Nó de controle: Intel® Xeon® E5649 Memória RAM 6GB	Ubuntu 16.04.2 LTS OpenStack Ocata
	Nó de computação: Intel® Xeon® E5649 Memória RAM 12 GB	Ubuntu 16.04.2 LTS OpenStack Ocata
Controlador de rede	Processador Intel i7 Memória RAM 8GB	Ubuntu 16.04.2 LTS Controlador Ryu v4.21 Sistema de Decisão Aplicação padrão
Pontos de Acesso	TP-LINK AC1200	OpenWrt 15.05 Open vSwitch 2.3.90
Switch	Switch Extreme X440-24p-10 G	ExtremeXOS *OpenFlow* 1.3

um *switch* já compatível de fábrica com o protocolo *OpenFlow*, responsável por interligar os APs com a *cloudlet*.

5.1.2 Aplicações de *Benchmark*

Para a execução dos testes, duas aplicações para a plataforma Android foram utilizadas. A primeira é uma aplicação de detecção de faces e foi desenvolvida pelo grupo de pesquisa IANG[3], enquanto a segunda, é uma aplicação de processamento de movimentos de esferas aleatórias em tempo real, e está disponível na *web* para fins acadêmicos. A Figura 18 mostra a tela dessas aplicações que são descritas a seguir:

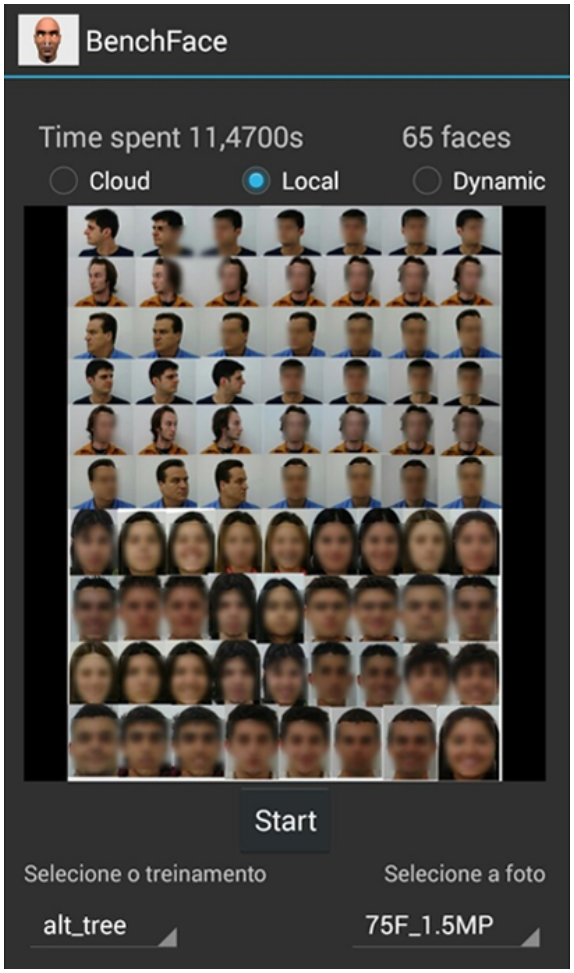
- *BenchFace*[4] é uma aplicação de detecção de faces que usa *Haar features* baseado em classificadores em cascata, um método proposto pelos pesquisadores Viola e Jones (2001). O algoritmo de detecção de faces usa uma abordagem baseada em aprendizagem de máquina, onde funções em cascata são treinadas com um conjunto de imagens positivas (imagens que contêm faces) e negativas (imagens que não possuem faces). O aplicativo é composto de uma única imagem com 78 faces em diferentes ângulos. O usuário pode alterar a mesma imagem para diferentes resoluções e algoritmos classificadores em cascata. Sua execução pode ocorrer na nuvem e local. Ao final da execução é exibida na tela a quantidade de faces detectadas e o tempo decorrido para detectá-las.

[3] IANG - http://www.cin.ufpe.br/~iang/
[4] BenchFace - http://cin.ufpe.br/~bjcc

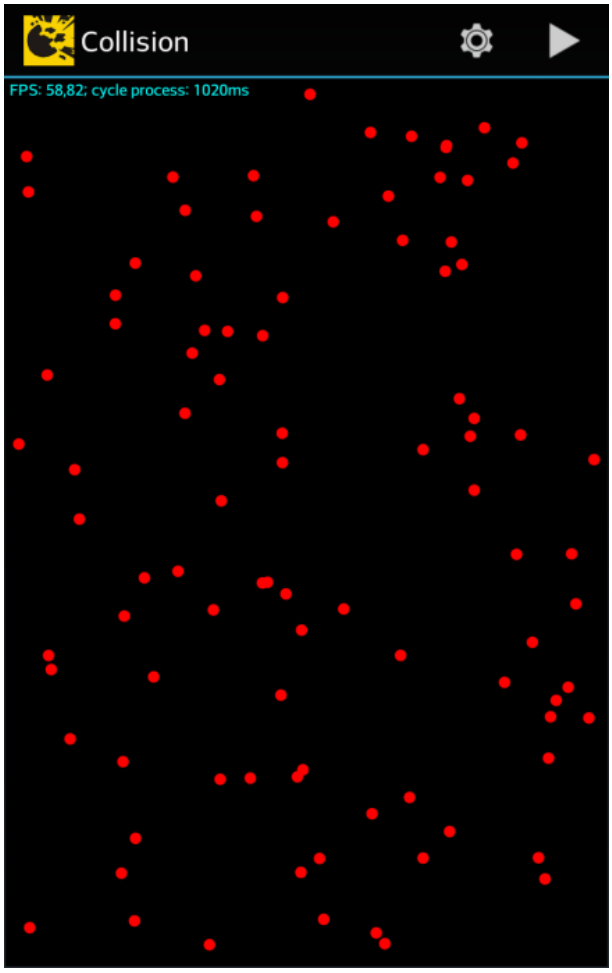
- *CollisionBall*[5] é uma aplicação de processamento em tempo real, que através de cálculos matemáticos, processa uma quantidade de esferas pré-definidas pelo usuário e detecta as colisões entre elas. O usuário pode escolher entre 250 até 1500 esferas, podendo também iniciar a execução de um lugar fixo ou aleatório da tela do dispositivo móvel. Durante sua execução, é exibida na tela a quantidade de *frames* renderizados por segundo (FPS) pelo programa, onde esta quantidade está diretamente relacionada com a qualidade da rede sem fio e disponibilidade dos recursos remotos, uma vez que esta aplicação realiza todo seu processamento através de interação contínua com a nuvem. Contudo, sua execução também pode ocorrer localmente, onde seu desempenho (FPS) será reflexo do poder de processamento do dispositivo.

Figura 18 – Tela das aplicações de *benchmark*

(a) BenchFace



(b) CollisionBall



Fonte: Elaborada pelo Autor

[5] CollisionBall - https://github.com/ufc-great/mpos/tree/master/android/ParticleCollision

5.1.3 Métricas

Nesta Seção são apresentadas as métricas utilizadas para avaliar o desempenho das aplicações através do sistema proposto nesta dissertação. Como cada aplicação possui características distintas, foram definidas para o BenchFace o tempo de execução médio e para o Collision a quantidade de FPS e perda média de pacotes. A seguir, estas métricas são apresentadas com mais detalhes.

- **Tempo de Execução:** é o tempo total que a aplicação levou para o usuário iniciar o envio da solicitação de *offloading* para a *cloudlet*, até o momento que o resultado do processamento é apresentado na tela do dispositivo móvel.
- **FPS:** é a quantidade de *frames* que a aplicação consegue renderizar através de múltiplos envios pela interação com a *cloudlet*, proporcionando ao usuário um fluxo constante de imagens na tela do dispositivo móvel.
- **Perda de Pacotes:** é a métrica de QoS que retorna a quantidade de pacotes perdidos durante o envio do dispositivo móvel para a nuvem e vice versa. Quanto menor a perda de pacotes melhor é a experiência do usuário.

5.2 Definição dos limiares de decisão

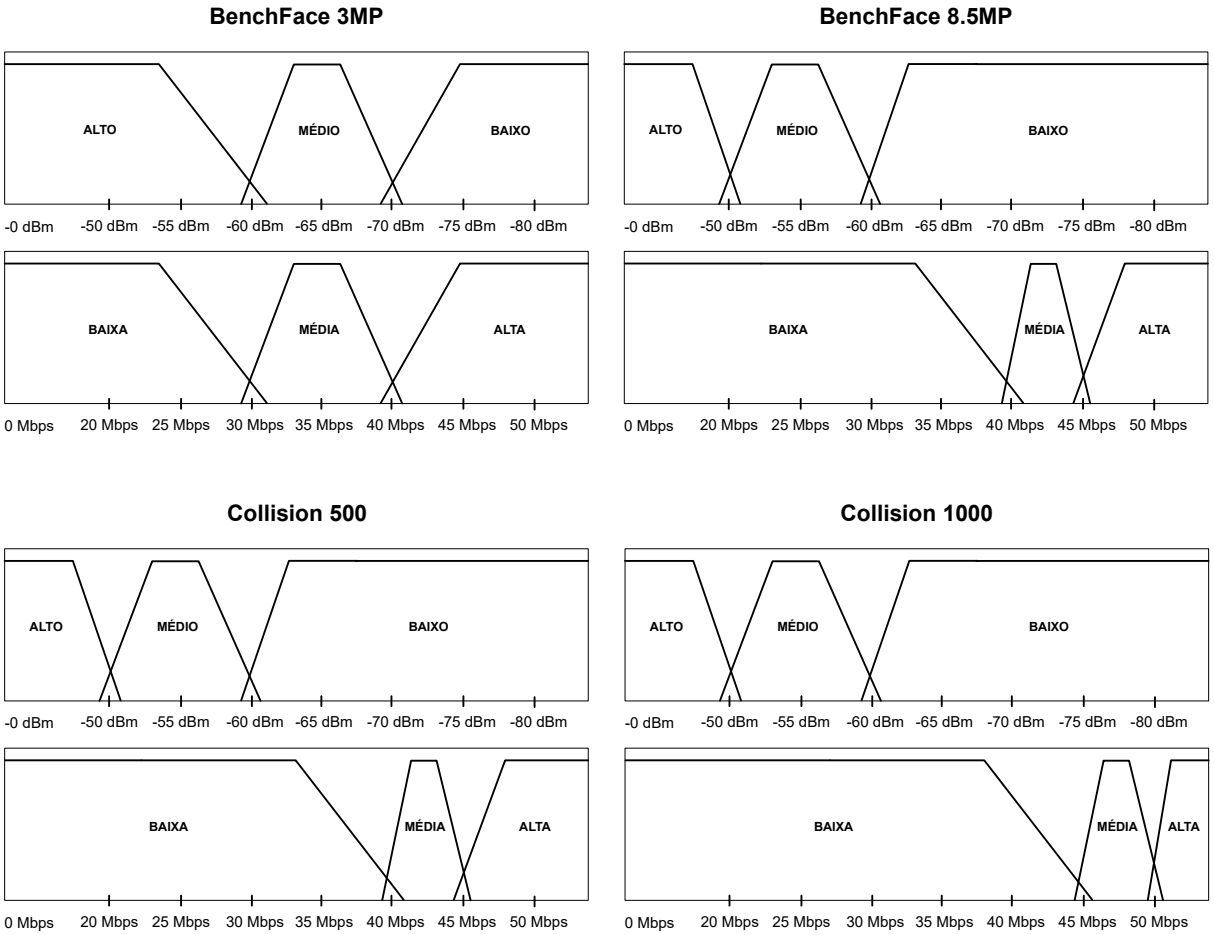
Para alcançar o objetivo principal do sistema proposto que é realizar as decisão de melhor rede e VM na *cloudlet* com base na aplicação em execução no dispositivo móvel, foram definidos os limiares para estas duas etapas através das funções de pertinência *fuzzy* e os critérios no cálculo de custo.

- **Decisão de rede**

Para a decisão de rede, cada grupo de funções de pertinência com os critérios de decisão faz parte de um perfil definido para cada tipo de aplicação e tamanho dos dados. Como o sistema *fuzzy* possui a característica de ser desenhado através do conhecimento de especialistas, foram realizados testes empíricos para definição dos graus de pertinência dos critérios utilizados. Para isso, foram executados *offloadings* com o BenchFace, nos tamanhos de imagens de 3 MP e 8.5 MP e com o Collision, para 500 e 1000 bolas. Os níveis dos critérios foram variados durante essas execuções. Para o RSSI, os níveis foram obtidos através do distanciamento do AP utilizado nos testes; para a vazão, foi utilizada a ferramenta iPerf[6] que permite injetar um determinado tráfego na rede, para variar o nível da largura de banda. Ao término destes testes, foram identificadas variações distintas entre cada tamanho de dados e tipo de aplicação, definindo assim, os graus de pertinência apresentados na Figura 19.

[6] iPerf - https://iperf.fr/

Figura 19 – Limiares das funções de pertinência das aplicações



Fonte: Elaborada pelo Autor

No BenchFace, para a métrica RSSI foram definidos para a resolução de 3 MP os valores maiores que -61 dBm como um nível de sinal "Alto", menor que -71 dBm como "Baixo" e os níveis entre estes dois valores como um sinal "Médio". Para a vazão, valores maiores que 39 Mbps é considerada uma taxa de transferência "Alta", menor que 31 Mbps como "Baixa" e entre estes valores como "Médio". Para a resolução de 8.5 MP, os valores adotados são um pouco mais restritivos, o RSSI considerado "Alto" deve ser maior que -51 dBm, "Baixo" a partir de -61 dBm e a faixa entre estes dois valores como "Médio". Na vazão, um valor maior que 46 Mbps foi definido como "Alto", menor que 41 Mbps como "Baixo" e entre estes valores como "Médio". Esta estratégia foi adotada para evitar ao máximo a redução no desempenho no *offloading* com tamanhos maiores de dados quando os valores dos parâmetros de QoS estão mais baixos.

Para a aplicação Collision que requer nível de sinal e vazão mais elevados, devido a suas características de constante interação com a rede sem fio, foram mantidos valores mais restritos para as métricas. Para 500 e 100 bolas foram mantidos os mesmos graus de pertinência utilizados no BenchFace 3 MP do RSSI e da vazão para 500 bolas. Para o tamanho de 1000 bolas, foi definida como vazão "Baixa" valores menores que 46 Mbps,

como "Alta", valores maiores que 51 Mbps, e entre estas duas faixas, como "Média".

Como último passo da configuração do sistema *fuzzy*, foram definidas as regras de inferência para os dois critérios utilizados (RSSI e vazão). Para saber a quantidade de regras necessárias, é preciso multiplicar a quantidade de variáveis linguísticas pelo número da potência dos termos linguísticos utilizados. No caso dessa proposta, obtemos portanto $3^2 = 9$, o Quadro 8 mostra as regras *fuzzy* criadas para o sistema de decisão, utilizadas nas duas aplicações de *benchmark* avaliadas.

Quadro 8 – Regras *fuzzy*

REGRA	**RSSI**	**Vazão**	***Handoff*?**
1	alto	alta	não
2	alto	média	não
3	alto	baixa	sim
4	médio	alta	não
5	médio	média	sim
6	médio	baixa	sim
7	baixo	alta	sim
8	baixo	média	sim
9	baixo	baixa	sim

- **Decisão de VM na *cloudlet***

Na decisão de VM na *cloudlet*, é preciso obter os valores do primeiro critério (tempo de processamento) para a fórmula do cálculo de custo. Para isso, foram realizadas algumas execuções na nuvem utilizando cada uma das aplicações, com o objetivo de realizar os experimentos com recursos de nuvem distintos. Foram levantados dois tipos de VMs na *cloudlet*, sendo o primeiro tipo com 1 vCPU e a segunda com 2 vCPUs. No BenchFace, foram calculados o tempo médio de execução de 30 *offloadings* nos tamanhos de imagens de 3 MP e 8.5 MP. Para o Collision, como não se obtém o tempo de execução, foi calculada a quantidade de FPS médios durante 10 segundos, repetindo 30 vezes cada uma para 500 e 1000 bolas.

Conforme o esperado devido ao menor poder computacional, os resultados do BenchFace na VM com maior número de vCPUs teve desempenho superior. As imagens de 3 MP tiveram o ganho percentual no tempo de execução de 39,11% e as de 8.5 MP de 40,55% na VM de 2 vCPUs quando comparadas com a de 1 vCPU. Seguindo a mesma tendência, a quantidade de FPS médios na aplicação Collision durante as execuções de 10s foram inferiores na VM com 1 vCPU em relação à de 2 vCPUs. O ganho médio percentual foi de 19,18% nos resultados com 500 bolas e de 23,96% para 1000 bolas na VM de 2 vCPUs em comparação com a de 1 vCPU. Um resumo dos resultados obtidos nos experimentos

que apresentam os valores aplicados na fórmula para cada aplicação e tamanho de dados são apresentados na Tabela 1.

Tabela 1 – Medições do tempo de processamento

Aplicação	**Tamanho**	**Tipo da VM**	**Resultado**
BenchFace	3 MP	1 vCPU	5,25 s
		2 vCPU	3,13 s
	8.5 MP	1 vCPU	10,53 s
		2 vCPU	6,26 s
Collision	500 bolas	1 vCPU	25,07 FPS
		2 vCPU	31,02 FPS
	1000 bolas	1 vCPU	20,03 FPS
		2 vCPU	26,34 FPS

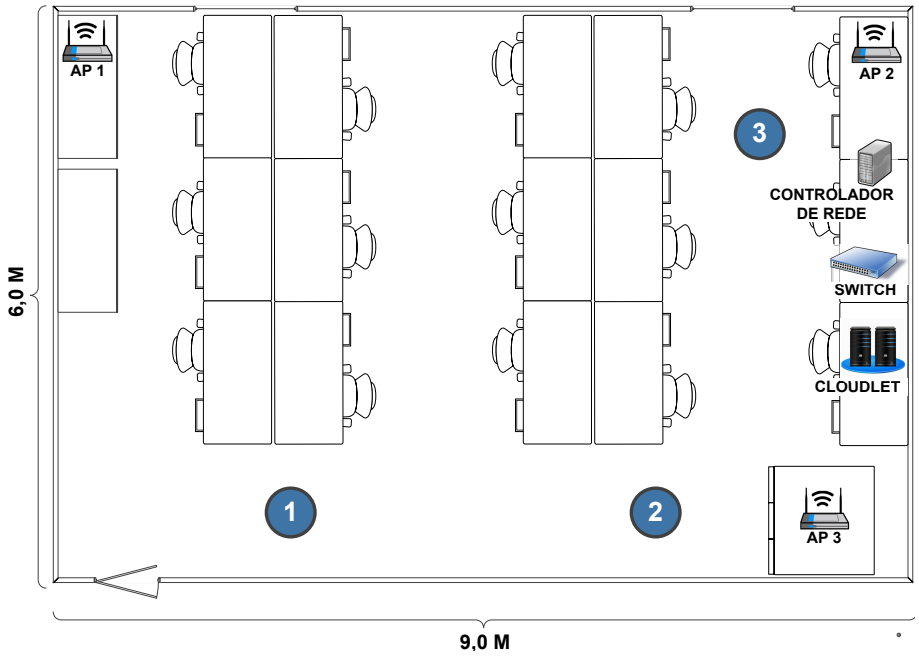
5.3 Experimentos

Esta Seção apresenta dois experimentos e analisa os resultados obtidos nas execuções. Em cada um deles, foram utilizados cenários variados com o objetivo de validar a proposta desta dissertação. O primeiro experimento, simula um conjunto de contextos de rede e nuvem, onde o sistema de decisão precisa selecionar o melhor destes recursos para execução de *offloading* utilizando o dispositivo móvel. O segundo experimento, utiliza a seleção de rede durante a mobilidade do usuário, para que os APs alvo das execuções de *handoff* contenham níveis de QoS que forneçam a melhor experiência ao usuário para sua aplicação. Mais detalhes são apresentados nas subseções a seguir.

5.3.1 Experimento 1 - Seleção de rede e VM na *cloudlet*

O objetivo deste experimento é inserir o dispositivo móvel em cenários de contextos de rede (com diferentes níveis de RSSI e vazão) e também das VMs na *cloudlet* (variando o nível de processamento, uso de memória e valores de RTT). Com isso, é possível verificar se o sistema de decisão irá selecionar os melhores recursos nestes cenários tomando como base a aplicação em execução. Os cenários foram definidos através de três locais no ambiente de testes, conforme apresentados na Figura 20. Inicialmente, são avaliadas para cada aplicação a seleção de rede através do sistema *fuzzy* e, em seguida, a seleção de VMs na *cloudlet* utilizando o cálculo de custo de *offloading*.

Figura 20 – Ambiente de testes para o Experimento 1



Fonte: Elaborada pelo Autor

5.3.1.1 Seleção de rede

Na seleção de rede, a nomenclatura utilizada para classificar os níveis de RSSI e vazão dos três cenários, estão contidas na Tabela 2. Estes níveis são baseados nos graus de pertinência *fuzzy* apresentados na Figura 19. Para cada um dos cenários, foram variados os tipos das aplicações e o tamanho dos dados.

Tabela 2 – Cenários de seleção de rede

Cenário	**AP 1**		**AP 2**		**AP 3**	
	RSSI (dBm)	**Vazão (Mbps)**	**RSSI (dBm)**	**Vazão (Mbps)**	**RSSI (dBm)**	**Vazão (Mbps)**
1	Baixo	Média	Alto	Média	Alto	Baixa
2	Baixo	Baixa	Alto	Baixa	Alto	Alta
3	Baixo	Alta	Alto	Média	Alto	Baixa

- **BenchFace**

A primeira aplicação avaliada para a seleção de rede sem fio é o BenchFace. A Tabela 3 apresenta para cada um dos cenários, o nível médio de RSSI capturado pelo dispositivo móvel e o tráfego médio em Mbps que foi gerado em cada um dos APs para atingir a classificação estabelecida na Tabela 2. Por exemplo, no AP 1 do primeiro cenário de 3 MP, o valor do RSSI que apresenta valor de -75,37 dBm é classificado como "Baixo" e para obter uma vazão "Média" foi necessário gerar um tráfego de 9,12 Mbps. Em cada

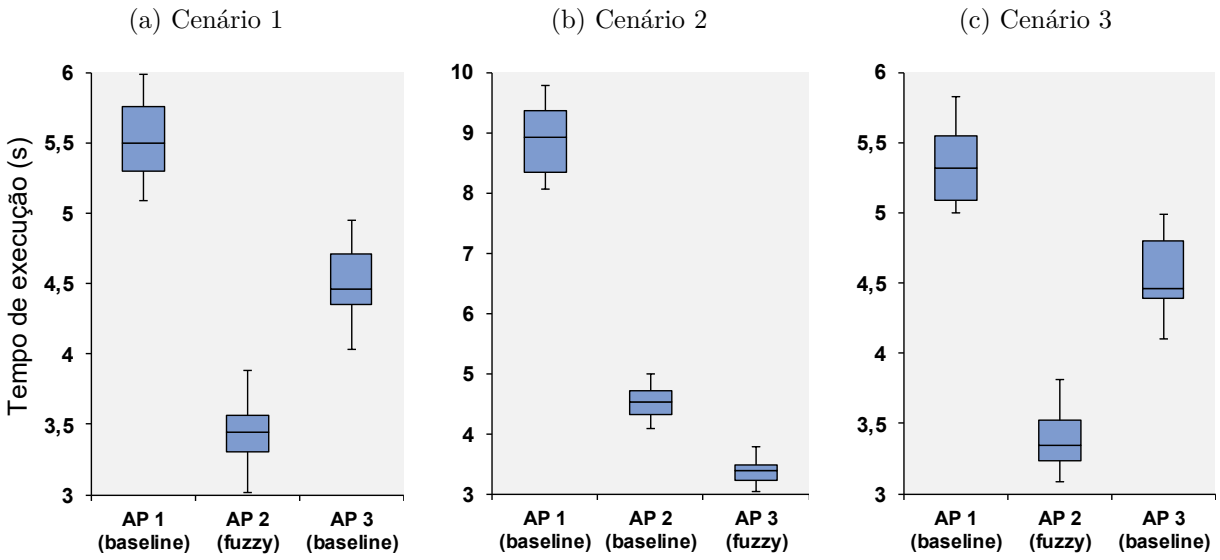
cenário, foram realizadas 30 execuções de *offloading* nos três APs para comparação dos resultados da rede selecionada nos tamanhos de imagem de 3 MP e 8.5 MP.

Tabela 3 – Cenários de seleção de rede para o BenchFace

Tamanho	**Cenário**	**AP 1**		**AP 2**		**AP 3**	
		RSSI (dBm)	**Tráfego (Mbps)**	**RSSI (dBm)**	**Tráfego (Mbps)**	**RSSI (dBm)**	**Tráfego (Mbps)**
3 MP	1	- 75,37	9,12	- 44,03	17,56	- 34,20	26,40
	2	- 82,27	41,99	- 42,37	30,19	- 33,33	9,91
	3	- 73,77	9,76	- 32,19	13,52	- 47,63	30,73
8.5 MP	1	- 62,53	8,89	- 49,12	9,14	- 42,31	27,32
	2	- 87,36	39,27	- 31,85	15,09	- 44,25	5,03
	3	- 73,83	8,47	- 32,19	8,72	- 46,12	29,32

A Figura 21 apresenta os resultados de *boxplot* correspondentes a todos os experimentos realizados com o tamanho de imagem de 3 MP. Os rótulos "*fuzzy*" representam o AP selecionado pelo sistema de decisão e "*baseline*" os resultados das execuções de *offloading* nos outros APs para comparação. No primeiro cenário (Figura 21a), foi escolhido o AP 2 pelo sistema de decisão por ser o único que não contém valores classificados como "Baixo", seguindo as regras *fuzzy* definidas no Quadro 8. As execuções de *offloading* nesta rede tiveram um ganho percentual no tempo médio de 23,82% em relação ao AP 3, que obteve o segundo melhor desempenho. Esta colocação foi obtida pelo AP 3 pois, apesar de ter um RSSI "Alto" como a rede selecionada, o tempo de execução foi influenciado devido a uma vazão "Baixa". Outro aspecto a ser observado nesse cenário, é que as execuções no AP 1 obtiveram o maior tempo médio devido principalmente ao "Baixo" nível de sinal.

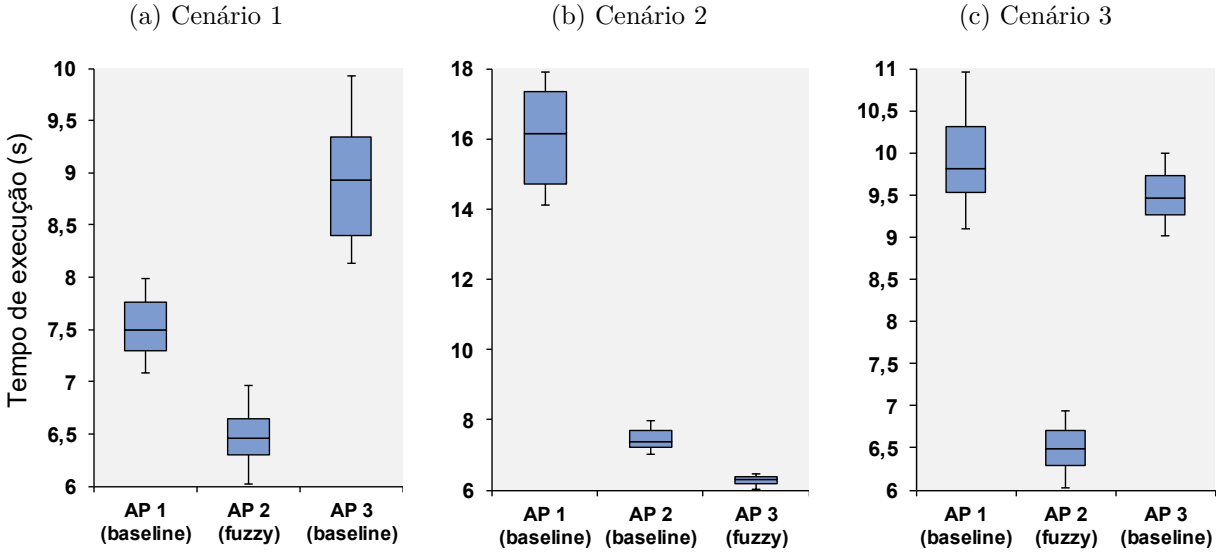
Figura 21 – Resultados BenchFace 3 MP para seleção de rede



No segundo cenário (Figura 21b), o sistema *fuzzy* dispara o *handoff* para o AP 3, que de acordo com os resultados, obteve um ganho médio de 24,74% no tempo de *offloading* comparado com o AP 2 que, dessa forma, obteve o segundo melhor desempenho. A rede escolhida foi beneficiada principalmente pelo valores do RSSI e vazão classificados como "Alto", proporcionando o melhor ambiente possível para as execuções de *offloading*, conforme definido nas funções de pertinência *fuzzy*. É possível também verificar na Figura que o primeiro AP apresentou o maior tempo de execução em todos os cenários. Esta colocação foi obtida devido a ambas as métricas da rede (RSSI e vazão) apresentarem valores definidos como "Baixo", que neste caso, torna esta rede considerada como inelegível para o *offloading*. O ganho percentual da rede escolhida comparado com o AP 3 foi de 61,71%.

Em relação ao terceiro cenário (Figura 21c), o nível de RSSI dos APs 2 e 3 se mantiveram classificados como "Alto", a exemplo do cenário anterior. O fator determinante para a seleção entre estas duas redes foi o nível da vazão, que tornou a terceira rede inelegível por receber a classificação "Baixa". O AP 1 também não foi selecionado por apresentar o RSSI como "Baixo", onde os resultados foram diretamente influenciados por esta métrica, mesmo possuindo a maior vazão das três redes, apresentando o maior tempo médio do cenário. Dessa forma, o *handoff* foi disparado para a segunda rede, que obteve um ganho de 25,74% comparado ao AP 3 e de 36,52% ao AP 1.

Figura 22 – Resultados BenchFace 8.5 MP para seleção de rede



Para seleção de rede no tamanho de imagem de 8.5 MP, os resultados são apresentados na Figura 22 por *boxplot*. Devido à repetição dos cenários, a ordem de decisão das redes em cada um deles permaneceram as mesmas dos resultados para 3 MP. No primeiro cenário (Figura 22a), o sistema *fuzzy* escolheu o segundo AP. Contudo, é importante observar que dessa vez o AP 1 foi inelegível com o RSSI médio de -62,53 dBm, situação esta

que não aconteceria no cenário de 3 MP. Para este tamanho menor de dados, os graus de pertinência definidos, classificam este nível de sinal como "Médio". Dessa forma, a primeira rede também seria candidata ao *handoff*, pois a regra *fuzzy* de número 5 seria acionada. No entanto, como se trata de um cenário de 8.5 MP, o ganho percentual do AP 2 para a primeira e terceira rede foi de 13,72% e 27,25%, respectivamente.

No cenário 2 (Figura 22b), o AP 1 apresentou o maior tempo médio e variação nos valores das medições, fato este que ocorreu principalmente pelo RSSI apresentar níveis muito baixos. O sistema de decisão escolheu o AP 3, que ocasionou um ganho médio de 15,91% comparado ao AP 2. O terceiro cenário (Figura 22c) apresenta o maior ganho percentual de todos os cenários avaliados, que foi de 31,58% e 34,73% em relação à primeira e à segunda redes, respectivamente. Nesse cenário, a similaridade entre os resultados dos APs 1 e 3 pode ser explicada devido a este maior tamanho de imagem, ter sofrido influências equivalentes nos valores das métricas obtidas nestas duas redes.

Para maiores detalhes acerca dos valores obtidos nesta primeira avaliação, a Tabela 4 mostra os valores médios com 95% de intervalo de confiança em cada cenário, contendo as variações de tamanho de imagem em cada um dos APs avaliados.

Tabela 4 – Média e intervalo de confiança do BenchFace nos cenários avaliados

Tamanho	**Cenário**	**AP 1**	**AP 2**	**AP 3**
		Tempo médio (s)	**Tempo médio (s)**	**Tempo médio (s)**
3 MP	1	5,51 ± 0,982	3,42 ± 0,077	4,49 ± 0,093
	2	8,86 ± 0,203	4,51 ± 0,090	3,39 ± 0,077
	3	5,33 ± 0,095	3,38 ± 0,072	4,45 ± 0,101
8.5 MP	1	7,51 ± 0,098	6,48 ± 0,097	8,91 ± 0,203
	2	16,04 ± 0,488	7,46 ± 0,106	6,27 ± 0,043
	3	9,94 ± 0,202	6,49 ± 0,094	9,48 ± 0,097

Os valores obtidos através da seleção de rede com a aplicação BenchFace, demonstram que a calibração do sistema *fuzzy* permitiu obter ganhos consideráveis nos percentuais de tempo nos dois tamanhos de imagens (3 MP e 8.5 MP). Foi possível alcançar resultados de execuções de *offloading* mais eficazes quando as regras foram acionadas principalmente pelo estado da vazão dos APs também em imagens menores, como é verificado nos cenários da Figura 22b e 22c. Além disso, foi demonstrado que a aplicação de graus de pertinência distintos para cada tamanho de dados mostrou-se eficaz na decisão de *handoff*, conforme explicado no cenário da Figura 22a. Por fim, também foi possível constatar como um baixo nível de RSSI causou grande impacto no desempenho do BenchFace, reforçando mais uma vez como a praticidade do uso das regras *fuzzy* proporcionaram decisões acertadas, que pode ser verificado no desempenho do AP 1 na Figura 22b.

- **Collision**

A segunda aplicação avaliada para a seleção de rede é o Collision. Foi seguida a mesma metodologia do BenchFace, utilizando três cenários com valores diferentes de RSSI e o tráfego médio em Mbps que foi gerado nas três redes sem fio, conforme apresentado na Tabela 5. Como esta aplicação não possui um tempo de execução padrão devido à sua característica de constante interação com a nuvem, foram realizadas para cada cenário 30 execuções de 10 segundos cada, para o tamanho de 500 e 1000 bolas.

Tabela 5 – Cenários de seleção de rede para o Collision

Tamanho	**Cenário**	**AP 1**		**AP 2**		**AP 3**	
		RSSI (dBm)	**Tráfego (Mbps)**	**RSSI (dBm)**	**Tráfego (Mbps)**	**RSSI (dBm)**	**Tráfego (Mbps)**
500 bolas	1	- 64,77	9,07	- 50,03	9,03	- 42,00	29,40
	2	- 81,43	41,99	- 32,37	15,19	- 47,33	5,03
	3	- 73,84	4,06	- 32,19	8,98	- 43,63	30,73
1000 bolas	1	- 65,81	8,15	- 49,15	4,34	- 43,73	31,72
	2	- 83,24	43,12	- 34,01	16,48	- 48,21	3,28
	3	- 75,32	3,15	- 32,16	4,18	- 44,23	31,66

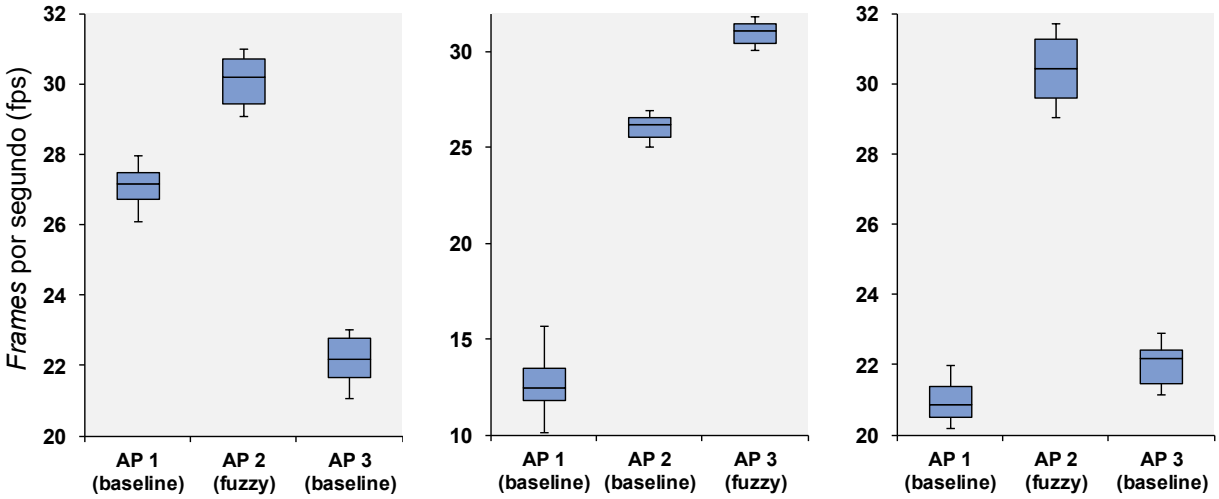
Os resultados das avaliações para 500 bolas são apresentados na Figura 23. No primeiro cenário (Figura 23a) o AP 2 foi escolhido pelo sistema *fuzzy*, com base nas regras definidas no Quadro 8. Apesar dos APs 1 e 2 apresentarem certa semelhança na vazão média e uma pequena diferença no nível de RSSI, observa-se um ganho percentual no FPS de 9,91% no AP selecionado em comparação com a rede que obteve o segundo melhor desempenho, que foi o AP 1. Já é possível obter estes ganhos através destas diferenças devido a este tipo de aplicação interagir a maior parte do tempo com a rede sem fio. Por esse motivo, é mais sensível a estas variações das métricas, dessa forma, a primeira rede é descartada por apresentar a classificação do nível de sinal como "Baixo". Seguindo a mesma tendência, o AP 3 obteve o menor valor de FPS, pois percebe-se um tráfego médio de 29,40 Mbps, sendo o mais elevado do que as outras redes sem fio no cenário.

No segundo cenário (Figura 23b), novamente a primeira rede é descartada devido à baixa disponibilidade de recursos da rede sem fio, através dos valores apresentados no nível RSSI e vazão. Esta situação fez com que fosse obtida uma quantidade de FPS médio de apenas 12,58, sendo o menor resultado de todos os três cenários para esta quantidade de bolas. Para a execução de *handoff*, foi selecionado pelo sistema *fuzzy* o AP 3 que apesar de apresentar um valor de RSSI um pouco inferior que o AP 2, conseguiu prover uma melhor experiência de *offloading* por conta da vazão ser classificada como "Alta". Esta rede selecionada, obteve um ganho percentual de 15,87% em relação ao AP 2 na quantidade de FPS médio para o tamanho de 500 bolas.

No último cenário (Figura 23c) foi selecionado pelo sistema *fuzzy* o segundo AP, impulsionado principalmente pelo valor da vazão "Média". Observa-se uma maior variação

Figura 23 – Resultados Collision 500 bolas para seleção de rede

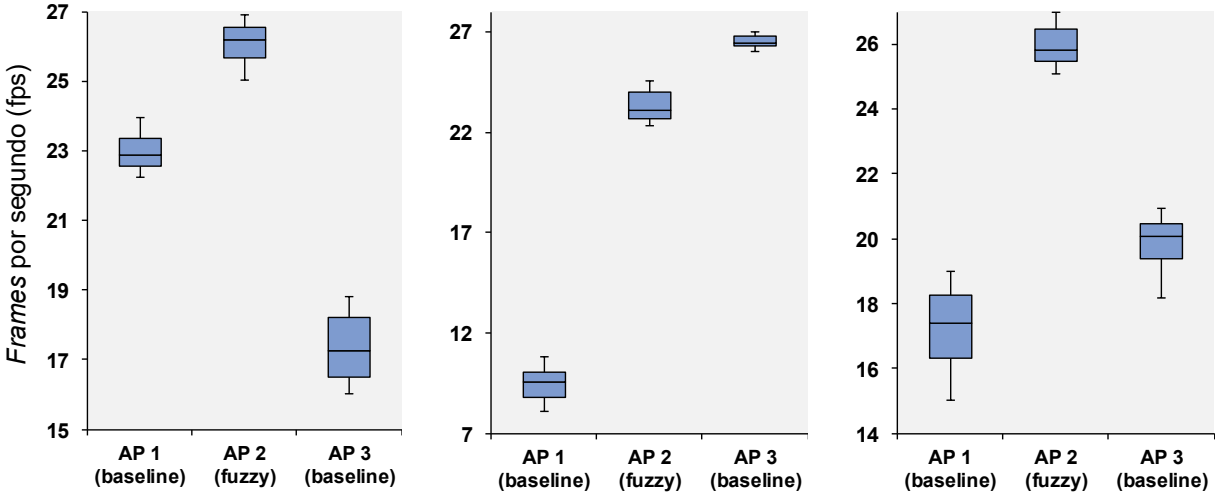
(a) Cenário 1 (b) Cenário 2 (c) Cenário 3



nos valores de FPS neste cenário, em comparação com o AP 3 no cenário anterior. Este evento se justifica pelo fato da pequena diferença no tráfego gerado em Mbps nestes dois ambientes, demonstrando mais uma vez o grande impacto que estas métricas causam neste tipo de aplicação. No cenário atual, a rede selecionada apresentou um ganho percentual de 31,07% e 27,52% comparando com a primeira e terceira rede, respectivamente.

Figura 24 – Resultados Collision 1000 bolas para seleção de rede

(a) Cenário 1 (b) Cenário 2 (c) Cenário 3



Para as execuções do Collision com 1000 bolas, a exemplo do tamanho de 500 bolas a ordem das redes selecionadas foram as mesmas. A Figura 24 apresenta os valores dos três cenários avaliados. No cenário 1 (Figura 24a), foi escolhida a segunda rede que obteve um ganho percentual de 12,07% comparado com o AP 1. Observa-se uma maior variação de

FPS no terceiro AP devido a esta quantidade de bolas sofrer mais degradação de desempenho com baixos recursos de QoS. Já no segundo cenário (Figura 24b), é selecionada a terceira rede sem fio, que apesar da proximidade entre os valores de FPS com o AP 2, as execuções na rede escolhida apresentaram um ganho de 12,13% para este AP. Por fim, no cenário 3 (Figura 24c) foi selecionado o AP 2 que obteve um ganho percentual em relação ao segundo AP com melhor FPS (terceira rede) de 23,41%.

Para fins de verificação, a Tabela 6 apresenta a seguir em detalhes o intervalo de confiança de 95% para quantidade de FPS médios renderizados pelo Collision, utilizando as quantidades de 500 e 1000 bolas nos cenários avaliados.

Tabela 6 – Média e intervalo de confiança do Collision nos cenários avaliados

Tamanho	**Cenário**	**Rede 1**	**Rede 2**	**Rede 3**
		FPS médio	**FPS médio**	**FPS médio**
500 bolas	1	27,09 ± 0,201	30,08 ± 0,234	22,19 ± 0,220
	2	12,58 ± 0,468	26,08 ± 0,203	31,00 ± 0,187
	3	20,94 ± 0,203	30,39 ± 0,319	22,02 ± 0,203
1000 bolas	1	22,95 ± 0,177	26,10 ± 0,208	17,37 ± 0,331
	2	9,47 ± 0,287	23,29 ± 0,259	26,51 ± 0,106
	3	17,18 ± 0,457	25,94 ± 0,200	19,86 ± 0,255

Para cada um dos cenários avaliados para esta aplicação, também foi calculada a porcentagem de perda de pacotes devido ao Collision possibilitar visualização dos ganhos com o sistema de decisão através desta métrica. A Tabela 7 apresenta estes valores obtidos com intervalo de confiança de 95%. Destacam-se os maiores ganhos obtidos nas redes selecionadas pelo sistema *fuzzy*, que foram de 96,09% e 94,77% no segundo cenário para 500 e 1000 bolas, comparando com os APs 1 e 3, respectivamente.

Tabela 7 – Perda de pacotes e intervalo de confiança do Collision nos cenários avaliados

Tamanho	**Cenário**	**Rede 1**	**Rede 2**	**Rede 3**
		Perda de Pacotes (%)	**Perda de Pacotes (%)**	**Perda de Pacotes (%)**
500 bolas	1	1,62 ± 0,108	0,93 ± 0,221	8,62 ± 0,272
	2	22,54 ± 0,631	1,83 ± 0,219	0,88 ± 0,222
	3	12,22 ± 0,527	0,51 ± 0,121	9,71 ± 0,321
1000 bolas	1	4,10 ± 0,204	1,26 ± 0,314	11,88 ± 0,436
	2	31,48 ± 0,329	3,13 ± 0,290	1,65 ± 0,321
	3	18,12 ± 0,428	0,96 ± 0,219	12,22 ± 0,527

A exemplo da aplicação BenchFace, os resultados obtidos através do Collision demonstram os benefícios da utilização do sistema proposto. Verifica-se principalmente que pequenas variações nos valores de RSSI, impactam de modo considerável o desempenho desta aplicação, conforme destacado para 500 bolas no cenário da Figura 23a, comparando o AP 1 com o 2. Para 1000 bolas também no primeiro cenário (Figura 24a) com o AP 3,

é possível observar variações consideráveis na quantidade de FPS durante o *offloading*. Devido principalmente a estes aspectos, percebe-se a necessidade de utilizar limiares mais reduzidos no sistema *fuzzy* para esta aplicação, constatando através dos experimentos que os graus de pertinência adotados apresentaram resultados satisfatórios.

5.3.1.2 Seleção de VM na cloudlet

A segunda decisão avaliada é a seleção de VM na *cloudlet*. A exemplo das avaliações anteriores, foram definidos três cenários, conforme apresentado na Tabela 8. A nomenclatura utilizada para classificar os níveis das métricas nesta Tabela foi definida da seguinte forma: nível de processamento da CPU menor que 45% como "Baixo", entre 45% e 50% como "Médio" e entre 70% e 80% como "Alto"; para a memória, foi considerado um nível "Baixo" nos valores menores que 50% e como "Alto" a partir de 80%; para o RTT, até 110 ms foi definido como "Baixo" e maior que 300 ms como "Alto". É importante destacar nesta etapa de seleção que foram utilizadas dois tipos de VMs, as duas primeiras contendo 2 vCPUs e a terceira com apenas 1 vCPU, ambas com 2 GB de memória. Para cada um dos cenários, foram variadas as aplicações e o tamanho de dados.

Tabela 8 – Cenários de seleção de VM na *cloudlet*

Cenário	**VM 1 (2 vCPUs)**		**VM 2 (2 vCPUs)**		**VM 3 (1 vCPU)**	
	CPU (%)	**Mem (%) RTT (ms)**	**CPU (%)**	**Mem (%) RTT (ms)**	**CPU (%)**	**Mem (%) RTT (ms)**
1	Baixo	Baixo	Alto	Baixo	Médio	Alto
2	Médio	Alto	Alto	Alto	Baixo	Baixo
3	Médio	Baixo	Baixo	Alto	Baixo	Baixo

- **BenchFace**

Para a seleção de VM na *cloudlet* com a aplicação BenchFace, a fórmula de custo recebe os critérios de tempo (TP) e nível (CPU) de processamento e uso de memória (MEM), conforme apresentado na Equação 5.1. Os três cenários utilizados nas avaliações nos tamanhos de imagem de 3 MP e 8.5 MP são apresentados na Tabela 9, onde foram realizadas 30 execuções de *offloading* em cada VM utilizada.

$$Ct = (pTP \times nTP) + (pCPU \times nCPU) + (pMEM \times nMEM) \tag{5.1}$$

Os resultados para imagens de 3 MP estão contidos na Figura 25, que apresenta os resultados de *boxplot*. Os rótulos "*custo*" representam a VM selecionada através do cálculo de menor custo de *offloading* e "*baseline*" os resultados das execuções nas outras VMs para fins de comparação. É importante destacar que em todos os cenários que o nível de memória das VMs recebem classificação "Baixo", foram medidos valores médios a partir de

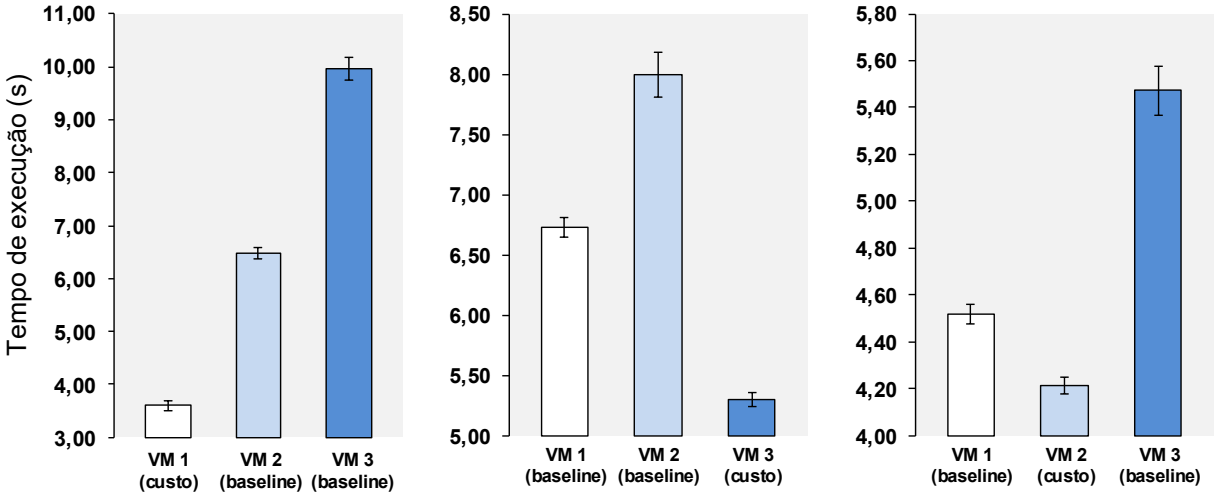
Tabela 9 – Cenários de seleção de VM na *cloudlet* para o BenchFace

Tamanho	Cenário	VM 1		VM 2		VM 3	
		CPU (%)	Mem (%)	CPU (%)	Mem (%)	CPU (%)	Mem (%)
3 MP	1	4,82	41,96	72,93	42,23	50,63	78,66
	2	48,27	79,37	78,02	80,71	3,07	41,15
	3	47,14	49,73	3,72	81,64	3,73	43,14
8.5 MP	1	1,87	41,67	73,77	41,98	51,53	81,37
	2	47,40	83,14	77,31	81,27	2,07	41,62
	3	48,93	43,57	2,05	81,23	2,23	42,21

40%. Esta situação aconteceu devido ao consumo de memória que o serviço de *offloading* ocupa através da *Java Virtual Machine* (JVM), já que estas máquinas virtuais possuem apenas 2 GB de RAM. No primeiro cenário (Figura 25a) a VM 1 obteve o menor custo de *offloading* principalmente por apresentar um nível de processamento menor da CPU em comparação com a segunda VM. Conforme já esperado, o último recurso de nuvem obteve o maior tempo médio por conta da menor disponibilidade computacional. O ganho percentual obtido pela VM selecionada em relação à segunda e terceira VMs foi de 44,36% e 63,84%, respectivamente.

Figura 25 – Resultados BenchFace 3 MP para seleção de VM na *cloudlet*

(a) Cenário 1 (b) Cenário 2 (c) Cenário 3

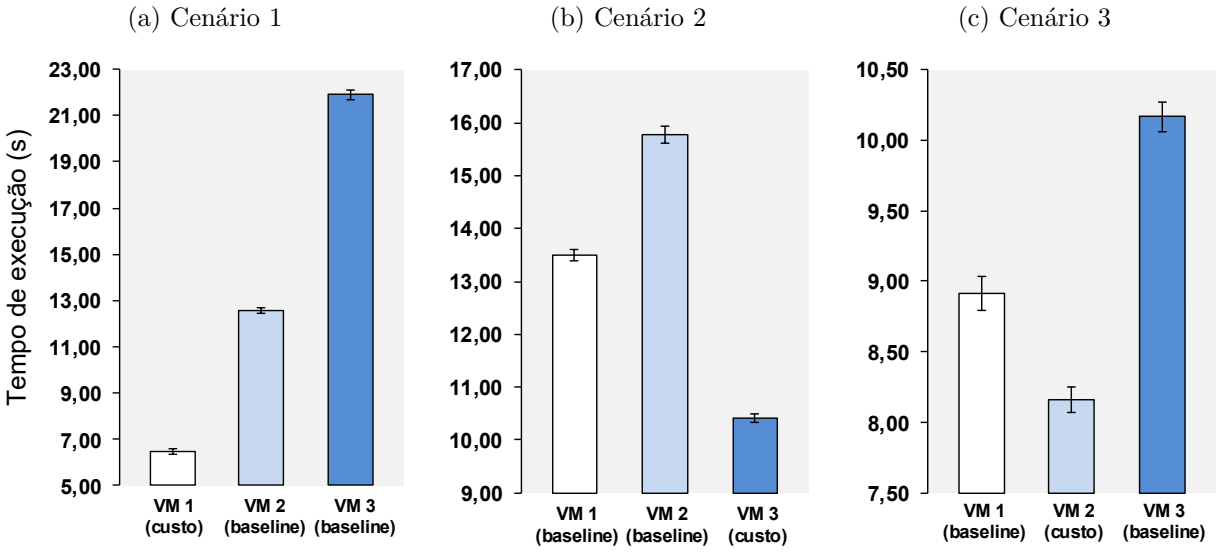


No segundo cenário (Figura 25b), as duas primeiras VMs que possuem 2 vCPUs apresentam um nível de processamento classificado como "Médio" e "Alto", além do "Alto" consumo de memória. Observa-se nesta situação que a VM 3 obteve o menor custo de *offloading*, mesmo possuindo apenas 1 vCPU, que comparada com as outras VMs, espera-se um tempo de execução mais lento. No entanto, no cenário atual foi possível identificar uma situação em que foi mais vantajoso executar o *offloading* numa VM com este tipo de

configuração. Através dos resultados obtidos, a terceira máquina virtual apresentou um ganho de 21,22% e 33,75% em comparação com as VMs 1 e 2, respectivamente.

O terceiro cenário (Figura 25c), apresenta uma situação em que a última VM possui o máximo de seus recursos disponíveis. No entanto, devido a suas características já destacadas no cenário anterior, obteve o maior tempo médio das execuções e também o maior custo em relação às outras VMs. A escolha entre as duas primeiras máquinas virtuais, foi definida principalmente pelo maior peso atribuído na fórmula para o nível da CPU (detalhes no Quadro 5). Neste caso, mesmo a memória apresentando o valor classificado como "Baixo" na primeira VM, o nível de processamento foi mais determinante para o custo de *offloading*. Dessa forma, foi selecionada a VM 2 onde os resultados mostram um ganho percentual de 6,76% em relação à VM 1 que obteve o segundo menor custo.

Figura 26 – Resultados BenchFace 8.5 MP para seleção de VM na *cloudlet*



Para imagens de 8.5 MP, a ordem de seleção das VMs foi a mesma dos cenários das imagens de 3 MP. Os resultados são apresentados na Figura 26 onde no primeiro cenário (Figura 26a), obteve o menor valor de custo a primeira máquina virtual, que teve um ganho no tempo de processamento de 48,33% e 70,38% comparada às VMs 2 e 3, respectivamente. Para o segundo cenário (Figura 26b), a exemplo da imagem de 3 MP, também obteve o menor custo a terceira VM que possui apenas 1 vCPU. O ganho percentual em relação às duas primeiras VMs foi de 22,88% e 34,02%, respectivamente. No último cenário (Figura 26c) para este tamanho de imagem, apesar de uma pequena diferença nos valores de custo entre a VM 1 e 2, foi determinante para uma decisão correta, conforme o ganho percentual de 8,43% em comparação com a VM 1.

A seguir, a Tabela 10 apresenta com mais detalhes os valores médios de *offloading* obtidos com intervalo de confiança de 95% em cada uma das VMs nos cenários avaliados,

bem como os respectivos valores de custo calculados. Destaca-se nesta etapa que o sistema de decisão sempre escolhe a VM com o menor valor obtido.

Tabela 10 – Média e intervalo de confiança do BenchFace nos cenários avaliados

Tamanho	**Cenário**	**VM 1**		**VM 2**		**VM 3**	
		Tempo médio (s)	**Custo**	**Tempo médio (s)**	**Custo**	**Tempo médio (s)**	**Custo**
3 MP	1	3,60 ± 0,095	0,314	6,47 ± 0,101	0,527	9,96 ± 0,206	0,776
	2	6,72 ± 0,080	0,572	7,99 ± 0,186	0,669	5,29 ± 0,054	0,506
	3	4,51 ± 0,044	0,472	4,21 ± 0,037	0,440	5,47 ± 0,102	0,514
8.5 MP	1	6,48 ± 0,110	0,304	12,55 ± 0,112	0,529	21,90 ± 0,213	0,788
	2	13,50 ± 0,112	0,581	15,78 ± 0,158	0,668	10,41 ± 0,083	0,504
	3	8,91 ± 0,119	0,454	8,15 ± 0,091	0,434	10,16 ± 0,107	0,507

Com base nos resultados das avaliações em ambos os tamanhos de imagens para a aplicação BenchFace, ficam evidentes os benefícios das decisões realizadas pelo sistema proposto através do cálculo de custo. Através desta estratégia, destaca-se principalmente a capacidade de seleção de VMs distintas, através do uso do critério de tempo de processamento (TP). Dessa forma, foi possível verificar situações em que um tipo de máquina virtual com uma menor quantidade de vCPUs apresentaria melhores resultados de *offloading*, conforme verificado em alguns cenários (Figura 25b e 26b). Além disso, foi possível obter decisões eficazes através dos pesos atribuídos a cada um dos critérios, onde verifica-se no terceiro cenário para as VMs 1 e 2 (Figura 25c e26c) que a CPU, de fato, foi mais determinante que a memória para estas duas máquinas virtuais, permitindo obter execuções de *offloading* com menor tempo médio.

- **Collision**

Na avaliação da seleção de VM na *cloudlet* utilizando a aplicação Collision, são apresentados três cenários que a exemplo da avaliação com o BenchFace, também possuem três de tipos de VMs distintas na *cloudlet*. Vale destacar que para esta aplicação, o terceiro critério utilizado na fórmula de custo é o RTT, conforme observado na Equação 5.2. Os cenários em detalhes com os níveis da CPU e RTT são apresentados na Tabela 11.

$$Ct = (pTP \times nTP) + (pCPU \times nCPU) + (pRTT \times nRTT) \tag{5.2}$$

Os resultados para os testes desta aplicação com 500 bolas podem ser observados na Figura 27. Um aspecto que é importante destacar nos cenários avaliados é que o nível do RTT classificados como "Baixo", apresentam valores médios de 100 ms. Estes valores registrados foram medidos através do dispositivo móvel que, devido ao número de redes sem fio existentes no ambiente avaliado, além das utilizadas nos testes, não permitiu obter valores menores nesta métrica. Em relação ao primeiro cenário (Figura 27a), o menor custo

Tabela 11 – Cenários de seleção de VM na *cloudlet* para o Collision

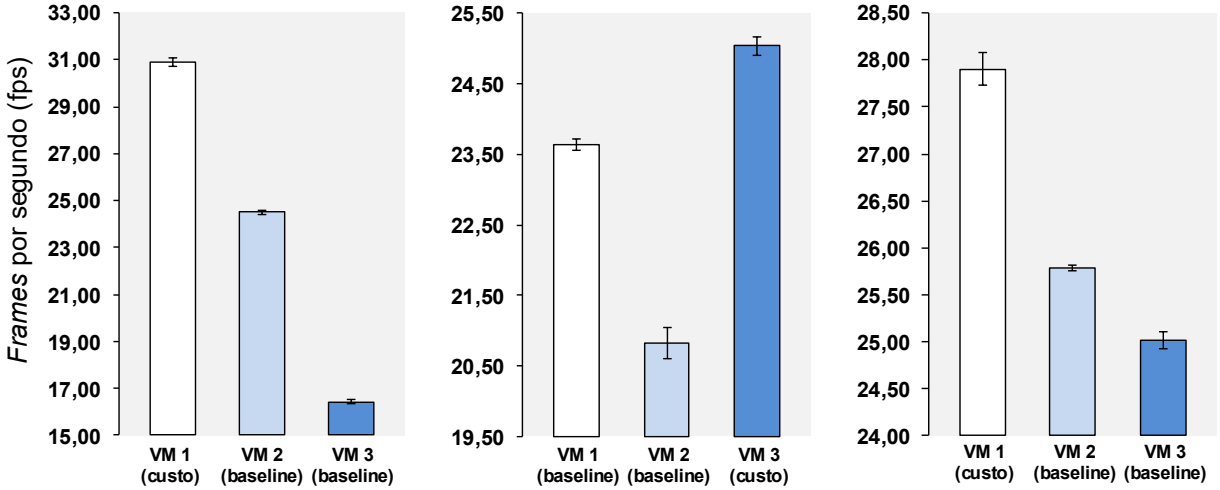
Tamanho	Cenário	Rede 1		Rede 2		Rede 3	
		CPU (%)	RTT (ms)	CPU (%)	RTT (ms)	CPU (%)	RTT (ms)
500 bolas	1	4,04	101,84	72,31	101,84	44,58	310,16
	2	42,12	321,76	77,12	306,68	2,72	98,24
	3	46,60	100,95	2,06	310,38	2,24	102,99
1000 bolas	1	3,03	102,54	76,49	100,43	47,47	307,54
	2	43,98	327,62	74,23	307,17	2,57	98,66
	3	46,95	99,28	3,04	304,81	2,25	99,19

foi obtido pela VM 1 por apresentar o nível de CPU menor que a segunda rede. A terceira rede sofreu mais degradação de desempenho, pois além de ter apenas 1 vCPU, o nível de RTT está classificado como "Alto". O ganho percentual médio de FPS na máquina virtual selecionada foi de 20,68% e 46,80% em comparação com a VM 2 e 3, respectivamente.

No segundo cenário (Figura 27b), a VM com apenas 1 vCPU obteve o menor custo. A exemplo do BenchFace, também com o Collision foi possível através da fórmula de custo, selecionar uma máquina virtual com menor recurso de processamento que pudesse fornecer ao usuário uma melhor experiência de *offloading* em relação às demais. O ganho médio de FPS na rede escolhida foi de 5,58% em relação à VM 1, que neste cenário, apresentou o segundo melhor resultado, tendo seu desempenho reduzido por conta do "Alto" valor de RTT medido, que aumentou o tempo de resposta das solicitações do dispositivo móvel.

Figura 27 – Resultados Collision 500 bolas para seleção de VM na *cloudlet*

(a) Cenário 1 (b) Cenário 2 (c) Cenário 3

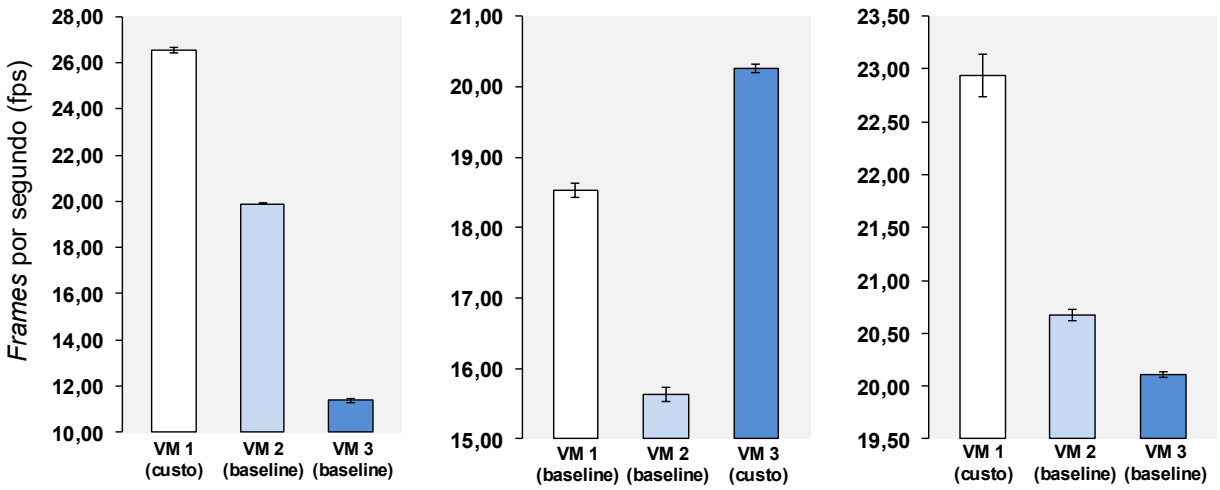


Para o terceiro cenário (Figura 27c), foi obtido o menor custo de *offloading* na primeira VM. Neste caso, percebe-se que o nível RTT gerou bastante impacto nas execuções do Collision na segunda VM. Também devido aos efeitos desta métrica na aplicação, as VMs

2 e 3 apresentaram valores equivalentes de FPS. Ao verificar o ganho percentual do recurso de nuvem selecionado, obtemos o valor de 7,58% e 10,35% em comparação com as VMs 2 e 3, respectivamente.

Figura 28 – Resultados Collision 1000 bolas para seleção de VM na *cloudlet*

(a) Cenário 1 (b) Cenário 2 (c) Cenário 3



Nos resultados de 1000 bolas, também é seguida a mesma sequência de decisão da quantidade de 500 bolas, conforme apresentado na Figura 28. No primeiro cenário (Figura 28a), a VM com menor custo é a primeira, com ganho no tempo médio de 25,07% e 57,10% comparadas com a segunda e terceira VMs, respectivamente. No cenário 2 (Figura 28b), foi obtido o menor custo pela VM 3 onde se percebe nesta situação, como também para o tamanho de 500 bolas, que foi preciso um determinado nível de CPU e com o "Alto" valor do RTT para que as VMs com 2 vCPUs fossem descartadas da seleção. A terceira VM escolhida apresentou um ganho de 8,54% e 22,83% em relação às VMs 1 e 2. E, finalmente, no último cenário (Figura 28c) a VM que foi calculada o menor custo foi a primeira, obtendo um ganho percentual de 9,87% em relação à VM 2, que obteve o segundo maior FPS médio.

Tabela 12 – Média e intervalo de confiança do Collision nos cenários avaliados

Tamanho	**Cenário**	**VM 1**		**VM 2**		**VM 3**	
		FPS médio (s)	**Custo**	**FPS médio (s)**	**Custo**	**FPS médio (s)**	**Custo**
500 bolas	1	30,89 ± 0,198	0,178	24,50 ± 0,097	0,391	16,43 ± 0,106	0,837
	2	23,64 ± 0,086	0,512	20,82 ± 0,220	0,606	25,04 ± 0,124	0,499
	3	27,90 ± 0,176	0,310	25,78 ± 0,032	0,376	25,01 ± 0,089	0,502
1000 bolas	1	26,54 ± 0,102	0,176	19,89 ± 0,020	0,402	11,38 ± 0,096	0,844
	2	18,52 ± 0,104	0,524	15,63 ± 0,100	0,598	20,25 ± 0,064	0,499
	3	22,93 ± 0,199	0,309	20,67 ± 0,058	0,374	20,10 ± 0,025	0,499

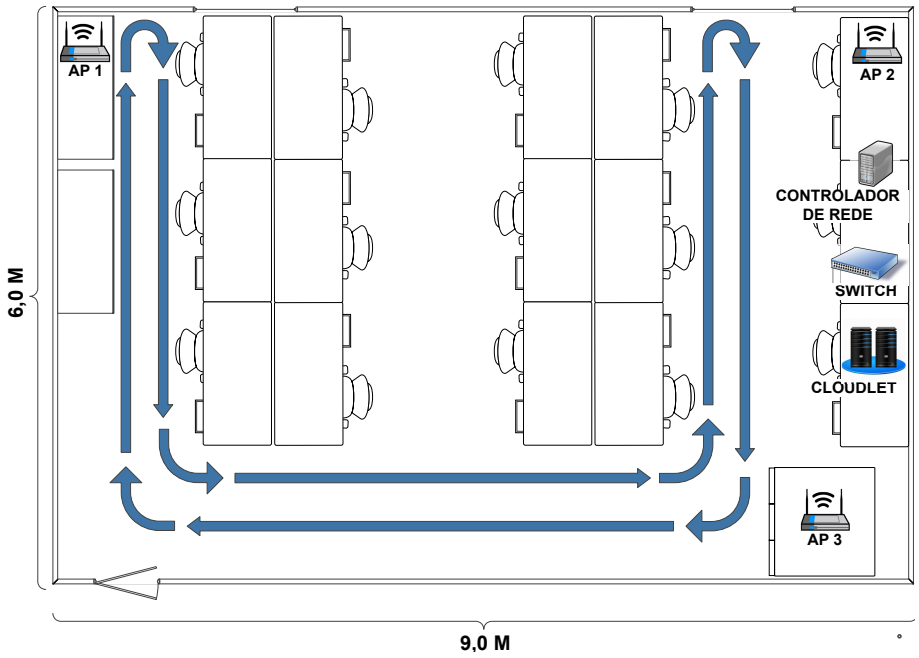
Na Tabela 12 são detalhados os valores de FPS médios de *offloading* obtidos para 500 e 1000 bolas. O intervalo de confiança é de 95% para cada VM. Além disso, também estão incluídos os valores de custo calculados pela fórmula em cada um dos cenários.

Através dos resultados obtidos, conclui-se que as decisões realizadas pela estratégia proposta contribuíram para obter a melhor experiência de *offloading* nos cenários avaliados. Também foi constatado que os pesos utilizados no balanceamento dos critérios da fórmula foram acertados, destacando que nesta aplicação foi utilizado o RTT, conforme verificado nas decisões nos cenários da Figura 27c e 28c. Além disso, da mesma forma que na aplicação BenchFace, com o Collision também foi possível identificar situações através dos valores atribuídos no critério *TP*, em que sejam mais vantajosa a seleção de uma VM com menor recurso de processamento, conforme se verifica a escolha da VM 3 na Figura 27b e 28b.

5.3.2 Experimento 2 - Seleção de rede com mobilidade

Este experimento avalia as aplicações móveis durante a seleção de rede sem fio, enquanto o usuário se move e executa as operações de *offloading*. O padrão de mobilidade adotado é apresentado na Figura 29. Como a estratégia padrão dos dispositivos é que seja disparado o *handoff* para a rede com maior RSSI, o sistema de decisão proposto não se restringe a este critério. Dessa forma, também é incluído no processo de decisão a vazão das redes disponíveis com o objetivo de tomar decisões mais adequadas. Os cenários são distintos para cada aplicação avaliada, mais detalhes são apresentados a seguir.

Figura 29 – Ambiente de testes para o Experimento 2



Fonte: Elaborada pelo Autor

- **BenchFace**

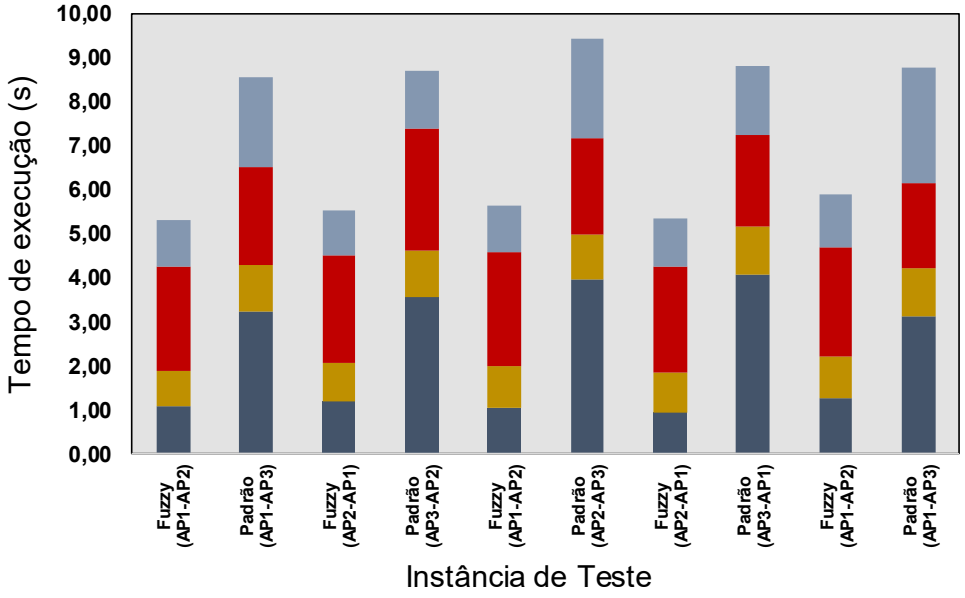
A aplicação BenchFace possui a característica de enviar a solicitação de *offloading* para a nuvem, e em seguida, aguardar o tempo de processamento dos dados a serem recebidos pelo dispositivo. O cenário utilizado para seleção de rede nesta aplicação, além de tomar decisões de *handoff* com base no RSSI e vazão dos APs disponíveis, também realiza o encaminhamento do resultado de *offloading* entre APs através das regras *OpenFlow*. Esta última funcionalidade é importante pois em determinadas situações, as execuções de *handoff* podem ser disparadas antes do recebimento dos dados processados na *cloudlet*, que foram solicitados no AP inicial. Dessa forma, além de selecionar uma nova rede ideal, o dispositivo não precisa solicitar novamente o processamento dos mesmos dados à nuvem.

Para esta avaliação, foram realizadas 30 execuções de *offloading* no cenário em que o dispositivo móvel se conecta inicialmente ao AP 1 e conforme se move, precisa decidir em executar o *handoff* na segunda ou na terceira rede sem fio (AP 2 e AP 3), e em seguida, realiza o percurso inverso. A Figura 30 apresenta os resultados obtidos em 5 execuções de *offloading* com as imagens de 3 MP e 8.5 MP. O rótulo "padrão" representam os testes executados com a decisão de *handoff* com base apenas no RSSI e "*fuzzy*" as execuções com o sistema proposto. Ainda nesta Figura, são detalhados o tempo de *upload* (envio dos dados à nuvem), de *download* (recebimento dos dados da nuvem), de processamento (processamento dos dados na nuvem) e de *handoff* (mudança de rede sem fio) para as duas estratégias avaliadas.

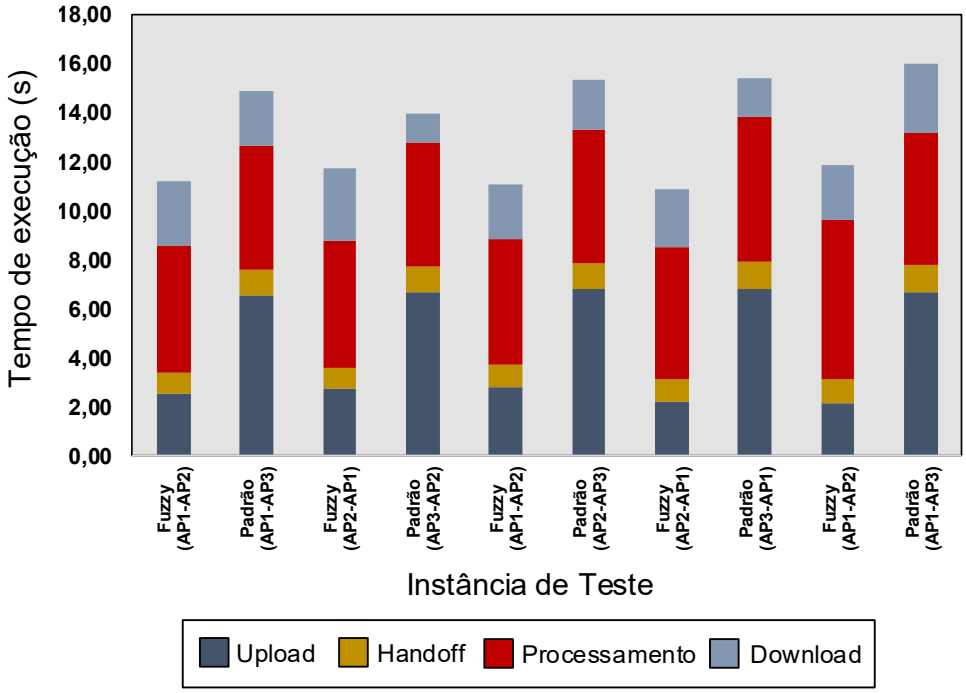
Nas execuções com a estratégia "padrão", conforme verificado nas Figuras 30a e 30b, o usuário realiza *upload* na primeira rede. No entanto, como não existe um sistema de decisão inteligente, a execução de *handoff* será feita com destino à segunda rede devido a esta possuir maior RSSI (AP 3). Como também nesta estratégia, não existe nenhuma técnica de encaminhamento dos dados processados da nuvem para a nova rede, nenhum *download* será recebido. Além disso, como apenas foi utilizado como critério de decisão o RSSI, esta rede selecionada não é ideal pois um tráfego médio de 25 Mbps está sendo gerado. Devido a estas características, o usuário precisará realizar um novo *upload* e aguardar o *download* que sofrerá impacto de variações de tempo, causando maior atraso na conclusão dessas atividades. Após receber o resultado, encerra-se este ciclo de *offloading*. Na segunda execução com a estratégia "padrão", o dispositivo encontra-se conectado ao AP 3 e executa um novo *upload*. Contudo, devido ao mesmo tráfego de 25 Mbps já referido, os dados também sofrerão variações de tempo para chegar na *cloudlet*. Como durante a mobilidade, houve redução do RSSI, o dispositivo executa o *handoff* para o AP 2 que possui maior nível de sinal. Devido aos motivos já mencionados, o resultado do processamento não é recebido pela nova rede conectada, sendo necessário um novo *upload*. Neste caso, a rede selecionada apresenta um tráfego médio de 5 Mbps, que é menor que a rede anteriormente conectada, proporcionando um *download* mais rápido.

Figura 30 – Resultados BenchFace para seleção de rede com mobilidade

(a) 3 MP



(b) 8.5 MP



No entanto, este ciclo de *offloading* foi prejudicado por ter sido iniciado numa rede com a vazão classificada pelo *fuzzy* como "Baixa".

Com a estratégia proposta nessa dissertação através da decisão pelo sistema *fuzzy*, o dispositivo inicia o primeiro ciclo de *offloading* executando o *upload* no AP 1. É importante destacar que só foi iniciada a solicitação remota nesta rede por ter sido previamente selecionada pelo sistema de decisão, evitando que sejam iniciados ciclos de *offloading* em redes que não sejam consideradas ideais. Em seguida, ocorre a situação em que antes

do *download*, o sistema *fuzzy* identifica a degradação do RSSI da rede atual e seleciona o AP 2 por ser a única rede a apresentar a vazão classificada como "Alta". Como nos testes executados o tempo de *handoff* foi menor que o tempo de processamento, as regras *OpenFlow* redirecionam os dados para o novo AP selecionado. Esta estratégia permite que o dispositivo faça o *download* da mesma solicitação de *offloading* que foi realizado na primeira rede, evitando repetir uma nova tarefa de *upload*. Em seguida, o segundo ciclo de *offloading* já será iniciado numa rede ideal definido para esta aplicação, proporcionando melhores condições de *upload*, ao contrário do que aconteceu no cenário com a mobilidade padrão. Em seguida, o usuário se move no ambiente da avaliação no sentido contrário, pois se conectada ao AP 2 e, em seguida, o sistema de decisão novamente executa o *handoff* no AP 1 por também apresentar os requisitos aceitáveis de RSSI e vazão definidos nos limiares *fuzzy*. Assim que conecta à nova rede, o dispositivo faz *download* do resultado pelo encaminhamento SDN.

A Tabela 13 apresenta o resumo dos resultados das avaliações realizadas com intervalo de confiança de 95%. É possível verificar que para a imagem de 3 MP, o ganho percentual no tempo médio de *offloading* da mobilidade "*fuzzy*" em relação à "padrão" foi de 37,47%. Para imagens de 8.5 MP, também houve ganho percentual no uso do sistema proposto nas execuções de *offloading* quando comparando à estratégia que apenas considera o RSSI, que foi de 25,01%.

Tabela 13 – Média e intervalo de confiança do BenchFace nos cenários avaliados

Ambiente Experimental	**Tamanho**	**Tempo médio (s)**
Mobilidade padrão	3 MP	8,84 ± 0,119
	8.5 MP	15,11 ± 0,270
Mobilidade com decisão *fuzzy*	3 MP	5,53 ± 0,083
	8.5 MP	11,33 ± 0,159

A avaliação da estratégia proposta utilizando o sistema *fuzzy* para seleção de rede durante a mobilidade do usuário e o encaminhamento dos dados processados pela *cloudlet* no novo AP selecionado, apresentaram ganhos consideráveis nas execuções de *offloading*. Além da redução no tempo médio das execuções remotas, a estratégia utilizando regras *OpenFlow*, evitou que no cenário avaliado, fosse preciso repetir solicitações de processamento na nuvem, proporcionando uma melhor experiência de *offloading* para o usuário móvel.

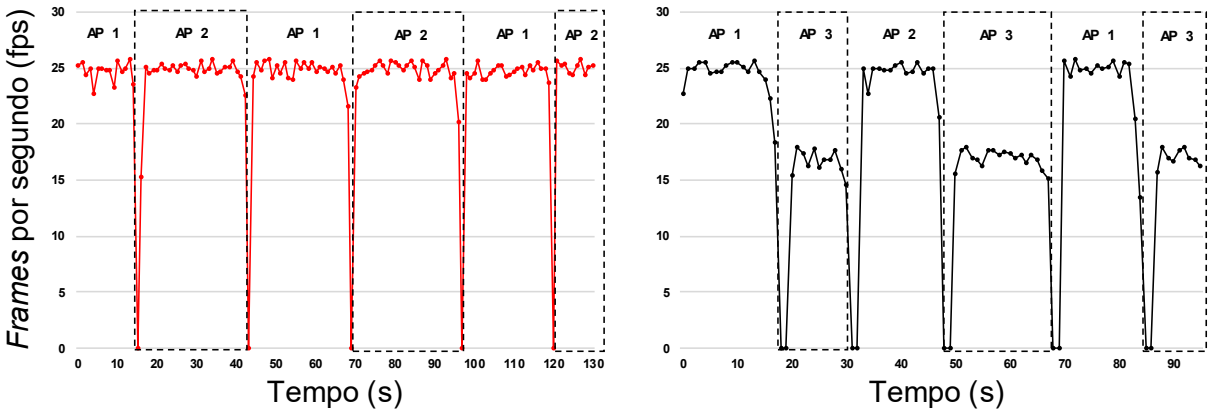
- **Collision**

A aplicação Collision possui como característica principal a constante interação com a nuvem durante as execuções de *offloading*. Dessa forma, torna-se essencial que as decisões de *handoff* sejam as mais eficazes para prover a melhor experiência possível ao usuário. O

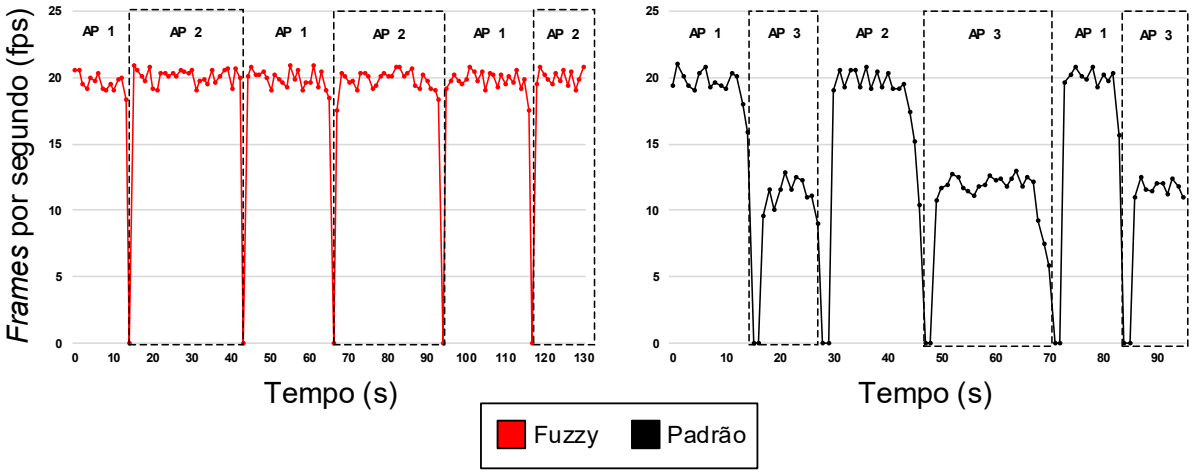
cenário definido para seleção de rede trata da mobilidade do dispositivo entre os APs no ambiente de execução, contendo três redes sem fio. Como a aplicação registra a quantidade de FPS, foram armazenados estes valores no intervalo de tempo que ocorreram 30 *handoffs*. As Figuras 31a e 31b apresentam uma amostra das execuções nas quantidades de 500 e 1000 bolas em 5 *handoffs*. A comparação do sistema proposto (*fuzzy*) que considera o RSSI e a vazão das redes, foi feita com a estratégia "padrão" que decide apenas com base no nível de sinal.

Figura 31 – Resultados Collision para seleção de rede com mobilidade

(a) 500 bolas



(b) 1000 bolas



Nas execuções sem o uso da proposta, o usuário inicia a execução de *offloading* no AP 1. Como nesta estratégia a seleção de rede é baseada apenas no RSSI, é possível observar nos gráficos que para ambas as quantidades de bolas, ocorre uma diminuição no número de FPS momentos antes das execuções de *handoff*. Este evento acontece devido à degradação do nível de sinal, prejudicando o acesso ao serviço de nuvem pelo dispositivo móvel, causado pelo distanciamento do AP conectado. Em seguida, quando o dispositivo se conecta ao AP 3, observa-se um atraso, que na percepção da aplicação, durou em torno de dois segundos para que fossem reiniciadas as tarefas de *upload* e *download* com a

cloudlet. A causa deste atraso é devido ao tempo de convergência que as redes tradicionais levam para reiniciar as conexões após o procedimento de *handoff*. Na nova rede conectada pelo dispositivo, a quantidade de FPS executados caiu quase pela metade, pois nesta rede está sendo gerado um tráfego médio de 25 Mbps. Também se percebe outra queda de FPS momentos antes do dispositivo disparar o *handoff* para o AP 2, que conforme já mencionado, ocorre devido à degradação do RSSI daquela rede.

Utilizando o sistema de decisão proposto para ambos os tamanhos de bolas, é possível verificar no gráfico que os valores de FPS são mais regulares, além das quedas de FPS antes das execuções de *handoff* também serem mais discretas. Estes comportamentos são mais evidentes quando comparados estes mesmos momentos sem o uso da proposta. Os dois benefícios referidos acontecem graças aos limiares *fuzzy* terem sido definidos com valores mais limitados, de modo a evitar ao máximo uma degradação do RSSI, que possa causar quedas consideráveis no desempenho da aplicação. Além disso, utilizando as regras *OpenFlow* os pacotes já são reencaminhados imediatamente, reduzindo o atraso de reconexão com a *cloudlet* para menos de dois segundos, um tempo menor que as execuções sem a proposta. Quando o dispositivo inicia sua mobilidade e as execuções de *offloading* no AP 1, o sistema de seleção decide disparar o *handoff* identificando também a vazão disponível nas redes do ambiente de execução. Dessa forma, é selecionado o AP 2 que apresenta um tráfego médio menor (5 Mbps) que o AP 3. As demais decisões de rede utilizando o sistema *fuzzy* também foram feitas utilizando esta estratégia, onde em seguida no cenário, foi selecionado o AP 1.

Apresentando os resultados com mais detalhes, a Tabela 14 detalha os valores do FPS médio nas 30 execuções de *handoffs* com intervalo de confiança de 95% durante as operações de *offloading*. Quando comparado o ganho percentual para 500 bolas utilizando a decisão *fuzzy* sem o uso da proposta, foi obtido o valor de 27,98%, e com 1000 bolas de 33,62%. A tabela também apresenta o percentual de perda de pacotes durante estas execuções, onde utilizando a proposta com 500 bolas, o ganho percentual comparado com a mobilidade padrão foi de 85,81%, e para 1000 bolas de 70,40%.

Tabela 14 – Média e intervalo de confiança do Collision nos cenários avaliados

Ambiente Experimental	**Tamanho**	**FPS médio**	**Perda de Pacotes (%)**
Mobilidade padrão	500 bolas	18,02 ± 1,313	8,25 ± 2,393
	1000 bolas	13,25 ± 1,471	13,34 ± 3,276
Mobilidade com decisão *fuzzy*	500 bolas	25,02 ± 0,182	1,17 ± 0,294
	1000 bolas	19,96 ± 0,192	3,95 ± 0,204

Através das execuções utilizando a estratégia do sistema proposto, também foi possível no Collision, melhorar o desempenho desta aplicação durante as execuções de *offloading*. Para ambas as quantidades de bolas (500 e 1000), verificam-se os benefícios dos limiares e regras *fuzzy* definidas, pois foi possível reduzir a degradação do FPS durante o uso da

aplicação, onde ocorre um tipo de antecipação na seleção de *handoff* para uma rede alvo. Além disso, a redução do atraso de reconexão do dispositivo com a *cloudlet* demonstra que as decisões do sistema *fuzzy* alinhadas com o encaminhamento de regras *OpenFlow* trouxeram grandes benefícios no cenário avaliado.

5.4 Considerações Finais

Este Capítulo apresentou os elementos que compuseram a avaliação da proposta dessa dissertação, incluindo o ambiente de testes, equipamentos de *hardware* e *software* utilizados, as aplicações de *benchmark* com os limiares e métricas definidos para as avaliações e por fim, o resultados dos experimentos. Foram realizados dois experimentos, o primeiro avaliou a seleção de rede e VM na *cloudlet* em cenários de contextos de RSSI, vazão, nível de CPU e memória e RTT. O segundo experimento avaliou o mecanismo de decisão durante a mobilidade do usuário. Em ambos os experimentos, foram variados o tipo de aplicação e tamanho dos dados de *offloading*. Os resultados comprovaram as melhorias do uso do sistema de decisão para as execuções de offloading através do menor tempo de resposta obtido na aplicação BenchFace e maior número de FPS com a aplicação Collision.

6 CONCLUSÃO

Este Capítulo resume os principais pontos discutidos nesta dissertação. As considerações finais deste trabalho são apresentadas na Seção 6.1. Na Seção 6.2, são descritas as contribuições obtidas, e, em seguida, na Seção 6.3, serão discutidos alguns pontos que podem ser explorados em trabalhos futuros.

6.1 Considerações Finais

Este trabalho apresentou um sistema de decisão de rede e de VM na *cloudlet* com base no tipo da aplicação em execução, visando proporcionar uma melhor experiência ao usuário durante as execuções de *offloading* no ambiente MCC. Com base na análise dos trabalhos relacionados foi possível abordar as diferentes estratégias de conectividade em MCC bem como as atuais propostas de seleção de rede e recursos de nuvem existentes, onde ficou claro que existem oportunidades de pesquisa a serem exploradas nestes aspectos discutidos. Dessa forma, foi evidenciada a necessidade de soluções que abranjam de forma satisfatória os principais critérios que degradam o desempenho das aplicações devido às necessidades distintas de recursos de QoS da rede e disponibilidade da nuvem pelas características distintas de cada uma. Foram apresentados os conceitos relacionados ao paradigma MCC detalhando a técnica de *offloading*, bem como as tecnologias que foram utilizadas na solução proposta, a saber, o protocolo *OpenFlow* do paradigma SDN e a Lógica *Fuzzy*.

Nesta dissertação, também foi apresentada a arquitetura da infraestrutura do *testbed* implementado, que permitiu principalmente a caracterização do ambiente MCC, detalhando os elementos existentes em cada uma das camadas, incluindo as funções delegadas em cada uma delas. Também foram abordados o funcionamento das etapas de decisão que foram implementadas no sistema proposto e como estão definidas as métricas e limiares utilizados nas estratégias em cada uma das etapas. Em seguida, foi destacada a maneira como cada tipo de aplicação é caracterizada dentro de cada etapa do sistema, além do esquema de sinalização do dispositivo com o controlador de rede e o pseudocódigo do algoritmo que gerencia o fluxo das etapas de decisão.

Através dos resultados da avaliação do sistema proposto em ambiente real, foi possível testificar a eficácia das decisões tomadas. Utilizando diferentes tipos de aplicações e tendo como foco analisar o comportamento das seleções de rede e VM na *cloudlet* em diferentes contextos do dispositivo móvel, verificou-se que as execuções de *offloading* foram mais vantajosas quando antes foram feitas as decisões de disparo de *handoff* e de mudança de VM na nuvem do que as execuções sem o uso do sistema apresentado nessa dissertação.

6.2 Contribuições

Esta seção apresenta as principais contribuições obtidas na proposta dessa dissertação.

A primeira contribuição foi a elaboração de um sistema de seleção de rede sem fio e VM na *cloudlet* ciente da aplicação móvel, permitindo obter decisões mais customizadas.

A segunda contribuição foi propor um sistema baseado em Lógica *Fuzzy* como gatilho de *handoff* para uma determinada rede, com base num perfil de funções de pertinência definidas de acordo com os requisitos de QoS de diferentes tipos de aplicação.

A terceira contribuição foi propor um cálculo de custo de *offloading* que permite através de pesos definidos utilizando o método AHP, distinguir entre tipos de VMs com quantidades de vCPUs distintas, qual deve ser selecionada para atender as solicitações de processamento do usuário.

A quarta contribuição relevante deste trabalho, foi propor uma estratégia de transparência de acesso à nova VM selecionada pelo sistema de decisão e o encaminhamento dos resultados de *offloading* entre APs utilizando regras *OpenFlow*.

6.3 Publicações

São apresentados a seguir, os artigos publicados relacionados a esta pesquisa.

- Bruno Silva, Atrícia Sabino, Warley M. Junior, Eduardo Oliveira, Francisco Júnior, Kelvin Dias - *Performance Evaluation of Cryptography on Middleware-Based Computational Offloading*; VII *Brazilian Symposium on Computing Systems Engineering*, 2017.

- Warley M. Junior, Bruno Silva, Kelvin Dias - *A Systematic Mapping Study on Mobility Mechanisms for Cloud Service Provisioning in Mobile Cloud Ecosystems; Elsevier Computers and Electrical Engineering*, 2018.

6.4 Trabalhos Futuros

Esta seção apresenta e discute possíveis extensões à pesquisa desenvolvida nesta dissertação.

- A implementação de uma estratégia de gerenciamento de mobilidade com execuções de *handoff* que evite quebras de conexões em aplicações de tempo real, evitando por exemplo, que o valor de FPS da aplicação Collision não apresente degradação durante as mudanças de APs. Dessa forma, seria melhorada a transição entre as redes.

- Estender o sistema de decisão proposto para que seja possível executar a seleção através de redes heterogêneas como, por exemplo, entre WiFi e 4G. Assim, o usuário estará conectado sempre na melhor rede, não importando a tecnologia empregada nelas, levando em consideração as características distintas (por exemplo, custo, atraso, consumo energético) de cada uma delas.

- Utilizar o sistema de decisão desta dissertação em ambientes de diferentes tipos de recursos de nuvem além de *cloudlets* como, por exemplo, nuvens públicas e ad-hoc. Com isso, o controlador SDN deve também capturar informações de cada um destes recursos, que possuem particularidades distintas, tais como, capacidade de processamento. Dessa forma, seria possível identificar quais destes ambientes deve ser escolhido para execução de *offloading*.

- Utilizar técnicas de aprendizagem de máquina junto com o sistema de decisão proposto para obter melhor acurácia na seleção de rede e VM na *cloudlet*. Através desta abordagem, é possível avaliar os principais classificadores sob uma base de dados composta de parâmetros relacionados às redes sem fio e nuvens disponíveis, permitindo também incluir o estado do dispositivo móvel como, por exemplo, o nível da bateria.

REFERÊNCIAS

ABDO, J. B.; DEMERJIAN, J. Evaluation of mobile cloud architectures. *Pervasive and Mobile Computing*, v. 39, p. 284 – 303, 2017. ISSN 1574-1192. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574119216304114>.

AHLGREN, B.; DANNEWITZ, C.; IMBRENDA, C.; KUTSCHER, D.; OHLMAN, B. A survey of information-centric networking. *IEEE Communications Magazine*, v. 50, n. 7, p. 26–36, July 2012. ISSN 0163-6804.

AHMED, E.; GANI, A.; KHAN, M. K.; BUYYA, R.; KHAN, S. U. Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications*, v. 52, p. 154 – 172, 2015. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804515000545>.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>.

ASHOK, A.; STEENKISTE, P.; BAI, F. Adaptive cloud offloading for vehicular applications. In: *2016 IEEE Vehicular Networking Conference (VNC)*. [S.l.: s.n.], 2016. p. 1–8.

BAHTOVSKI, A.; GUSEV, M. Cloudlet challenges. *Procedia Engineering*, v. 69, n. Supplement C, p. 704 – 711, 2014. ISSN 1877-7058. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877705814002914>.

BAHWAIRETH, K.; TAWALBEH, L.; BENKHELIFA, E.; JARARWEH, Y.; TAWALBEH, M. A. Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications. *EURASIP Journal on Information Security*, v. 2016, n. 1, p. 15, Jun 2016. ISSN 1687-417X. Disponível em: <https://doi.org/10.1186/s13635-016-0039-y>.

BARROS, L. de; BASSANEZI, R. *Tópicos de lógica fuzzy e biomatemática*. Grupo de Biomatemática, Instituto de Matemática, Estatística e Computação Científica (IMECC), Universidade Estadual de Campinas (UNICAMP), 2006. (Coleção IMECC. Textos Didáticos). ISBN 9788587185051. Disponível em: <https://books.google.com.br/books?id=5G9xBAAACAAJ>.

BARUA, A.; MUDUNURI, L.; KOSHELEVA, O. Why trapezoidal and triangular membership functions work so well: Towards a theoretical explanation. v. 8, p. 164–168, 08 2014.

CHATZIMILIOUDIS, G.; KONSTANTINIDIS, A.; LAOUDIAS, C.; ZEINALIPOUR-YAZTI, D. Crowdsourcing with smartphones. *IEEE Internet Computing*, v. 16, n. 5, p. 36–44, Sept 2012. ISSN 1089-7801.

CINGOLANI, P.; ALCALA-FDEZ, J. jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In: IEEE. *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*. [S.l.], 2012. p. 1–8.

CINGOLANI, P.; ALCALA-FDEZ, J. jfuzzylogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming. In: *International Journal of Computational Intelligence Systems*. [S.l.: s.n.], 2013. p. 61–75.

CISCO, I. Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021. *Cisco White paper*, 2017.

COMMUNITY. *WHAT'S RYU?* 2017. <hhttps://osrg.github.io/ryu/index.html>. [Online; acessado em 28-Dezembro-2017].

DINH, H. T.; LEE, C.; NIYATO, D.; WANG, P. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, v. 13, n. 18, p. 1587–1611, 2013. ISSN 1530-8677. Disponível em: <http://dx.doi.org/10.1002/wcm.1203>.

EBERHART, R. C. *Computational Intelligence: Concepts to Implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 1558607595, 9780080553832.

ENZAI, N. I. M.; TANG, M. A taxonomy of computation offloading in mobile cloud computing. In: *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. [S.l.: s.n.], 2014. p. 19–28.

ETSI. *Mobile-Edge Computing Introductory Technical White Paper*. 2014. Disponível em: <http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing>.

FARHADY, H.; LEE, H.; NAKAO, A. Software-defined networking: A survey. *Computer Networks*, v. 81, p. 79 – 95, 2015. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128615000614>.

FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: A survey. *Future Generation Computer Systems*, v. 29, n. 1, p. 84 – 106, 2013. ISSN 0167-739X. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X12001318>.

GANI, A.; NAYEEM, G. M.; SHIRAZ, M.; SOOKHAK, M.; WHAIDUZZAMAN, M.; KHAN, S. A review on interworking and mobility techniques for seamless connectivity in mobile cloud computing. *Journal of Network and Computer Applications*, v. 43, p. 84 – 102, 2014. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804514000927>.

GUSTAFSSON, E.; JONSSON, A. Always best connected. *IEEE Wireless Communications*, v. 10, n. 1, p. 49–55, Feb 2003. ISSN 1536-1284.

HA, K.; PILLAI, P.; RICHTER, W.; ABE, Y.; SATYANARAYANAN, M. Just-in-time provisioning for cyber foraging. In: *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA: ACM, 2013. (MobiSys '13), p. 153–166. ISBN 978-1-4503-1672-9. Disponível em: <http://doi.acm.org/10.1145/2462456.2464451>.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011.

HU, X.; CHU, T. H. S.; LEUNG, V. C. M.; NGAI, E. C. H.; KRUCHTEN, P.; CHAN, H. C. B. A survey on mobile social networks: Applications, platforms, system architectures, and future research directions. *IEEE Communications Surveys Tutorials*, v. 17, n. 3, p. 1557–1581, thirdquarter 2015. ISSN 1553-877X.

JANG, J. R. *MATLAB: Fuzzy logic toolbox user's guide: Version 1*. [S.l.]: Math Works, 1997.

JEMAL, H.; KECHAOU, Z.; AYED, M. B.; ALIMI, A. M. Cloud computing and mobile devices based system for healthcare application. In: *2015 IEEE International Symposium on Technology and Society (ISTAS)*. [S.l.: s.n.], 2015. p. 1–5.

JUNIOR, A. H. W.; LOPES, K. Avaliação de desempenho da técnica de offloading computacional em nuvens móveis. *14º WPerformance - XIV Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, 2015.

JÚNIOR, E. C. d. A. Wi-Flow: uma arquitetura baseada em SDN para o gerenciamento e autenticação 802.1x . *Dissertação de Mestrado - Universidade Federal de Pernambuco*, 2016. Disponível em: <https://repositorio.ufpe.br/handle/123456789/18587>.

JUNIOR, W.; HENRIQUE, A.; DIAS, K.; SOUZA, J. N. de. Supporting mobility-aware computational offloading in mobile cloud environment. *Journal of Network and Computer Applications*, v. 94, p. 93 – 108, 2017. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804517302321>.

JUNIOR, W.; SILVA, B.; DIAS, K. A systematic mapping study on mobility mechanisms for cloud service provisioning in mobile cloud ecosystems. *Computers & Electrical Engineering*, p. –, 2018. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790617317470>.

KHALAJ, A.; LUTFIYYA, H. Handoff between proxies in the proxy-based mobile computing system. In: IEEE. *MOBILe Wireless MiddleWARE, Operating Systems and Applications (Mobilware), 2013 International Conference on*. [S.l.], 2013. p. 10–18.

KHAN, A. u. R.; OTHMAN, M.; MADANI, S. A.; KHAN, S. A survey of mobile cloud computing application models. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 393–413, First 2014. ISSN 1553-877X.

KHONDOKER, R.; ZAALOUK, A.; MARX, R.; BAYAROU, K. Feature-based comparison and selection of software defined networking (SDN) controllers. In: *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. [S.l.: s.n.], 2014. p. 1–7.

KOSKO, B.; ISAKA, S. Fuzzy logic. *Scientific American*, JSTOR, v. 269, n. 1, p. 76–81, 1993.

KREUTZ, D.; RAMOS, F. M. V.; VERISSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 0018-9219.

KUMAR, K.; LIU, J.; LU, Y.-H.; BHARGAVA, B. A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, v. 18, n. 1, p. 129–140, Feb 2013. ISSN 1572-8153. Disponível em: <https://doi.org/10.1007/s11036-012-0368-0>.

KUMAR, K.; LU, Y. H. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, v. 43, n. 4, p. 51–56, April 2010. ISSN 0018-9162.

LEE, D.; LEE, H.; PARK, D.; JEONG, Y.-S. Proxy based seamless connection management method in mobile cloud computing. *Cluster Computing*, v. 16, n. 4, p. 733–744, Dec 2013. ISSN 1573-7543. Disponível em: <https://doi.org/10.1007/s10586-013-0249-8>.

LI, J.; BU, K.; LIU, X.; XIAO, B. Enda: Embracing network inconsistency for dynamic application offloading in mobile cloud computing. In: ACM. *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. [S.l.], 2013. p. 39–44.

LI, W.; ZHAO, Y.; LU, S.; CHEN, D. Mechanisms and challenges on mobility-augmented service provisioning for mobile cloud computing. *IEEE Communications Magazine*, v. 53, n. 3, p. 89–97, March 2015. ISSN 0163-6804.

MA, X.; CUI, Y.; WANG, L.; STOJMENOVIC, I. Energy optimizations for mobile terminals via computation offloading. In: *2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing*. [S.l.: s.n.], 2012. p. 236–241.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

MENDEL, J. M. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, v. 83, n. 3, p. 345–377, Mar 1995. ISSN 0018-9219.

MITRA, K.; SAGUNA, S.; ÅHLUND, C.; LULEå, D. G. M2c2: A mobility management system for mobile cloud computing. In: *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.: s.n.], 2015. p. 1608–1613. ISSN 1525-3511.

ONF, O. N. F. *Software-Defined Networking (SDN) Definition*. 2017. <https://www.opennetworking.org/sdn-definition/>. [Online; acessado em 28-Dezembro-2017].

ONF, O. N. F. *Specifications*. 2017. <https://www.opennetworking.org/software-defined-standards/specifications/>. [Online; acessado em 29-Dezembro-2017].

ONG, E. H.; KHAN, J. Y. Dynamic access network selection with QoS parameters estimation: A step closer to ABC. p. 2671–2676, May 2008. ISSN 1550-2252.

OSRG. *Using Ryu Network Operating System with OpenStack as Network controller*. 2017. <https://github.com/osrg/ryu/wiki/OpenStack/>. [Online; acessado em 28-Dezembro-2017].

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 414–454, First 2014. ISSN 1553-877X.

RAVI, A.; PEDDOJU, S. K. Mobility managed energy efficient android mobile devices using cloudlet. In: *Students' Technology Symposium (TechSym), 2014 IEEE*. [S.l.: s.n.], 2014. p. 402–407.

RAVI, A.; PEDDOJU, S. K. Handoff strategy for improving energy efficiency and cloud service availability for mobile devices. *Wireless Personal Communications*, v. 81, n. 1, p. 101–132, Mar 2015. ISSN 1572-834X. Disponível em: <https://doi.org/10.1007/s11277-014-2119-y>.

ROSS, T. J. *Fuzzy logic with engineering applications*. [S.l.]: John Wiley & Sons, 2009.

SAAD, H. B.; KASSAR, M.; SETHOM, K. Always best connected and served based scheme in mobile cloud computing. In: *2016 3rd Smart Cloud Networks Systems (SCNS)*. [S.l.: s.n.], 2016. p. 1–8.

SAMAD, J.; LOKE, S. W.; REED, K. *Mobile Cloud Computing*. Cloud Services, Networking, and Management, 2015. 153–190 p. (Book Chapter). ISBN 9781119042655. Disponível em: <http://dx.doi.org/10.1002/9781119042655.ch7>.

SATYANARAYANAN, M. Fundamental challenges in mobile computing. In: *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: ACM, 1996. (PODC '96), p. 1–7. ISBN 0-89791-800-2. Disponível em: <http://doi.acm.org/10.1145/248052.248053>.

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, v. 8, n. 4, p. 14–23, Oct 2009. ISSN 1536-1268.

SDXCENTRAL. *What is Ryu Controller?* 2017. <https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-ryu-controller/>. [Online; acessado em 28-Dezembro-2017].

TANSCHEIT, R. Sistemas fuzzy. VI Simpósio Brasileiro de Automação Inteligente (SBAI'03), Bauru, SP., 2004.

TZAFESTAS, S. G.; BLEKAS, K. D. Hybrid soft computing systems: a critical survey with engineering applications. *National Technical University of Athens*, 1997.

VERBELEN, T.; SIMOENS, P.; TURCK, F. D.; DHOEDT, B. Aiolos: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, v. 85, n. 11, p. 2629 – 2639, 2012. ISSN 0164-1212. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121212001641>.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* IEEE Comput. Soc, 2001. v. 1, p. I–511–I–518. ISBN 0-7695-1272-0. ISSN 0920-5691. Disponível em: <http://ieeexplore.ieee.org/document/990517/>.

YAN, X.; ŞEKERCIOğLU, Y. A.; NARAYANAN, S. A survey of vertical handover decision algorithms in fourth generation heterogeneous wireless networks. *Computer Networks*, v. 54, n. 11, p. 1848 – 1863, 2010. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610000502>.

YANG, L.; CAO, J.; TANG, S.; HAN, D.; SURI, N. Run time application repartitioning in dynamic mobile cloud environments. *IEEE Transactions on Cloud Computing*, v. 4, n. 3, p. 336–348, July 2016. ISSN 2168-7161.

ZADEH, L. A. Fuzzy sets. *Information and control*, Academic press, v. 8, n. 3, p. 338–353, 1965.

ZANAKIS, S. H.; SOLOMON, A.; WISHART, N.; DUBLISH, S. Multi-attribute decision making: A simulation comparison of select methods. *European Journal of Operational Research*, v. 107, n. 3, p. 507 – 529, 1998. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221797001471>.

ZHOU, B.; DASTJERDI, A. V.; CALHEIROS, R. N.; SRIRAMA, S. N.; BUYYA, R. A context sensitive offloading scheme for mobile cloud computing service. In: *2015 IEEE 8th International Conference on Cloud Computing*. [S.l.: s.n.], 2015. p. 869–876. ISSN 2159-6182.