UNIVERSIDADE ABERTA
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

UMA PROPOSTA DE ESCALONAMENTO DE TAREFAS SENSÍVEL AO CONTEXTO DE APLICAÇÕES MÓVEIS NO PARADIGMA *FOG COMPUTING*

Celestino Lopes de Barros

Doutoramento em Ciência e Tecnologia Web

(doutoramento em associação)

2020

UNIVERSIDADE ABERTA
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

UMA PROPOSTA DE ESCALONAMENTO DE TAREFA SENSÍVEL AO CONTEXTO DE APLICAÇÕES MÓVEIS NO PARADIGMA *FOG COMPUTING*

Celestino Lopes de Barros

Doutoramento em Ciência e Tecnologia Web

(doutoramento em associação)

Tese orientada pelos Professores Doutores Hugo Alexandre Paredes Guedes da Silva, Vítor Jorge Ramos Rocio e André Filipe Esteves de Sousa.

2020

RESUMO

Os pedidos de execução de aplicações na arquitetura *cloud* e no paradigma *fog* são geralmente heterogéneos em termos de contextos ao nível dos dispositivos e das aplicações. O escalonamento dos pedidos nessas arquiteturas é um problema de otimização com múltiplas restrições. Apesar dos inúmeros esforços, o escalonamento de tarefas nessas arquiteturas e paradigmas continua a apresentar alguns desafios aliciantes que nos levam a questionar a forma como as tarefas são encaminhadas entre os diferentes dispositivos físicos, nós da *fog* e *cloud*. A *fog* é definida como uma extensão da *cloud*, que disponibiliza serviços de processamento, armazenamento e rede próximo da *edge network*, e devido à densidade e heterogeneidade de dispositivos, o escalonamento é muito complexo e, na literatura, encontramos ainda poucos estudos. Contrariamente, o escalonamento na *cloud* é amplamente estudado. Diversos trabalhos de investigação abordam, no entanto, essa questão na perspetiva de provedores de serviço ou otimizam os níveis da qualidade de serviço (QoS) da aplicação. Ignoram, porém, informações contextuais ao nível do dispositivo e dos utilizadores finais e as suas experiências de utilização (QoE).

Procurando trazer contributos inovadores nas áreas de escalonamento de tarefas e computação distribuída, nesta tese, é proposta uma solução para o problema de escalonamento de pedidos sensível ao contexto para o paradigma *fog* que minimiza os tempos de execução da aplicação e maximiza as suas prioridades. Os diferentes parâmetros de contexto são normalizados através da normalização *Min-Max*. A prioridade de cada pedido é definida através da aplicação da técnica de análise *Multiple Linear Regression* (MLR) e o seu escalonamento com vista a otimizar a QoE dos utilizadores, é feito recorrendo a técnica de Otimização *Multi-Objective Non-Linear Programming*[1] (MONLP). Os resultados experimentais, encontrados a partir de simulações no *kit* de ferramentas *iFogSim*, demonstram que a nossa proposta de escalonamento apresenta um melhor desempenho em comparação com as propostas não sensível ao contexto (*First Come First Served*, *Shortest Job First e QoS-based*), relativamente às métricas: percentagem de execução dos pedidos com sucesso, tempo de espera e QoE dos utilizadores.

Palavras-chave: Qualidade de experiência; Sensibilidade ao contexto; Escalonamento de tarefas; *Cloud computing; Fog computing*.

[1] Tradução livre do autor: “Programação Não Linear Multiobjetivo”.

ABSTRACT

Application execution requests in cloud architectures and fog paradigm are generally heterogeneous in terms of device and application contexts, and the scheduling of requests in these architectures is an optimization problem with multiple constraints. On the other hand, despite the numerous efforts, task scheduling in these architectures continue to present some enticing challenges that lead us to question how tasks are routed between different physical devices, fog nodes and *cloud*. Fog is defined as an extension of the cloud, which provides processing, storage and network services near the edge network, and due to its density and heterogeneity of devices, the scheduling is very complex and in the literature, there are still few studies that have been conducted. Conversely, scheduling in the cloud has been widely studied. Nonetheless, many surveys address this issue from the perspective of service providers or optimize application quality of service (QoS) levels of the application. In addition, they ignore contextual information at the level of the device, end-users and their user experiences (QoE).

Seeking to bring innovative contributions in the areas of task scheduling and distributed computing, in this thesis, we propose a solution to the problem of context-aware requisition scheduling for fog paradigm that minimizes application execution times (i.e. optimizes QoE) and maximizes its priorities. The different context parameters are normalized using Min-Max normalization. The priority of each request is defined through the application of the Multiple Linear Regression analysis technique and the scheduling of the requests in order to optimize the users QoE, respecting the various constraints, is made using the multi-objective non-linear programming optimization technique.

Our experimental results, obtained from simulation executions in the iFogSim toolkit, demonstrate that our scheduling proposal performs better than non-context-sensitive proposals (FCFS, SJF e QoS-based) in terms of metrics: success rate, waiting time and user QoE.

Keywords: Quality of experience; Context awareness; Task scheduling; Cloud computing; Fog computing

DEDICATÓRIA

Aos meus queridos pais, aos meus irmãos companheiros, aos meus filhos maravilhosos e a minha incansável e imprescindível esposa.

AGRADECIMENTOS

Foram muitas as pessoas que me ajudaram durante o desenvolvimento desta tese. A todos o meu agradecimento:

À Fundação *Calouste Gulbenkian*, por ter acreditado no meu projeto e me ter apoiado com uma bolsa de doutoramento, que permitiu financiar os meus estudos durante esta fase.

Aos meus professores e orientadores, Doutor Hugo Alexandre Paredes Guedes da Silva, da UTAD, Doutor Vítor Jorge Ramos Rocio, da UAb, e Doutor André Filipe Esteves de Sousa, da *Critical TechWorks*, pela ótima orientação que concederam à realização desta tese. A eles devo todo o apoio e confiança ilimitados que me deixaram sempre entusiasmado para levar avante o trabalho.

Aos meus pais, Venâncio Afonso Barros e Maria Tereza Barros, aos meus irmãos Lucindo, Felismino, Auriza, Edmilson, Júlio e Sandra, à minha esposa Eloisa e os meus filhos Celestino e Eloi pela força, confiança, amor e incentivo e por suportarem meu mau humor em certos momentos.

Gostaria também de agradecer aos professores Emanuel de Pina, Arminda Brito, Anick da Cruz e Astrigilda Silveira por terem lido com paciência a nossa tese e terem corrigido os erros de linguagem em todo o texto.

Finalmente, resta agradecer aos meus colegas de curso e a todos aqueles que, diretamente ou indiretamente, contribuíram para a realização deste trabalho.

A Deus por tudo.

O trabalho desenvolvido nesta tese foi financiado pela Fundação *Calouste Gulbenkian* na forma de uma bolsa de doutoramento com a referência Nº. 234242, 2019 - Bolsas de Pós-Graduação, destinadas aos estudantes dos PALOP e Timor-Leste.

ÍNDICE GERAL

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

ABREVIATURAS, ACRÓNIMOS E SIGLAS

Nesta tese são utilizadas abreviaturas, siglas e acrónimos de designações comuns, apresentadas aquando da sua primeira utilização, na sua denominação original.

Abreviaturas, Acrónimos e Siglas	Descrição
APIs	*Application Programming Interfaces*
ARM	*Advanced RISC Machine*
BLA	*Bees Life Algorithm*
BS/AP	*Base Station /Access Point*
CMST	*Cooperative-Based Model for Smart–Sensing Tasks*
DDF	*Distributed Data Flow*
DEBTS	*Delay Energy Balanced Task Scheduling*
DSR	*Design Science Research*
EDF	*Earliest Deadline First*
ETSI	*European Telecommunications Standards Institute*
EWMA	*Exponentially Weighted Moving Average*
FCFS	*First Come First Served*
FO	Função Objetivo
Fog-EDF	*Baseline Cloud-Unaware Strategy*
FUGE	*FUzzy GEnetic Task Scheduling*
GPS	*Global Positioning System*
HPJF	*Highest Priority Job First*
Hybrid-EDF	*Hybrid Fog and Cloud-Aware Heuristic*
IaaS	*Infrastructure as a Service*
IE	Intervalo de Escalonamento
IoT	*Internet of Things*
ISSO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
MCC	*Mobile Cloud Computing*
MDC	*Micro Data Centers*

Abreviaturas, Acrónimos e Siglas	Descrição
MEC	*Mobile Edge Computing*
MeFoRE	*Media fog Resource Estimation*
MI	Milhões de Instruções
MIPS	Milhões de Instruções por Segundos
MONLP	*MultiObjective Non-Linear Programming*
MOS	*Mean Opinion Score*
MVs	Máquinas Virtuais
NFV	*Network Functions Virtualization*
P2P	*Peer-to-Peer*
PaaS	*Platform as a Service*
PTMM	*Priority-Based Two-Phase Min–Min Scheduling Policy*
QCASH	*QoE and Context-Aware Scheduling*
QI	Questões de Investigação
QoC	Qualidade de Contexto
QoE	Qualidade de Experiência
QoS	Qualidade de Serviço
QoS-based	*QoS based Priority Scheduling*
RAS	Fiabilidade, Disponibilidade e Manutenibilidade
RISC	*Reduced Instructions Set Computers*
MLR	*Multiple Linear Regression*
SaaS	*Software as a Service*
SDN	*Software Defined Networking*
SJF	*Shortest Job First*
SLAs	*Service Level Agreements*
SOA	*Service-Oriented Architecture*
SQL	*Structured Query Language*
TI	Tecnologias de Informação

1. INTRODUÇÃO

Neste primeiro capítulo, fizemos um enquadramento do trabalho desenvolvido ao longo desta tese, apresentamos as motivações, os objetivos para a sua realização e definimos as suas principais contribuições. Este capítulo culmina com a apresentação da estrutura da tese, complementada com uma descrição resumida dos capítulos.

CAPÍTULO 1 - *Introdução*

1.1. Enquadramento

As técnicas atuais de computação que utilizam a *cloud* estão-se a tornar insustentáveis, visto que bilhões de dispositivos, impulsionados sobretudo pelo rápido crescimento da *Internet of Things (IoT)* estão conectados à Internet. Os dados obtidos pelos sensores e aplicações têm aumentado exponencialmente e muitos destes dispositivos permitem agregar num único aparelho funcionalidades de execução de aplicações, comunicação, entretenimento, jogos, entre outros. Além disso, uma das suas principais características é a sua capacidade de identificar e partilhar diferentes tipos de informações ao nível de utilizadores, dispositivo, aplicações e rede. Por outro lado, eles possuem várias limitações como: capacidade de processamento reduzida, escassez de recursos, autonomia reduzida da bateria, baixa conetividade, entre outras. Estas limitações impõem aos analistas e desenvolvedores a adoção de serviços que ampliam a capacidade dessas aplicações em execução nesses dispositivos através da utilização de serviços hospedados na *cloud* (Fernando, Loke, & Rahayu, 2013).

Não obstante as vantagens, em algumas situações, não é benéfica a utilização da arquitetura *cloud*. Ela é centralizada e consequentemente o processamento é feito em *data centers* concentrados, para otimização de custos energéticos e de comunicações. Diferentes técnicas que minimizam a execução na *cloud* através do processamento local em elementos periféricos que permitem resolver limitações deste modelo têm sido propostas. E uma dessas técnicas passa pela utilização do paradigma *fog* (Bonomi *et al*., 2014).

Segundo OpenFog (2017), a *fog* é uma extensão da *cloud* e tem sido proposta para colmatar as inconveniências deste, visto que visa reduzir, entre outros, o tempo de resposta, consumo de energia e latência. No entanto, devido a sua densidade e heterogeneidade o escalonamento de tarefas precisa ser tratado com perícia.

Assim, visando a formulação da proposta de um modelo de escalonamento sensível ao contexto para o paradigma *fog* que tenha como âmbito a criação de conhecimento na forma de técnicas, métodos, modelos e teoria, para além da revisão da literatura sobre os principais algoritmos de escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog*, estudámos e discutimos as suas limitações, explorámos e sugerimos algumas perspetivas de melhorias, constituindo-se subsídios muito úteis para o seu desenvolvimento.

Atendendo aos objetivos traçados e à procura de respostas às questões de investigação que a conduziram, optamos, em termos metodológicos, pelo *Design Science Research (DSR)*, que consiste em definir a investigação na perspetiva de criação de artefactos.
A dependência das execuções das tarefas descarregadas na *fog*, com diferentes parâmetros de contexto, estão fora das nossas questões de investigação. Assim, a tónica deste trabalho de investigação coloca ênfase na sensibilidade ao contexto no escalonamento de tarefas no paradigma *fog*.

Focalizando a nossa atenção neste aspeto, definimos nas secções 1.2 e 1.3, o nosso problema e questões de investigação.

1.2. Problema de investigação

Os pedidos descarregados na *fog* são heterogéneos em termos do contexto do dispositivo, do utilizador final e dos requisitos da aplicação. A otimização da QoE e o escalonamento de tarefas heterogéneas e sensível ao contexto no paradigma *fog*, é um desafiante problema de investigação. Ademais, o escalonamento de tarefas não-preemptivas e sensível ao contexto é um problema de otimização com múltiplas restrições devido aos contextos associados a um pedido (por exemplo, de nível de bateria, de sinal da rede, de requisito de QoS, entre outros) e a configuração da *fog*.
Assim, nesta tese, abordamos o problema do escalonamento de aplicações móveis sensível ao contexto no paradigma *fog.* A política de escalonamento sensível ao contexto proposta possui condições para ser implementada em ambientes práticos. Pois, é eficiente em relação à quantidade de tarefas escalonadas e que satisfazem a QoS, diminuição do tempo de resposta, otimização de QoE, redução do tempo médio de espera, entre outros, em comparação com as políticas de escalonamentos não sensível ao contexto. Assim, esta tese pode ser enunciada como:

Uma proposta de escalonamento de tarefas sensível ao contexto de aplicações móveis no paradigma fog computing.

1.3. Questões de investigação

Esta tese tem associado um conjunto de questões de investigação (**QI**), orientadoras deste trabalho de investigação, apresentadas em seguida:

QI-1. Quais os diferentes tipos de contexto em computação móvel e como definir os parâmetros que se associam ao nosso domínio do problema?

QI-2. Quais os principais algoritmos de escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog*?

QI-3. Quais as limitações dos principais algoritmos de escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog* e como esses algoritmos podem ser aperfeiçoados?

QI-4. Como normalizar e resolver o problema da heterogeneidade dos pedidos em termos do contexto do dispositivo do utilizador final e dos requisitos da aplicação?

QI-5. Como definir a prioridade das tarefas sensível ao contexto para os recursos disponíveis?

QI-6. Como otimizar a QoE dos utilizadores através da exploração das informações do contexto como: o nível de bateria, intensidade do sinal da rede e QoS da aplicação?

1.4. Objetivos e contribuições

Nesta secção, são apresentados o objetivo geral e um conjunto de objetivos específicos e contribuições que visam atingir esse objetivo geral.

1.4.1. Objetivo geral

O objetivo principal desta tese, consiste em propor um modelo de escalonamento de aplicações móveis sensível ao contexto para o paradigma *fog* que visa otimizar a qualidade de experiência dos utilizadores.

1.4.2. Objetivos específicos e contribuições

Pretendemos como objetivos específicos, resultantes da principal questão desta tese, bem como as questões de investigação derivadas:

- Aprofundar e contribuir para o estado da arte com o conhecimento sobre o contexto no âmbito da aplicação aos dispositivos móveis, os seus principais desafios, sensibilidade ao contexto, e ciclo de vida contextual. Também, sobre a *cloud computing* e as suas tecnologias emergentes, destacando a *fog computing* a sua necessidade, benefícios, características e arquitetura base;
- Identificar os principais algoritmos de escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog*, estudar e discutir as suas limitações, explorar e sugerir algumas perspetivas de melhorias;
- Propor uma arquitetura de escalonamento de tarefas sensível ao contexto para o paradigma *fog* que otimiza a QoE dos utilizadores;
- Utilizar a normalização *Min-Max* para normalizar e resolver a heterogeneidade dos diferentes parâmetros de contexto;
- Determinar a prioridade de contexto dos pedidos através da análise da MLR;
- Escalonar os pedidos com vista a otimizar a QoE dos utilizadores, respeitando as várias restrições ao nível da aplicação e serviço através da utilização da técnica MONLP;

- Definir os parâmetros do sistema a fim de possibilitar um escalonamento eficiente, que garanta uma boa utilização dos recursos, bem como a execução eficaz das tarefas;
- Conceber comparações experimentais e teóricas entre o escalonamento sensível e não sensível ao contexto, em termos das seguintes métricas: percentagem dos pedidos executados com sucesso, tempo de espera e QoE.

1.5. Metodologia de investigação

A investigação científica é definida como atividade que contribui para a compreensão de um determinado fenómeno que é, tipicamente, um conjunto de comportamentos ou entidades que apresenta relevância e interesse por parte da comunidade científica (Kuhn & Hacking, 2012). Para esta contribuição ser válida, deve ser aceite pelos pares, através de disseminações científicas devendo, por isso, apresentar um grau elevado de originalidade para essa comunidade (Gregor & Hevner, 2013).

Assim, também as atividades realizadas durante a investigação devem ser alvo de escrutínio científico, pois apresentam uma grande relevância na qualidade do produto final, sendo designadas por metodologias de investigação.

Sendo o objetivo principal desta tese, a elaboração de uma proposta que tenha como âmbito a criação de conhecimento na forma de técnicas, métodos, modelos e teoria, a metodologia que mais se adequa é o DSR, que visa definir a investigação sob o ponto de vista de criação de artefactos. A investigação que utiliza a ciência para criação de artefactos, também cria o conhecimento exigido para essa construção, através de técnicas de *design*, análise, reflexão e abstração (Hevner & Chatterjee, 2010).

De uma forma geral, de acordo com os autores citados, a investigação segundo a metodologia DSR, pode ser representada conforme a figura 1.1.

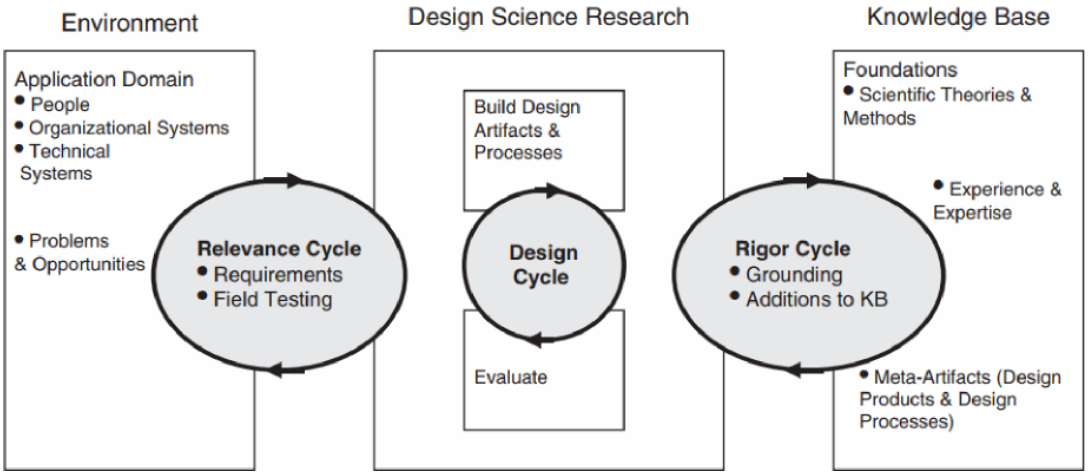


Figura 1.1: Fases da metodologia de investigação baseada em DSR
Fonte: (Hevner & Chatterjee, 2010).

A figura 1.1, realça a contribuição do novo conhecimento como um fato chave na aplicação da metodologia sendo que, de uma forma geral, o processo segue as seguintes fases:

- *Consciência do problema,* perceção da origem do problema, as necessidades ou propostas existentes. Esta etapa possui como resultado uma promessa para a atividade de investigação.
- *Sugestão*, é uma fase criativa onde se começa a esboçar o artefacto, com base nos elementos existentes ou com vista aos elementos a serem criados.
- *Desenvolvimento*, é o desenvolvimento efetivo do projeto, deve seguir métodos e técnicas do estado da arte. A novidade e a contribuição para o conhecimento científico devem estar nos aspetos do artefacto e não nas técnicas de construção e desenvolvimento.
- *Avaliação*, a avaliação dos artefactos, de forma qualitativa e quantitativa, com vista à comparação com os pressupostos do que era esperado na fase de proposta inicial. Esta fase tem como missão confirmar e contradizer as hipóteses.
- *Conclusão,* fase final da investigação, que ocorre tipicamente quando todos os períodos de investigação se encontram satisfatoriamente atingidos, apesar da possibilidade de poderem ter ocorrido diversos desvios e alterações à proposta inicial.

- *Contribuição de conhecimento (Divulgação),* validação do trabalho feito pelos pares, a qual necessita de detalhe suficiente para permitir a replicação do artefacto proposto e implementado.

A metodologia apresentada agrupa uma sequência de diferentes fases, que se repetem, de forma iterativa, levando ao aperfeiçoamento do artefacto final com o conhecimento das avaliações anteriores.

A tabela 1.1, sintetiza as etapas de investigação levadas a cabo neste trabalho, em consonância com a metodologia de investigação DSR.

Tabela 1.1: Fases da implementação da metodologia de investigação DSR

(continua)

Etapa	Descrição
Consciência do problema	Propor e validar um modelo de escalonamento sensível ao contexto para o paradigma *fog* que visa otimizar a QoE dos utilizadores.
Sugestão	Propor um modelo que com base nos valores dos pedidos, normaliza a heterogeneidade dos diferentes parâmetros de contexto utilizando a normalização *Min-Max,* de forma que a informação recuperada possa ser explorada para prever a prioridade de contexto dos pedidos. As prioridades das tarefas são previstas através da utilização da técnica de MLR. O escalonamento ótimo das tarefas visando a otimização da QoE dos utilizadores, respeitando às suas várias restrições e as dos provedores de serviços é realizado através da utilização da técnica MONLP.
Desenvolvimento	Utilizar a ferramenta de simulação *iFogSim,* para implementar, validar e comparar o desempenho do modelo proposto sensível ao contexto com os escalonadores não sensível ao contexto: *First Come First Served* (FCFS); *Shortest Job First* (SJF) e *QoS Based Priority Scheduling (QoS-based)*.
Avaliação	Avaliar o cumprimento das especificações e determinar o desempenho do escalonador proposto comparando-o com os escalonadores: FCFS, SJF e *QoS-based*, onde os parâmetros de contexto são levados em consideração dentro de um escopo restrito. Esta fase é constituída por várias categorias de testes.

(conclusão)

Etapa	Descrição
Conclusão	Está fase será atingida quando a solução proposta se considerar suficientemente sólida e corresponder na completude aos objetivos propostos.
Contribuição de conhecimento	Será realizada sob a forma de publicações científicas periódicas e sob a forma do presente documento.

Fonte: o autor.

1.6. Estrutura da tese

Além do atual capítulo que apresenta o enquadramento, a motivação, as questões de investigação, os objetivos e contribuições e a metodologia, a presente tese apresenta ainda mais seis capítulos organizados da seguinte forma:

Capítulo 2: Contexto, qualidade de serviço e experiência do utilizador; neste capítulo abordamos conceitos relacionados com o contexto no âmbito da aplicação aos dispositivos móveis, os seus principais desafios, incluindo a definição de sensibilidade ao contexto e o ciclo de vida do contexto, tipos de contextos em computação móvel e plataforma para a ativação de serviços móveis sensível ao contexto. De seguida, são explicitados conceitos como a QoS e QoE dos utilizadores.

Capítulo 3: Cloud computing e tecnologias emergentes; neste capítulo apresentamos conceitos como a *cloud computing* e alguns dos seus paradigmas destacado a *fog computing*, a sua necessidade, benefícios, características e arquitetura referência proposta pelo consorcio *OpenFog*. Por fim, são apresentados as tecnologias e componentes de suporte ao *fog computing* e o simulador *iFogSim*.

Capítulo 4: Algoritmos de escalonamento de tarefas; neste capítulo abordamos o escalonamento de tarefas, identificamos algumas dificuldades na sua conceção, falamos sobre decisões de escalonamento e fizemos uma revisão da literatura sobre os principais algoritmos de escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog* que foram categorizados, considerando as suas características e propriedades, em: básicos; baseados em prioridade e orientado a QoS e sensível ao contexto. Estudámos e discutimos as suas limitações, explorámos e sugerimos algumas perspetivas de melhorias.

Capítulo 5: Modelo e arquitetura proposto; neste capítulo são apresentados os contextos previstos e assunções, ilustramos e descrevemos o nosso modelo e a nossa proposta de arquitetura e, por fim, apresentamos um exemplo ilustrativo que ajuda a visualizar as

funcionalidades do modelo proposto. Três parâmetros de contexto foram considerados: nível de bateria, intensidade do sinal da rede e QoS da aplicação.

Capítulo 6: Avaliação do desempenho; neste capítulo o desempenho da arquitetura proposta é analisado e comparados com os escalonadores FCFS, SJF e *QoS-based*. O desempenho é avaliado em termos de: percentagem de pedido executados com sucesso; tempo médio de espera e QoE dos utilizadores em relação ao aumento dos pedidos, Número de máquinas virtuais (MVs) e QoS.

Capítulo 7: Conclusão; neste capítulo final é apresentada uma síntese dos pontos mais relevantes do trabalho desenvolvido, são tecidas algumas conclusões e evidenciadas as contribuições, com particular destaque para a validação dos objetivos propostos inicialmente. Serão identificadas as limitações do estudo e exploradas potenciais linhas de investigação futuras, de forma a permitir a continuidade do trabalho realizado.

2. CONTEXTO, QUALIDADE DE SERVIÇO E EXPERIÊNCIA DO UTILIZADOR

Neste segundo capítulo definimos o contexto no âmbito da aplicação aos dispositivos móveis, os seus principais desafios, incluindo a definição de sensibilidade ao contexto. Definimos e apresentamos o ciclo de vida do contexto e abordamos conceitos como a QoS e QoE dos utilizadores.

2.1. Contexto

Segundo Sousa (2017), o termo "contexto" é algo que, à partida, pode parecer simples, mas à medida que refletimos sobre o tema, a sua definição torna-se cada vez mais complexa. Avança ainda que o contexto possuiu uma elevada importância, pois depende dele a interpretação e o significado que se atribui à informação. Um bom exemplo disso é a área de Inteligência e Segurança, onde Heuer & CIA (1999), destacam um dos mais conhecidos trabalhos na área de análise de informação de inteligência, que realça a importância da informação como uma reunião entre a natureza da informação e o contexto em que a mesma é interpretada. Os autores acrescentam que o contexto tem importância na obtenção da informação e na forma como esta é utilizada ou interpretada, podendo mesmo alterar a sua importância ou significado.

Nesta secção será apresentada uma revisão das várias definições de contexto nas diferentes áreas científicas.

2.1.1. Definições do contexto

Em computação móvel o contexto de um utilizador é muito dinâmico. Ao utilizar aplicações nesse ambiente, o comportamento dessa aplicação deve ser personalizado para a situação atual do utilizador. Para promover uma utilização efetiva do contexto, muitos autores disponibilizam definições e categorizações de contexto e computação sensível ao contexto. Em Dey *et al.* (1999), é disponibilizada uma definição, as categorias do contexto e aplicações sensível ao contexto. Ainda são apresentadas, informações e serviços, marcação de informações com contexto e metodologia de execução automática de serviços. Bem como, o levantamento do estado da arte relativa à computação sensível ao contexto.

Bazire & Brézillon (2005), examinaram 150 definições de contexto, nas diferentes áreas de investigação e concluíram que a criação de uma única definição é um esforço árduo e provavelmente, impossível visto que a mesma varia com a área científica e depende principalmente do campo em que ela está a ser aplicada. No entanto, avançam que contexto é um conjunto de restrições que influenciam o comportamento relativo a uma determinada tarefa.

A definição de contexto mais utilizada atualmente, mesmo noutros campos, como na vertente da operacionalização foi proferida em (Dey, 2001).

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between user and an application, including the user and applications themselves." (Dey, 2001:45)[2]

Esta definição defende que a ação do utilizador é caracterizada como sendo também ela própria um objeto do contexto, visão partilhada por outros autores (Crowley *et al.* 2002; Cassens & Kofod-Petersen, 2006). Em trabalhos posteriores, como Chen (2003), este estende a definição de contexto às atividades e ações que estão a ocorrer numa determinada localização.

Zimmermann, Lorenz, & Oppermann (2007), partiram da proposta de Dey (2001), e a desconstruíram em cinco elementos: (1) *Individualidade* - propriedades e atributos da entidade, (2) *Atividade* - todas as tarefas que a entidade está envolvida, (3) *Localização* - área ou local onde a entidade está ou pretende ir, (4) *Tempo* - relativa à ação pretendida e (5) *Relações*- informações que a entidade pode estabelecer com outras entidades.

Segundo Sousa (2017), apesar da generalidade da definição, proposta por Dey (2001), ela apresenta algumas lacunas, especialmente ao nível das fronteiras entre alguns conceitos, especialmente entre contexto, modelo contextual e informação contextual.

Henricksen (2003), caracteriza a informação contextual como um mecanismo que possibilita a realização de ações pelo utilizador de forma automatizada e flexível. Avança ainda, que o contexto é considerado em relação as ações entre o utilizador e as aplicações. Assim, propõe a sua interpretação para os seguintes conceitos, contexto, modelo contextual e informação contextual. O contexto é considerado ao nível da ação, e dessa forma propõe que o contexto de uma ação é o conjunto de circunstâncias que a envolvem e que são de potencial relevância para a concluir. Ao nível do modelo contextual, ele é proposto como sendo um subconjunto concreto do contexto, que é realisticamente capaz de ser obtido a partir dos sensores, aplicações e utilizadores, e que pode ser explorado para a execução de uma ação (Henricksen, 2003). Por fim, o mesmo autor define a informação contextual como sendo um conjunto de dados obtidos dos sensores, que pertence ao modelo contextual. Desta forma, o atributo contextual providencia uma amostra, de um determinado ponto no

[2]Tradução livre do autor: "Contexto é qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, um local ou um objeto que possa ser considerado relevante para a interação entre o utilizador e a aplicação incluindo o utilizador e aplicação." (Dey, 2001:45)

tempo, do subconjunto do contexto (Henricksen, 2003). A caracterização contextual ajuda a compreender, utilizar e interpretar os diferentes tipos de contextos de uma forma eficiente.

Sendo o escalonamento de tarefas sensível ao contexto na *fog* e a otimização da QoE dos utilizadores as nossas principais questões de investigação, a nossa definição de contexto centra-se em volta das informações do dispositivo e da aplicação que estão associadas a um pedido (nível da bateria, força do sinal da rede, valores da QoS da aplicação, entre outros). Pois, o escalonamento é efetuado levando em consideração essas informações de contexto. Por este motivo, a definição de contexto assumida no nosso trabalho segue a definição de Dey (2001). Assim, nesta tese, consideramos contexto como sendo:

"Qualquer informação utilizada para representar a ação que uma determinada entidade pretende realizar e as restrições que ela sofre em um determinado momento."

Nesta definição, a *ação* é o que se pretende obter quando a execução é bem-sucedida; a *entidade* pode ser um utilizador ou uma aplicação; as restrições são todas as caraterísticas que impossibilitam ou prejudicam a realização da ação de forma correta, e o *momento* é a duração em que essas restrições se aplicam à ação que se pretende realizar.

2.2. Ciclo de vida de contexto

Conforme Bazire & Brézillon (2005), a utilização de contexto varia de acordo com a arquitetura do projeto e com o autor, mas, de uma forma geral, obedece um ciclo de vida. Avançam ainda, que este ciclo é caracterizado, de uma forma geral, por quatro fases: aquisição; modelação; inferência e distribuição do contexto; descritos nas subseções 2.2.1 a 2.2.4.

2.2.1. Aquisição do contexto

Aquisição do contexto, visa obter o máximo de dados possíveis, de forma a possibilitar que as aplicações, sensível ao contexto, sejam mais eficientes e eficazes devido à qualidade da informação contextual. Assim, a qualidade de contexto (QoC) apresenta-se como um conceito fundamental na aquisição de contexto (Buchholz, Kupper, & Schiffers, 2003), e nesse sentido, a sua importância é apresentada por Bellavista *et. al* (2012), onde definem a QoC como sendo agrupamento de parâmetros que expressam a qualidade dos requisitos e as propriedades que definem o atributo contextual. Avançam ainda, que a QoC se baseia em quatro variáveis: validade; precisão; atualização e o tempo desde a aquisição do atributo contextual.

Conforme Baldauf, Dustdar, & Rosenberg (2007), a aquisição das informações contextuais é geralmente feita através de um conjunto de sensores heterogêneos e não se limita apenas aos dispositivos que recolhem variáveis físicas do ambiente, pode inclusive ocorrer a partir de qualquer fonte de dados capaz de disponibilizar informações contextuais utilizáveis. Acrescentam, que os tipos de sensores podem ser classificados em três grupos:

- Sensores físicos: é o tipo de sensor mais utilizado. São dispositivos que podem captar grandezas físicas diretamente do ambiente. Como exemplos temos: sensores de temperatura, movimento, localização, entre outros;
- Sensores virtuais: nestes as informações proveem de aplicações ou serviços. Por exemplo: identificação da atividade de um utilizador através de uma aplicação que monitora os movimentos do rato ou as entradas do teclado;
- Sensores lógicos: disponibilizam informações contextuais originadas através da combinação de dados que advém de fontes diversificadas, como sensores lógicos, virtuais, físicos ou informações em base de dados. Como exemplo, podemos ter a localização de um utilizador através da utilização de um sensor lógico, utilizando o *Global Positioning System* (GPS) do seu smartphone, combinadas com informações das suas contas armazenadas em base de dados.

Em ambientes IoT, encontramos muitos e heterogêneos dispositivos entre as quais os dispositivos móveis (Ray, 2018). O que obriga que as soluções desenvolvidas para esses cenários possuam de forma intrínseca estratégias que promovam a escalabilidade. Por isso, o melhor momento para recolha das informações contextuais deve considerar a natureza do elemento monitorado para que não se façam pedidos desnecessários. Para apoiar essa decisão, algumas técnicas de recolha são apontadas por Perera *et al.* (2014), as quais são classificadas quanto a responsabilidade e quanto a frequência.

Conforme Sousa (2017), quanto à responsabilidade, as informações de contexto recolhidas podem ocorrer de duas formas:

- *Pull*: nessa situação, o procedimento de recolha da informação contextual é disparado a partir do *middleware*, o qual faz uma consulta diretamente ao sensor. Nesta estratégia de recolha, a aplicação que gere o comportamento do sensor não precisa executar

computações elaboradas, pois não cabe ao mesmo avaliar o momento oportuno de realizar a recolha;

- *Push:* nesse caso o processo de recolha é executado a partir do sensor, que envia os dados recolhidos para o *middleware* mediante um processo de publicação. Esta estratégia proporciona um processo de aquisição de maneira reativa, podendo ser disparada através de eventos do ambiente.

Relativamente a frequência, a aquisição das informações contextuais também pode ser classificada em dois tipos (Sousa, 2017):

- Instantânea: é disparada através de eventos do ambiente e acontece instantaneamente. Como exemplos temos: o acender de uma luz, a abertura de uma porta, uma temperatura atingindo um determinado valor, entre outros. Essa frequência de aquisição associada ao método *push* promove uma reação rápida e é muito conveniente em eventos críticos, onde a decisão é tomada de forma célere;
- Periódica: obedece a períodos temporais especificados conforme as necessidades da aplicação do utilizador. A periocidade pode estar associada a um horário e dia específico, por exemplo, uma informação contextual sendo adquirida de quarta e quinta às 10:00. A aquisição também pode obedecer a intervalos periódicos de tempo, podendo ser adquirida a cada 20 segundos.

As frequências de aquisição instantânea e periódica podem ser implementadas através da utilização tanto do método *pull* quanto do *push*.

2.2.2. Modelação do contexto

Durante a execução da fase de aquisição de dados sensoriais, os dados são obtidos em grandes quantidades e são tipicamente estruturados conforme o formato disponibilizado pelo sensor. De forma a ser possível utilizar os dados, nas diferentes aplicações que deles necessitem, é importante que sejam uniformizados e convertidos num formato adequado para que possam ser armazenados, interpretados e partilhados, entre os vários intervenientes. Existem diversas técnicas que visam dar resposta a esse desafio e, por isso, é comum existirem diversos artigos de revisão que analisam as técnicas mais populares e disseminam as boas práticas já existentes (Moore *et al.*, 2007; Hoareau & Satoh, 2009).

Na modelação de contexto deve-se conceber um modelo de entidades do mundo real, suas propriedades, estado do seu ambiente e situações que podem ser utilizados como referência para a aquisição, interpretação e raciocínio de informações contextuais (Knappmeyer *et al*, 2013). A modelação das informações de contexto diminui a complexidade das aplicações sensível ao contexto, facilita o acesso às informações, permite fazer pesquisas de forma eficiente, melhora a manutenção e evolução da aplicação.

Segundo Perera *et al.* (2014), existem vários modelos para representação do contexto, nomeadamente: chave-valor, linguagem de marcação, gráfico, orientado a objetos, lógico e ontológico. Avançam ainda, que os modelos híbridos de modelação de contexto por combinarem diferentes técnicas de modelagem, com diferentes níveis de interpretação, para diferentes aspetos são considerados muitos promissores.

2.2.3. Inferência do contexto

A inferência de contexto surge devido à natureza imperfeita e incerta dos dados contextuais. O objetivo da inferência é deduzir informação contextual, de alto nível, a partir de dados bruto. A inferência permite também algumas funcionalidades base tais como: validação dos dados; preenchimento de dados em falta; deteção e remoção de anomalias; verificação da consistência; e aquisição de novos dados através de cálculo ou a partir dos existentes (Ameyed, Miraoui, & Tadj, 2015).

2.2.4. Distribuição do contexto

A distribuição está relacionada com a forma como o contexto será entregue aos interessados e disponibiliza métodos para essa entrega. Sistemas utilizam diferentes métodos como:

(i) Consultas, a partir dos quais consumidores executam o seu pedido de contexto;

(ii) Subscrição, através do qual o sistema entrega o contexto aos subscritos em uma abordagem denominada *publisher/subscriber*.

O ambiente computacional deve ser adaptado enquanto a situação ocorre, utilizando valores atuais de sensores, bem como, informações históricas dos mesmos (Ye, Stevenson, & Dobson, 2011).

2.3. Tipos de contextos em computação móvel

Segundo Musumba & Nyongesa (2013), os tipos de contextos podem ser categorizados em termos de mudança no ambiente de execução, de acordo com:

- *Ambiente de computação*, onde são previstos tipos de contextos como: processadores disponíveis, dispositivos acessíveis para entrada e visualização pelo utilizador, capacidade de rede, conectividade e custos de computação;
- *Ambiente do utilizador*, onde se enquadram os diferentes tipos de contextos: localização, recolha de pessoas próximas e situação social;
- *Ambiente físico*, onde se destacam tipos de contextos como: nível de iluminação e ruído.

Os mesmos autores definem os principais contextos que podem ser considerados em qualquer ambiente de computação móvel. Partindo da definição destes autores, fizemos uma adaptação para realçar os tipos de contexto que consideramos ser importantes integrar num ambiente de computação móvel, conforme ilustrado na figura 2.1.

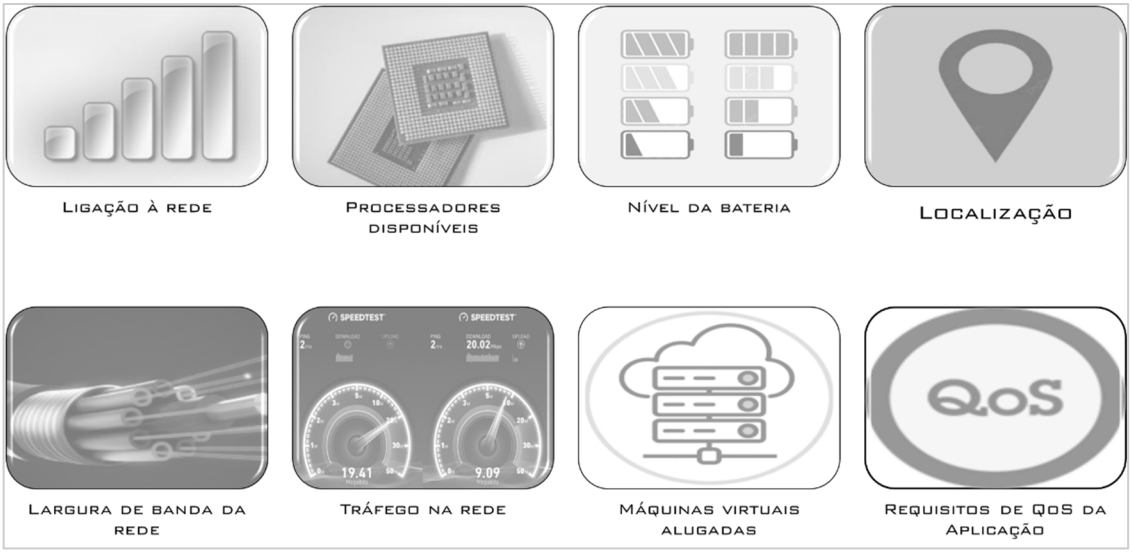


Figura 2.1: Diferentes tipos de contexto em computação móvel
Fonte: adaptada de (Musumba & Nyongesa, 2013).

Avançam ainda, que os tipos de contextos podem ser categorizados em quatros tipos:

- *Contexto do dispositivo do utilizador*, onde se destacam os tipos de contextos: nível da bateria; processadores disponíveis, localização;

- *Contexto da rede*, onde se ressaltam tipos de contextos como: conetividade da rede, largura de banda, tráfico na rede;
- *Contexto de tempo de execução da aplicação*, onde se apontam o tipo de contexto QoS da aplicação;
- *Contexto do provedor de serviços*, onde se sinalizam o tipo de contexto máquinas virtuais alugadas.

No domínio do nosso problema consideramos apenas os contextos do dispositivo do utilizador final, contextos da rede e o contexto de tempo de execução da aplicação. Portanto, foram omitidos o contexto do provedor de serviços (por exemplo, máquinas virtuais alugadas). Também depois de descarregada a tarefa para o nó da *fog*, os processadores disponíveis do dispositivo móvel são desnecessários. A localização do dispositivo móvel também não afetará o escalonamento da tarefa. O tráfego da rede também é igual para todos os utilizadores finais.

2.4. Sensibilidade ao contexto

Segundo Sousa (2017), sensibilidade ao contexto é uma visão da computação centrada no utilizador, na qual, o sistema computacional se adapta dinamicamente e de forma não intrusiva, às necessidades do utilizador no ambiente em que se encontra. Avança ainda, que por este motivo, uma aplicação sensível ao contexto pode apresentar diferentes semânticas, diferentes formatos, diferentes tipos de ações e informação em função do contexto a ser utilizada. Como exemplos de sensibilidade ao contexto podemos descrever o cenário de um dispositivo mudar automaticamente para o modo noturno quando for noite, ou quando um dispositivo em função da pouca bateria disponível muda automaticamente para o modo de economia de energia.

O conceito de sensibilidade ao contexto aplicacional foi apresentado pela primeira vez por Schilit & Theimer (1994), onde referem que, as aplicações são sensível ao contexto quando elas próprias conseguem adaptar ao contexto.

Conforme Pascoe (1998), a sensibilidade ao contexto é a capacidade dos dispositivos de detetar, interpretar e reagir, às alterações do ambiente local do utilizador e dos próprios dispositivos.

Outros autores (*e.g.* Schilit & Theimer, 1994; Dey *et al.*, 1999), apresentam definições semelhantes, que tratam a sensibilidade ao contexto como habilidade de se adaptar ou alterar dinamicamente a sua forma de atuar, de acordo com o estado da aplicação ou do utilizador.

Nas secções 2.1 a 2.4, foram descritos o contexto, a sua importância e o seu ciclo de vida. Também foram identificados os principais tipos de contextos em computação móvel. Surge agora a questão de como fazer a utilização do contexto. Segundo Dey *et al.* (1999), a resposta a essa questão passa pela utilização de aplicações e sistemas sensível ao contexto, que será descrito na secção 2.5.

2.5. Sistemas e serviços móveis sensíveis ao contexto

O contexto inclui toda a situação relevante para uma aplicação e seus utilizadores. Um sistema é sensível ao contexto se o mesmo for utilizado para facultar informações e serviços relevantes ao utilizador, onde a relevância depende da tarefa do utilizador (Dey *et al.*, 1999). Por exemplo, quando um sistema sensível ao contexto detetar que um utilizador não atende chamadas telefónicas enquanto está a conduzir, ele transfere automaticamente as chamadas recebidas para o correio de voz do utilizador. Como outro exemplo, quando prevê a próxima aplicação a ser utilizada e o disponibiliza na janela inicial, ou quando prevê o trajeto a ser feito pelo utilizador e informa o estado do trânsito, ou então quando prevê e sugere os próximos lugares a serem visitados, com base na localização atual do utilizador conforme o *framework* apresentado em Ye, Zhu, & Cheng (2013), onde através de *logins* feitos pelo utilizador na rede social Gowalla (comprada pelo *Facebook* em 2011 e descontinuada em 2012) prevê o próximo lugar a ser visitado pelo utilizador.

Segundo Han (2013), três funcionalidades básicas devem ser implementadas em qualquer aplicação sensível ao contexto:

- *Apresentação de informações e serviços*, refere as funções que apresentam informações de contexto ao utilizador ou utilizam o contexto para propor escolhas apropriadas de ações ao utilizador;
- *Execução automática de serviços*, descreve as funções que desencadeiam um comando ou reconfiguração do sistema em nome do utilizador de acordo com mudanças de contexto;

- *Armazenamento e recuperação de informações de contexto,* aplicações destacam os dados capturados com informações relevantes de contexto.

Por outro lado, atualmente, como a maioria dos *smartphones* possuem vários sensores, as leituras de sensores como GPS, acelerômetro, sensor de luz, microfone, termômetro, relógio, bússola, entre outros, podem ser associadas ao tempo e vinculadas a outras leituras dos *smartphones*. As consultas podem ser feitas sobre esses dados a fim de se agrupar informações de contexto valioso. Um *widget* de contexto é responsável por obter um determinado tipo de informação de contexto e disponibilizá-los as aplicações de forma genérica, independentemente de como elas são realmente detetadas.

Uma plataforma para ativar serviços móveis sensível ao contexto na *cloud* é apresentada por La & Kim (2010). A sua estrutura permite tarefas de: captura de contexto; determinar o ajuste específico de contexto; adaptar serviços candidatos para o contexto e executar o serviço apropriado.

Os autores citados propuseram uma plataforma que permite a integração da *edge network* no ecossistema de computação, e sugeriram um método de provisionamento que facilita a comunicação entre os diferentes componentes.

Uma análise sobre a sensibilidade ao contexto e o tratamento da heterogeneidade em ambientes de computação móvel é apresentada em Schmohl & Baumgarten (2008). Nesta ainda, é descrita uma arquitetura abrangente, que visa resolver os problemas da heterogeneidade presentes nos domínios da sensibilização ao contexto e interoperabilidade.

Uma estrutura de recolha de contexto que introduz os formatos de dados do sensor, conjunto de interfaces e protocolos de mensagens que permite a recolha de contexto dos recursos do sensor é exposta em Devaraju, Hoh, & Hartley (2007).

As plataformas que permitem a ativação de sistemas e serviços móveis sensível ao contexto devem primar pelas qualidades de serviço e de experiência do utilizador que será descrita na secção 2.6.

2.6. Qualidade de serviço e de experiência do utilizador

A qualidade de experiência é a avaliação das perceções e expetativas dos utilizadores na utilização de um serviço. Essa avaliação pode ser afetada por aspetos funcionais, técnicos, humanos, entre outros e podem interferir na experiência do utilizador. Segundo Silva (2017), em âmbitos relacionados com as telecomunicações, a QoE tem sido estudada como uma extensão da QoS, afiança ainda, que muitas vezes, apenas os aspetos relacionados com os parâmetros técnicos do desempenho da rede são utilizados para a definição do nível de satisfação do utilizador em relação a um produto ou serviço. Para uma boa gestão da QoE torna-se necessária uma compreensão profunda e abrangente das múltiplas dimensões da perceção humana da qualidade e dos fatores que as influenciam. Nesta secção, abordaremos os conceitos de QoS e QoE.

2.6.1. Qualidade de Serviço

A definição mais genérica de QoS provém da definição da qualidade proposta pela *International Organization for Standardization* (ISO), ISO 8402:1994, onde a qualidade é definida como:

"The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs" (ISO, 1994:5)[3]

Na recomendação da *International Telecommunication Union* (ITU), ITU-T Rec. E.800, a QoS é definida como:

"Totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service." (ITU, 2008:3)[4]

Em ISO (2015), é apresentada uma definição mais abrangente da QoS que engloba a qualidade na perspetiva de serviços e produtos de uma organização que é determinada:

[3]Tradução livre do autor: "A totalidade de características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explicitas e implícitas." (ISO, 1994:5).
[4]Tradução livre do autor: "A totalidade de características dos serviços de telecomunicações que lhe confere a capacidade de satisfazer as necessidades explicitas e implícitas do utilizador do serviço." (ITU, 2008:3).

"by the ability to satisfy customers and the intended and unintended impact on relevant interested parties." (ISO, 2015:8)[5]

É descrito ainda, que a qualidade dos produtos e serviços não incluem apenas as funcionalidades e os desempenhos previstos, mas também os benefícios percetíveis do seu valor acrescentado para os utilizadores.

Por outro lado, A *European Telecommunications Standards Institute* (ETSI) no seu relatório técnico, define QoS na perspetiva da rede como:

"The ability to segment traffic or differentiate between traffic types in order for the network to treat certain traffic differently from others." (ETISI, 2003:13)[6]

Quando comparadas as definições do ETSI na perspetiva da rede, com as da ITU e ISO, verificamos que as definições da ITU e ISO estão centradas no serviço enquanto entidade em consideração. No entanto, é importante notar que as diferentes definições tendem a refletir pontos de vista sobre diferentes áreas como: redes; sistemas de telecomunicações/TI; e serviços/aplicações na perspetiva do utilizador e da rede.

Nesta tese, é utilizada as definições da QoS proferidas pela ITU e ISO, que derivam da definição da qualidade enunciada pela ISO.

Segundo ITU (2017), tradicionalmente, a QoS aborda principalmente a perspetiva de um utilizador final como uma pessoa (por exemplo, de telefonia), com capacidades de ouvir, ver e ser tolerante com alguma degradação dos serviços. Avança ainda, que com o advento de novas formas de comunicações onde os serviços podem não exigir a entrega em tempo real e o emissor ou o utilizador final pode não ser uma pessoa, podendo ser uma máquina, é importante ter presente que nem todos os serviços são iguais (por exemplo, IoT). E que mesmo os serviços semelhantes podem ter tratamentos diferenciados dependendo por exemplo, se são utilizados por máquinas ou por seres humanos numa ou em ambas as extremidades de uma determinada sessão ou ligação.

[5]Tradução livre do autor: "Pela capacidade de satisfazer os clientes e pelo impacto intencional e não intencional nas partes interessadas relevantes." (ISO, 2015:8).
[6]Tradução livre do autor: "a capacidade de segmentar o tráfego ou diferenciar os tipos de tráfego de modo a que a rede possa tratar um determinado tráfego de uma forma diferente de outros." (ETISI, 2003:13).

A perceção do utilizador final de um serviço de Telecomunicações/Tecnologias de Informação (TI) também é influenciada por diferentes fatores, tais como: tendências sociais, publicidade, tarifas e custos, que estão relacionados com a expetativa do utilizador em relação à QoS.

Em relação à qualidade do serviço, a perceção do utilizador não se limita apenas às características objetivas da interação homem-máquina. Para os utilizadores finais, a qualidade que experimentam individualmente durante a utilização de um produto/serviço também conta (ITU, 2017b).

Segundo Brooks & Hestnes (2010), tal como em outras áreas de investigação psicológica, as boas práticas aceites para a definição da QoS requerem a identificação e medição de comportamentos objetivos, sendo as medidas subjetivas da opinião dos utilizadores aproveitadas principalmente como suplemento.

2.6.2. Qualidade de experiência

Soldani *et al.* (2006), definiram a QoE como:

"…term to describe user perceptions of the performance of a service." (Soldani *et al.,* 2006: xiv)[7].

"How a user perceives the usability of a service when in use - how satisfied he or she is with a service…" (Soldani *et al.*, 2006:3)[8].

De acordo com a recomendação ITU-T P.10/G.100, a QoE foi inicialmente definida como:

"The overall acceptability of an application or service, as perceived subjectively by the end-user." (ITU, 2008:5)[9]

Nestas definições a QoE é analisada num escopo muito restrito, visto que, para a sua descrição é colocado a ênfase apenas nas medidas subjetivas do utilizador final.

[7]Tradução livre do autor: "... termo para descrever as perceções do utilizador sobre o desempenho de um serviço." (Soldani et al., 2006: xiv).
[8]Tradução livre do autor: "Como um utilizador percebe a usabilidade de um serviço quando em utilização - o quão satisfeito está com um serviço..." (Soldani *et.al*, 2006: 3).
[9]Tradução livre do autor: "Aceitação total de uma aplicação ou serviço, tal como é percecionada subjetivamente pelo utilizador final." (ITU, 2008:5).

Em 2019, a recomendação ITU-T P.10/G.100 foi retificada e atualizou-se a definição da QoE formulada em 2008 para:

"The degree of delight or annoyance of the user of an application or service." (ITU, 2019:25)[10]

A mesma recomendação da ITU-T ratificada, associada a QoE, definiu dois novos termos: fatores que influenciam a QoE e a avaliação da QoE.

Os fatores que influenciam a QoE incluem o tipo e características da aplicação ou serviço, o contexto de utilização, as expectativas do utilizador em relação à aplicação ou serviço e ao seu cumprimento, o contexto cultural do utilizador, as questões socioeconómicas, perfis psicológicos, estado emocional do utilizador, entre outros (ITU, 2019).

Realça ainda, que a *avaliação da QoE*, constitui o processo de medir ou estimar a QoE para um conjunto de utilizadores de uma aplicação ou serviço mediante um procedimento específico, e considerando os fatores que a influenciam (possivelmente controlados, medidos, ou simplesmente recolhidos e relatados). E que os resultados do processo pode ser um valor escalar, representação multidimensional dos resultados, e/ou descrições.

Portanto, uma avaliação de QoE limita os fatores que a influenciam, incluindo apenas um ou um pequeno grupo de fatores.

Na nossa perspetiva e atendendo as investigações em curso sobre este tema, trata-se de uma definição funcional, que se completa com a definição dos fatores que a influenciam e a sua avaliação.

Segundo Brooks & Hestnes (2010), QoE de utilizador é:

"A measure of user performance based on objective and subjective psychological measures of using a service or product." (p.12)[11].

Notam ainda que a QoE leva em consideração os parâmetros técnicos (por exemplo, QoS) e a utilização de variáveis de contexto (por exemplo, tarefa de comunicação). Além disso,

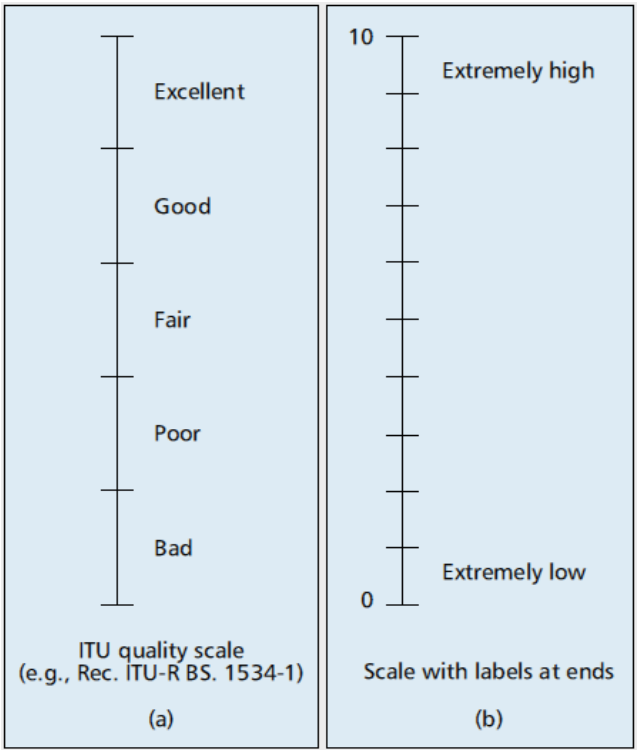
[10]Tradução livre do autor: "Grau de satisfação ou de aborrecimento do utilizador em relação a uma aplicação ou um serviço." (ITU, 2019:25).
[11]Tradução livre do autor: "Uma medida de desempenho do utilizador baseada em medidas psicológicas objetivas e subjetivas da utilização de um serviço ou produto." (Brooks & Hestnes, 2010:12).

mede o processo e os resultados da utilização (por exemplo, eficácia, eficiência, satisfação e prazer do utilizador).

Prosseguem, dizendo que as medidas psicológicas objetivas não dependem da opinião do utilizador (por exemplo, tempo de conclusão da tarefa medido em segundos, precisão da tarefa medida em número de erros). São medidas diretas do processo ou resultado do comportamento do utilizador. Por outro lado, as medidas psicológicas subjetivas são muito influenciadas pela reação do utilizador (por exemplo, o conhecimento da qualidade do meio, a satisfação com um serviço).

Quando recolhemos dados subjetivos dos utilizadores, há uma tendência em os categorizar como qualitativos com escalas de classificação ordinais. Por exemplo, a escala de qualidade *Mean Opinion Score* (MOS), ilustrado na figura 2.2a, é uma escala qualitativa ordinal e não quantitativa. Mesmo que as cinco posições de escala: *Bad, Poor, Fair, Good, and Excellent* forem categorizadas de 1 a 5 conforme ITU (2008), ou de 0 a 100 conforme ITU (2015), os números podem não refletir com precisão a distância entre as posições da escala, pelo que pode inviabilizar o cálculo de estatísticas tais como: a média; o desvio padrão; coeficiente de correlação; entre outros porque estas estatísticas exigem dados intervalares ou de razão. Seria igualmente inválido utilizar algumas técnicas úteis, como: a análise de fatores para agrupamento de dados e *Multiple Linear Regression* para a modelagem preditiva.

Uma escala de classificação subjetiva com propriedades intervalares ou de razão poderá ser obtida através da utilização de uma escala sem rótulos ou com um rótulo em cada extremidade da escala, conforme ilustrado na figura 2.2b.



igura 2.2: Escala para medição de respostas subjetivas
Fonte: (Brooks & Hestnes, 2010).

Segundo Zieliński, Rumsey, & Bech (2008), a escolha da escala depende de alguns fatores como: aspetos do teste; tentativas de diminuição de erros e enviesamentos dos participantes no teste, entre outros. A utilização de escala de classificação subjetiva com propriedades intervalares ou de razão, permite aumentar a gama e validade das técnicas estatísticas que podem ser utilizadas e, combinar de forma mais útil dados subjetivos e objetivos para a definição e otimização de QoE.

Na nossa perspetiva, a utilização da combinação de variáveis objetivas e subjetivas, tal como definido em Brooks & Hestnes (2010), reflete melhor a complexidade da QoE e a medição das variáveis subjetivas e objetivas da experiência do utilizador de forma quantitativa permite que sejam aplicados posteriormente várias técnicas para permitir obter diferentes classificações globais de QoE.

Nesta Tese, a otimização de QoE dos utilizadores foi feita recorrendo a medição das variáveis subjetivas e objetivas de forma quantitativa.

3. *CLOUD COMPUTING* E TECNOLOGIAS EMERGENTES

Neste terceiro capítulo apresentamos uma visão sobre cloud computing e, algumas das suas paradigmas serão explicadas com objetivo de identificar o seu escopo, requisitos e integração. Serão evidenciadas a maturidade dessas tecnologias, os problemas não resolvidos e as motivações para a fog computing. Este capítulo termina com a descrição do simulador iFogSim.

3.1. *Cloud computing*

Segundo Vaquero *et al.* (2008), o termo *cloud computing*, não pode ser visto como um conceito totalmente novo, atendendo que no passado existiram várias abordagens onde a ideologia subjacente era muito semelhante ao que hoje chamamos, *cloud computing*. Na literatura, encontramos diferentes definições, embora todas relativamente parecidas. Os mesmos autores, fizeram uma análise as distintas definições utilizadas na literatura e concluíram que:

> *"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs."*[12]
>
> (Vaquero *et al.*, 2009:51)

Para Taurion (2009), *cloud computing* é um ambiente composto por diferentes servidores físicos ou virtuais e serviços com capacidade de processamento, armazenamento, disponíveis como serviço na Internet. O autor acrescenta ainda, que a utilização da *cloud computing* apresenta vantagens como: Serviços com métrica, elasticidade, agrupamento de recursos, acesso amplo aos recursos da rede, autosserviço baseado na demanda, baixo custo, fácil utilização, disponibilidade, gestão simplificada do ambiente, entre outros. Segundo o mesmo autor, privacidade e segurança por estarem associadas a jurisdição, apesar dos esforços em garantir padrões confiáveis e reconhecidos internacionalmente constituem os maiores desafios.

Segundo Zhang, Cheng, & Boutaba (2010), a arquitetura da *cloud* é dividida em quatro camadas: *Data Center*, infraestruturas, plataformas e aplicações. Esta arquitetura ganhou notoriedade através da disponibilização da infraestrutura *cloud* em três planos de serviço essenciais: *Software as a Service* (SaaS), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS) (Zhang *et al.,* 2010). De acordo com os mesmos autores, podemos ter diferentes tipos de *cloud*: *Private; Community; Public e Hybrid*. Afiançam ainda, que cada

[12]Tradução livre do autor: "*Cloud* é um conjunto de recursos virtuais facilmente usáveis e acessíveis tais como hardware, plataformas de desenvolvimento e serviços. Estes recursos podem ser reconfigurados para ajustarem as necessidades, permitindo assim a otimização da utilização dos recursos. Estes recursos são tipicamente explorados através do modelo *pay-per-use* disponibilizadas pelo provedor através de *Service Level Agreements* (SLAs)."

(Vaquero *et al.*, 2009:51)

tipo possui vantagens e inconveniências, e a opção por um modelo depende do cenário, dos diferentes aspetos do negócio, dos requisitos de utilizadores e ambientes envolvidos.

Na perspetiva de Luan *et al.*(2016), o nosso *networking* é moldado pelas tendências da Internet baseada na *cloud*, sendo que, cerca de 90% dos utilizadores da Internet diretamente ou indiretamente estão a confiar nos serviços disponibilizados pela *cloud* e os acessos são preferencialmente através dos dispositivos móveis.

Referem ainda que, desde 2011 a venda dos dispositivos móveis a nível mundial ultrapassou a dos computadores pessoais, que o processamento na *cloud* se tornou na abordagem global da Internet para armazenamento, processamento, recuperação e gestão de informações e que os dispositivos móveis se tornarem em principais dispositivos para o acesso às aplicações e serviços. Pelo que, a integração entre a *cloud computing* e os dispositivos móveis é promissor. No entanto, enfrenta alguns importantes desafios:

- *Agilidade dos serviços:* ao contrário dos utilizadores de computadores tradicionais, que normalmente solicitam aplicações/serviços de Internet comuns, como *e-mails* e navegação na Internet, os utilizadores dos dispositivos móveis conseguem solicitar aplicações muito diversificadas e que podem adaptar às suas localizações e ambiente, como por exemplo, aplicações *eHealthcare* e da IoT. A *cloud* devido a sua centralidade não consegue gerir, em tempo oportuno, os vários pedidos de serviços de bilhões de utilizadores móveis ao redor do mundo. Sobretudo as que necessitam de baixa latência.

- *Resposta em tempo real:* como os dispositivos móveis são por natureza, limitados em recursos as aplicações móveis normalmente precisam terceirizar suas tarefas de processamento para a *cloud* e esperar por respostas. Os emergentes dispositivos da realidade virtual e aumentada por exemplo, dependeriam ainda mais da *cloud* para suportar o sensoriamento e o processamento de dados em tempo real.

- *Ligação longa:* trocas de dados de alto débito entre a *cloud* e dispositivos móveis é fundamental para suportar as aplicações móveis que necessitam de muitos recursos. No entanto, utilizando a *cloud*, pode ser impraticável, devido a longa ligação entre os utilizadores móveis e a própria *cloud* e tem impacto no aumento do custo.

Para enfrentar esses desafios, foram propostos alguns paradigmas da *cloud computing* que visam permitir uma convergência harmoniosa entre a *cloud* e os dispositivos móveis proporcionando a distribuição de conteúdos e processamentos de dados em tempo real. Estes paradigmas serão elucidados na subsecção 3.1.1.

3.1.1. Tecnologias *cloud*

Não obstante das vantagens da *cloud*, algumas situações não são benéficas à sua aplicação. A *cloud* são demasiadamente centralizadas e o grosso da execução é feito em grandes centros de dados.

Em ambientes onde encontramos dispositivos próximos uns dos outros, não seria eficiente enviar dados para núcleos de processamento distantes e aguardar comandos de um centro remoto para dispositivos atuadores individuais. Nestas situações, demoras e sobrecargas inviabilizam a utilização de soluções que necessitem de baixa latência.

Para Fritsch & Walker (2014), diferentes técnicas que diminuem o volume de dados enviados para *cloud* através do processamento local em elementos periféricos serão fundamentais para diminuir custos e habilidades de respostas das aplicações.

Outros paradigmas decorrentes da *cloud* utilizam diferentes terminologias para representarem conceitos parecidos, onde recursos estão próximos dos utilizadores e permitem resolver limitações como: baixa latência; mobilidade e reconhecimento de localização.

O interesse de pesquisa na *web* ao nível mundial relativo aos paradigmas da *cloud computing* nos últimos cinco anos (2015 a 2020) é ilustrado no gráfico 3.1. Através dela, podemos concluir que os paradigmas: *Edge Computing, Fog Computing e Mobile Cloud Computing*, de forma inconstante, despertaram maior interesse de pesquisa enquanto que as *cloudlets e micro data centers* despertaram menor interesse.

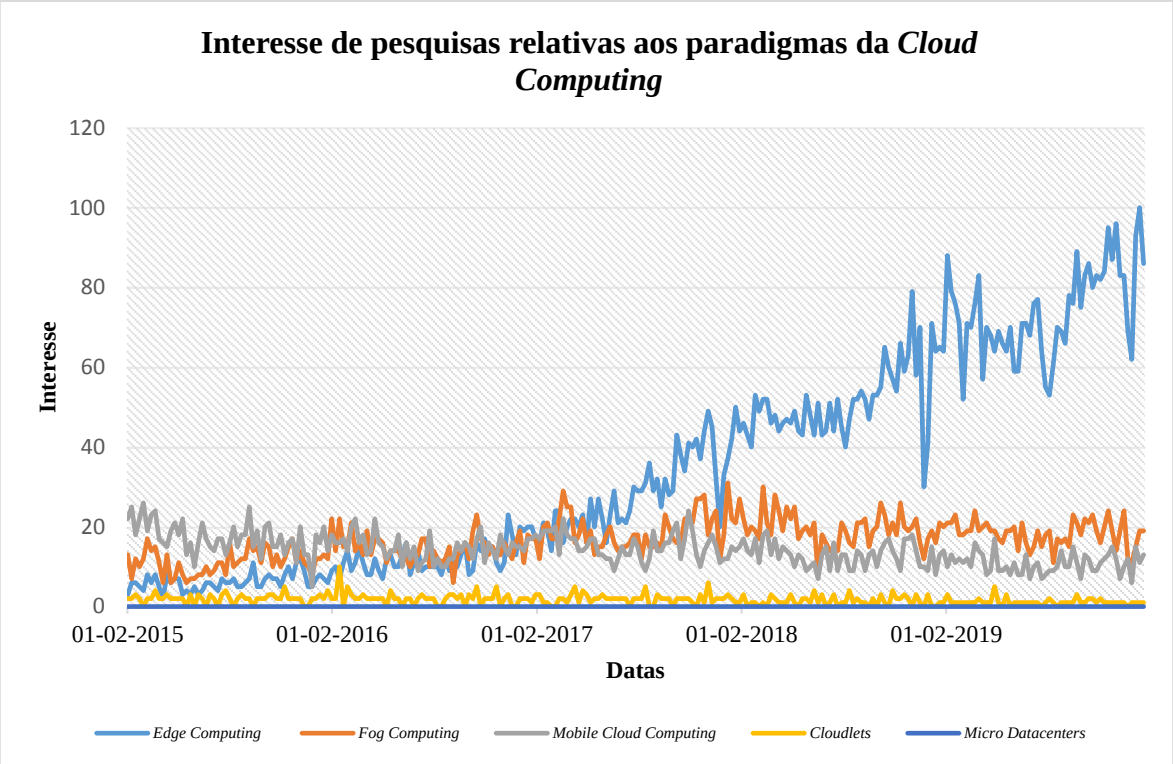


Gráfico 3.1: Paradigmas da *cloud computing*
Fonte: (Google trends, 2020).

Seguidamente descreveremos, de forma resumida, estes paradigmas:

- *Edge computing*, conforme o gráfico 3.1, é o paradigma da *cloud* que despertou maior interesse de pesquisa nos últimos cinco anos. É genericamente utilizado para designar tecnologias na extremidade da rede. No entanto, é utilizado em conceções que nem sempre envolve *cloud* como autenticação em base de dados distribuídos e replicação de *caches*. Embora pouco utilizado, o termo *Edge Cloud Computing*, é mais adequado uma vez que ela ressalta a relação com as tecnologias *cloud*.

- *Cloudlets*, segundo Satyanarayanan *et al.* (2009), é uma camada intermédia entre dispositivos moveis e a *cloud*, foi criado por pesquisadores do *Carnegie Mellon University* e disponibilizado como sistemas de código aberto em *Github*. O conceito das *cloudlets* é utilizado em muitas áreas que não envolvam exclusivamente a *cloud computing*.

- *Mobile Cloud Computing* (MCC), é intendida por Dinh *et al.* (2013), como uma infraestrutura que permite executar pedido de um dispositivo móvel com o auxílio da *cloud*. Possibilita que dispositivos móveis sem capacidade de suportar a execução de uma determinada aplicação, execute essa aplicação. Visto que, ela é executada

remotamente na *cloud* e acedida através da Internet, a interação entre utilizadores e sistema é feita apoiado em interfaces leves ou *web browsers*.

- *Micro Data Centers* (MDC), Microsoft anunciou na NetworkWorld (2015), a sua utilização como prolongamento dos *Data Centers Microsoft Azure* com o objetivo de diminuir custos com tecnologias *cloud*, aperfeiçoar o desempenho dos pequenos dispositivos e aprimorar o desempenho das aplicações da IoT.
- *Fog computing*, segundo Bonomi *et al.* (2014), é uma ampliação da *cloud*. Por ser o paradigma escolhido na implementação desta tese, será descrito em detalhe na secção 3.2.

Segundo Klas (2016), não existe uma clara diferença entre esses paradigmas, visto que, eles convergem para:

- *Pequenos centros de processamentos de dados*, permitem a disponibilização de um ambiente *cloud* minimizada, com suporte a virtualização de recursos e geridos por empresas de telecomunicações e outras em consórcio. Dependendo da arquitetura, podem ser mencionados como: *micro datacenters, cloudlets, fog nodes ou servidores MEC*.
- *Recursos distribuídos em grande escala*, apontam a multiplicação da distribuição de pequenas *cloud* em vários pontos estratégicos a fim de, disponibilizar recursos computacionais aos clientes. A sua instalação e localização dependerá de fatores como economia de energia ou requisitos de latência das aplicações, distribuídas em uma arquitetura com no mínimo três camadas.
- *Infraestrutura e plataforma como serviço*, oferecem uma infraestrutura integrada em pequenas *clouds* seguindo os modelos de serviços semelhante aos modelos *Infrastructure as a Service* (IaaS) e *Platform as a Service* (PaaS) da *cloud*. Nesta plataforma, aplicações podem ser executadas de forma ágil utilizando MVs. As questões de mobilidade dos clientes podem ser tratadas através do modelo PaaS, visto que ela, favorece a migração de MVs em tempo de execução e perante pedidos entre pequenas *cloud*.
- *Serviços especiais*, oferecem serviços exclusivos em algumas pequenas *cloud* utilizando apenas informações da rede local ou nas suas proximidades. Por exemplo,

uma pequena *cloud* pode alojar um serviço que disponibiliza acesso as estimativas do tráfego rodoviário de uma determinada zona. Estas informações podem ser utilizadas pelas aplicações em tempo real para ajustar as rotas disponibilizadas aos utilizadores.

Conforme a gráfico 3.1, o interesse de pesquisa na *web* ao nível mundial relativo a *fog computing* nos últimos anos tem crescido sobretudo desde a sua adoção no final de 2012 pela *Cisco Systems*.

3.2. *Fog computing*

Segundo Stojmenovic (2014), *fog* é um paradigma que visa suprir as limitações da *cloud* através da disponibilização de serviços na extremidade da rede. Em Vaquero & Rodero-Merino (2014), é definida de forma ampla e são dadas ênfase a algumas características como distribuição geográfica, predominância de acesso *wireless*, heterogeneidade, ambientes distribuídos, entre outros.

Segundo Bonomi *et al.* (2014), a *fog* tem crescido muito desde a sua adoção em finais de 2012 pela *Cisco Systems*. Ela desaponta como uma solução integrada para estender os recursos da *cloud* para a extremidade da rede e permitir respostas as inconveniências do modelo clássico centralizado.

Conforme OpenFog (2017), a *fog computing* é:

"A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum." (OpenFog, 2017:1)[13]

Para Gupta *et al.* (2016), ela é uma extensão não trivial da *cloud*, porque disponibiliza serviços de processamento, de armazenamento e de rede próximos à *edge network* empresarial. As suas principais características são a proximidade aos utilizadores finais, sua densa distribuição geográfica e seu suporte a mobilidade.

Conforme Luan *et al.* (2016), a *fog computing* na perspetiva da computação móvel, visa disponibilizar uma instalação semelhante a *cloud*. No entanto, mais leve, mais próximo dos

[13]Tradução livre do autor: uma arquitetura horizontal a nível do sistema que distribui funções de processamento, armazenamento, controlo e ligação à rede mais próxima dos utilizadores através de ligações contínuas. (OpenFog, 2017:1)

utilizadores dos dispositivos móveis, logo, pode servir estes utilizadores através de conexão direta, mais curta, em comparação com à ligação com a *cloud*. Salientam ainda, que como a *fog* pode ser implementada localmente, consegue disponibilizar serviços personalizados e comprometidos com a localização, que são mais desejáveis para os utilizadores móveis.

Nesta tese, assumimos que o conceito de *fog computing* converge em termos de abordagem as definições descritas nesta secção. Isto é, combina o processamento e armazenamento de dados na *edge network* e tem a capacidade de, quando necessário, procurar recursos computacionais adicionais na *cloud*. Permite a disponibilização de serviços que possibilitam maior distribuição geográfica do sistema ou aplicação e baixa latência.

A figura 3.1, ilustra o ambiente de execução de aplicações móveis no paradigma *fog*, composto por vários dispositivos em diferentes localizações, diversos servidores *fog* na *edge network* que visam responder os pedidos dos dispositivos e servidores *cloud* centralizados que são acionados caso os servidores *fog*, por insuficiência de recursos, não forem capazes de executar o pedido.

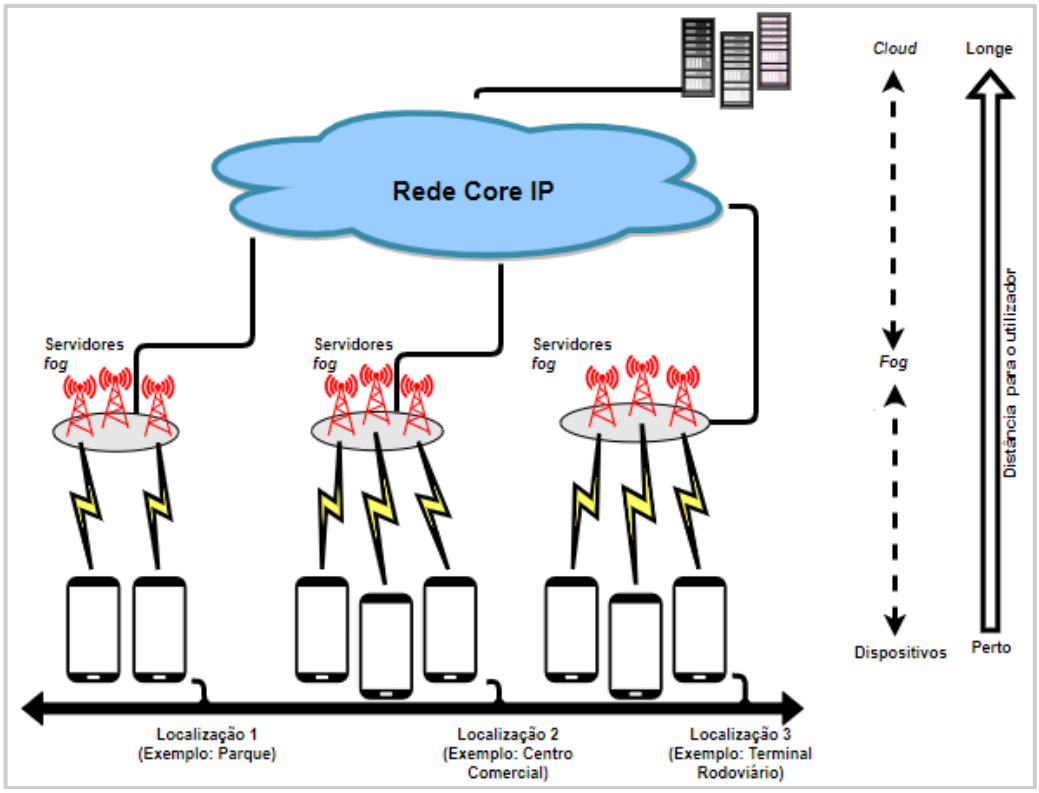


Figura 3.1: Ambiente de execução de aplicações móveis no paradigma *fog*
Fonte: o autor.

3.2.1. Premência da *fog computing*

Apesar das inúmeras vantagens da arquitetura *cloud* entre outros destacamos: menor risco de perdas de dados; maior segurança; maior capacidade de integração; menor consumo de energia devido aos recursos partilhados e otimizados o que traz vantagens ambientais e a diminuição da pegada do carbono. Ela possui inconveniências que não podem ser ignorados. A centralização de dados e dos diferentes recursos de processamento, armazenamento, entre outros, em grandes centros de dados, motiva pedidos de clientes aos servidores centralizados para a execução das tarefas. Isto é, os clientes que pretenderem aceder aos dados e/ou aos outros recursos, precisam primeiramente de endereçar esses pedidos aos servidores remotos, a fim de serem processados para depois, serem devolvidos as respostas, o que origina um aumento de latência na execução dos pedidos.

Quando dados específicos são necessários, o cliente precisa aguardar a disponibilização do acesso pelo servidor remoto. Além disso, os recursos do servidor são partilhados por vários clientes, o que pode propiciar a indisponibilidade de recursos o que impacta negativamente na eficiência do sistema. Ainda, a *cloud* requer uma boa largura de banda, por forma a possibilitar que os dados sejam transferidos do cliente ao servidor e depois fazer o percurso inverso. Isto é, devolvidos do servidor para o cliente. Este processo é repetido para cada pedido de um cliente.

Por outro lado, como o processamento é feito em um computador remoto, acrescem as preocupações relacionadas com a privacidade e a segurança. Ainda, a execução de algumas aplicações na *cloud*, tende a ser ineficiente. Como solução, surgiu o conceito da *fog computing*, pelo fato dos requisitos da informação e processamento estarem a aumentar exponencialmente principalmente na *edge network* impulsionado pelo IoT. Segundo Bonomi *et al.* (2014), servidores e *data centers* centralizados deixam de ser adequados quando consideramos dispositivos da IoT.

A *fog* é um dos paradigmas da *cloud* e visa proporcionar processamento fácil, rápida e com baixa latência. Ela utiliza os dispositivos na *edge network* para executar a maioria do processamento, armazenamento e comunicação. Pequenos servidores chamados servidores *fog* estão próximos dos *edge devices*. Estes servidores *fog*, possuem capacidades de armazenamento de dados e informações, habilidades computacionais, recursos para

responder aos pedidos e comunicação entre vários clientes e outros servidores *fog* (Barros *et al.*, 2019).

Em novembro de 2015, um conjunto de empresas, indústrias, utilizadores finais, consumidores de tecnologias e academias reuniram para criarem o consórcio *OpenFog*, cujo objetivo é padronizar, incentivar e promover a adoção da *fog computing* em diversas áreas e campos, conforme descrito na secção 3.3. Desde então, várias pesquisas vêm sendo feitas por diferentes investigadores, a fim de promover e aprimorar essa tecnologia para que ela possa resolver com eficiência, entre outros os problemas: da latência, do reconhecimento da localização e da escassez de recursos dos dispositivos.

Cada servidor *fog* possui dados armazenados sobre elas e são capazes de responder aos pedidos dos clientes. Genericamente os clientes enviam pedidos ao servidor *fog* mais próximo; se o servidor *fog* correspondente for capaz de responder ao pedido, ele disponibiliza os recursos necessários para a execução da tarefa, efetua cálculos sobre os dados e envia os resultados aos clientes, garantindo assim, uma menor latência. Os servidores *fog* estão mais próximos dos dispositivos e permitem entre outros: baixa latência, aproximar o processamento e o armazenamento dos utilizadores finais, mais segurança.

Também, a *fog computing*, permite resolver o problema da falta de consciência de localização da *cloud* tornando assim, numa mais valia aos utilizadores dos dispositivos móveis.

Segundo Swaroop (2019), ao contrário dos utilizadores dos computadores pessoais, os dos dispositivos móveis têm exigências de serviço previsíveis, sujeitas à sua localização. Apresenta como exemplo, o fato de um utilizador móvel num centro comercial ter a tendência de estar interessado nos locais de vendas, horário de funcionamento, restaurantes e eventos do centro comercial frequentado. Avança ainda, que tais informações se tornam inúteis quando esse utilizador sair desse centro comercial. Como outro exemplo, apontou o fato de um visitante de uma nova cidade geralmente procurar informações sobre os lugares de interesse, notícias e condições meteorológicas daquela cidade, sendo improvável que esteja interessado em tais informações de outros lugares.

A *cloud* permite disponibilizar informações num portal central, não possui a consciência da localização. Esse modelo é adequado para utilizadores de computadores pessoais, conectados através de redes “cabeadas” com grandes débitos. No entanto, além de ser

dispendioso para utilizadores móveis que utilizam uma largura de banda móvel que é mais cara, é também inconveniente, uma vez que seria mais difícil pesquisar em um conjunto global, as informações locais e específicas.

3.2.2. Benefícios da *fog computing*

Segundo Bonomi *et al.* (2012), a *fog Computing* por ser uma plataforma virtualizada que permite serviços de processamento, comunicação e armazenamento entre dispositivos localizados normalmente, na extremidade da rede e na *cloud*. Proporciona alguns benefícios que são extensíveis aos utilizadores dos dispositivos móveis como:

- *Controlo de privacidade e segurança de dados*, a *fog* permite aos utilizadores armazenar, gerir e processar os dados localmente, mais próximos dos *edge devices*, sem a necessidade de enviá-los para servidores centralizados e distantes. Dessa forma, os dados não são partilhados por um grande número de utilizadores e sua confidencialidade pode ser assim preservada. Além disso, podemos monitorar, rastrear e controlar qualquer dispositivo que recolhe ou armazena dados, uma vez que estão a ser guardados localmente.

  Por outro lado, a segurança de dados, constitui uma das principais preocupações decorrentes da crescente utilização da Internet e da partilha de informações. Mais do que nunca, é necessário manter a privacidade e a confidencialidade dos dados. Várias ameaças e ataques tem surgido. A *fog* possibilita a ligação e a partilha de informações entre vários sistemas conectados através da rede. No entanto, em vez das informações estarem concertadas num único ponto, que é mais vulnerável a ataques cibernéticos e roubo de informações, ela é distribuída tornando assim, menos vulneráveis a ataques, possibilitando que ameaças sejam identificadas nos níveis mais baixos e não permitindo a sua penetração nos níveis mais altos da rede, impossibilitando a contaminação dos outros dispositivos.
- *Maior produtividade e agilidade nos negócios*, a *fog* permite entre outros, processamento e armazenamento na *edge network*. Isto é, próximo do lugar onde os dados são efetivamente produzidos. Esta característica impacta na velocidade dos processos de negócios.
- *Baixa latência e reconhecimento da localização*, a *fog* possibilita a redução da latência na partilha de informações, visto que, os dados são processados próximos

dos *edge devices*. O tempo dos pedidos e respostas são consideravelmente menores. Além disso, o cliente no paradigma *fog* consegue reconhecer a localização do servidor em que seus dados estão a ser processados o que permite melhoria na qualidade do serviço em aplicações com baixo requisitos de latência.

3.2.3. Características da *fog computing*

A *fog* permite a execução de aplicações de forma distribuída em camadas entre os dispositivos e a *cloud*. Elementos como *gateways* inteligentes, *routers* e dispositivos *fog* dedicados disponibilizam processamento e armazenamento a fim de permitir a ampliação dos serviços da *cloud* à *edge network*. Segundo Bonomi *et al.* (2012), a *fog* possui as seguintes características que a tornam numa expansão da *cloud* e propiciar a execução de aplicações móveis:

- *Reconhecimento da localização e reduzida latência*, o surgimento da *fog* pode ser amputado às propostas que envolvem acessos com suporte a serviços sofisticados no limite da rede, incluem aplicações com exigências de baixas latências como sejam: *streaming*, jogos, realidade virtual e aumentada, entre outros.
- *Distribuição geográfica*, contrastando com o processamento mais centralizado da *cloud*, os serviços e aplicações *fog* implicam instalações muito distribuídas.
- *Suporte a redes sensoriais em grande escala*, permite a monitorização do ambiente através das redes de sensores, o controlo de tráfego entre viaturas conectados, as redes de energia inteligentes, entre outros.
- *Grande quantidade de nós*, é uma consequência da distribuição geográfica evidenciadas principalmente pelas tecnologias de redes sensoriais, e pela aplicação *Smart Grid* em particular, onde serviços são processados pelos nós da *fog* como componente da aplicação em *cloud* distribuída.
- *Suporte a computação móvel*, no paradigma *fog* é importante uma comunicação direta, ela dispõe de recursos associados a fim de suportar várias técnicas utilizadas em computação móvel.
- *Interações em tempo real*, aplicações no paradigma *fog* requerem de interações em *real time*, em vez do processamento em serie.

- *Predomínio de acesso wireless*, e computação móvel, protocolos de comunicação *wireless* como *RFID, Bluetooth, ZigBee, Wi-Fi* ou *LTE* constituem as principais formas da conexão a rede. Os nós da *fog* conseguem disponibilizar serviços que só podem ser necessários no contexto da computação móvel.
- *Heterogeneidade*, a *fog* pela sua abrangência e escopo possibilita que dispositivos possam ser disponibilizados por diferentes fabricantes, utilizar diferentes ambientes e abarcar várias linguagens de programação.
- *Interoperabilidade e/ou federação*, o integrado suporte aos serviços necessitam de colaboração de vários provedores.
- *Análise de dados em real time*, o fato da *fog* interagir com a *cloud* e ser implementada próxima das origens de dados, faz com que ela esteja bem localizada para desempenhar um importante papel no consumo e processamento de dados em *Big Data* com restrições temporais.

O desenvolvimento harmonioso da *fog*, demanda a existência de organizações com objetivos de a padronizar e a promover em diversas áreas e campos. Assim, em novembro de 2015, foi criado o primeiro consórcio com esses objetivos, a *OpenFog*, conforme descrito na secção 3.3.

3.3. Consórcio *OpenFog*

O consórcio *OpenFog* foi criado em 2015 e a sua composição envolveu membros de empresas, indústrias, utilizadores finais e academias, baseou na premissa de que uma arquitetura aberta é fundamental para o sucesso do ecossistema *fog* ubíquo para plataformas, aplicações, computação móvel, entre outros. Consiste na arquitetura de referência *OpenFog* que foi projetada como uma extensão da *cloud* composta por um conjunto de camadas cujo objetivo primordial é entre outros:

- Propiciar a disponibilização de aplicações interoperáveis, seguras, escaláveis, confiáveis, disponíveis e ágeis, com suporte a diferentes clientes ou dispositivos na extremidade da rede;
- Funcionar, simultaneamente ou não, com serviços *cloud* a fim de, possibilitar o armazenamento, processamento, comunicação e tarefas de gestão otimizadas através de requisitos de volume de tarefas.

Segundo OpenFog (2017), a arquitetura de referência *OpenFog* possibilita interfaces *fog-cloud* e *fog-fog* e apresentam vantagens singulares, que a denominaram *SCALE*:

- *Security;* segurança adicional com o objetivo de garantir transações seguras e confiáveis;
- *Cognition;* consciência dos objetivos centralizados no cliente a fim de permitir a autonomia;
- *Agility;* inovação rápida e escalabilidade acessível sob uma infraestrutura comum;
- *Latency*; processamento em tempo real e controlo dos sistemas ciberfísicos;
- *Efficiency*; agrupamento dinâmico de recursos locais não utilizados pelos dispositivos dos utilizadores finais intervenientes.

Avança ainda, que a arquitetura de referência *OpenFog* foi impulsionada por um conjunto de princípios fundamentais, chamados pilares. Esses pilares, constituem a convicção, abordagem e intenção que levaram a definição dessa arquitetura e constituem os pontos-chave que um sistema necessita para incorporar a definição da arquitetura *OpenFog* horizontal ao nível do sistema que proporciona a distribuição de funções de processamento, armazenamento, controlo e redes mais próximas da origem dos dados.

A figura 3.2, ilustra os oito pilares da arquitetura *OpenFog:* segurança; escalabilidade; abertura; autonomia; fiabilidade, disponibilidade e Manutenibilidade (RAS); agilidade; hierarquia e programabilidade.

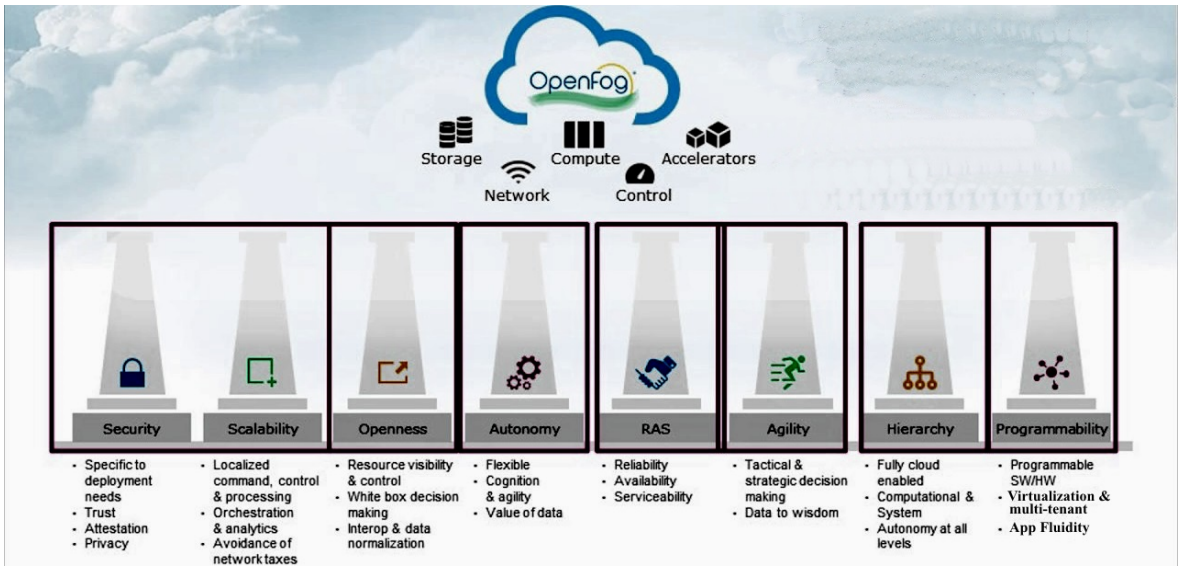


Figura 3.2: Pilares da arquitetura OpenFog.
Fonte: (OpenFog, 2017)

A utilização correta de cada pilar como base para plataforma é chave para uma implementação bem-sucedida. A descrição de cada um dos pilares se encontra disponível no Anexo A desta tese.

3.3.1. Arquitetura de referência para a *fog computing*

Fog computing permite execuções de aplicações em equipamentos distribuídos em diferentes camadas localizados entre os dipositivos finais e a *cloud*. Visando padronizar e normalizar o desenvolvimento da *fog*, o consórcio *OpenFog*, baseada nos oito pilares descritas no Anexo A desta tese e utilizando a norma internacional ISO/IEC/IEEE 42010:2011 como diretriz, propuseram a arquitetura de referência do *OpenFog*.

A arquitetura proposta visa descrever os requisitos/perspetivas de cada parte interessada (fabricante de *hardwares*, desenvolvedores de sistemas, integradores de sistemas, consumidores de tecnologias, entre outros) no processo contínuo da *fog*. Também ajudará a harmonizar as várias tecnologias da *edge computing* que compõem os paradigmas da *cloud*, descritos na subsecção 3.1.1, que embora díspares e potencialmente divergentes, com um trabalho singular podem encontrar uma linha de base comum e trabalhar para satisfazer o desejo de um ecossistema *fog* interoperável.

A figura 3.3, ilustra a arquitetura de referência da *fog computing* proposta pela *OpenFog* com as respetivas perspetivas, mostradas em barras verticais cinzentas nos lados da descrição arquitetónica.

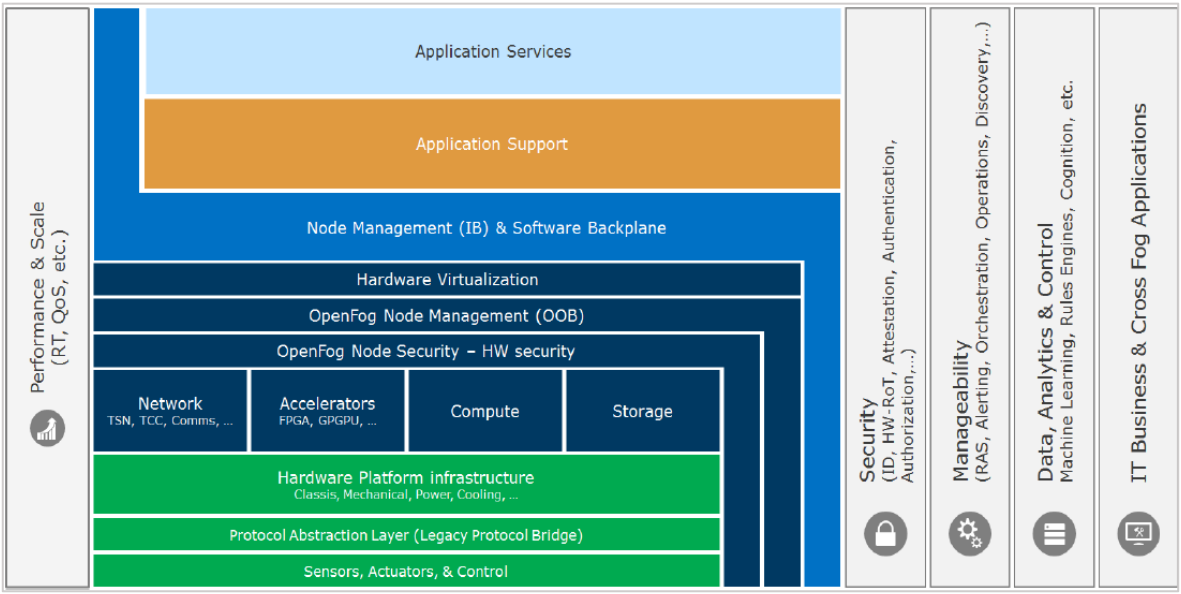


Figura 3.3: Arquitetura de referência da *fog computing* com respetivas perspetivas
Fonte: (OpenFog, 2017).

As perspetivas incluem:

- *Performance;* a baixa latência é uma das razões para a adoção do paradigma *fog*. Existem vários requisitos e considerações de *design* entre as várias partes interessadas para garantir a baixa latência. Por ter impactos no sistema e nos cenários de implementações faz com que ela seja uma preocupação transversal.
- *Segurança;* a segurança de ponta-a-ponta é fundamental para o sucesso de todos os cenários de implementação da *fog computing*. Se o hardware for seguro, mas o software da camada superior ter problemas de segurança (e vice-versa), a solução no seu todo deixa de ser segura.
- *Capacidade de gestão;* a gestão de todos os aspetos de implementações da *fog*, onde incluem *RAS, DevOps*, entre outros, é um aspeto crítico em todas as camadas de hierarquia da *fog computing*.
- *Análise e controlo de dados;* a capacidade dos nós da *fog* serem autónomos requer uma análise de dados localizada associada a um controlo. A atuação/controlo precisa acontecer na camada ou local correto na hierarquia, conforme o cenário definido.
- *Negócios TI que cruzam aplicações fog*; em um cenário composto por vários fornecedores, as aplicações do ecossistema precisam ter a capacidade de migrar e operar adequadamente em qualquer nível da hierarquia de uma implementação *fog*. Aplicações devem também ter a capacidade de abranger todos os níveis de uma implementação para maximizar o seu valor.

Segundo OpenFog (2017), a arquitetura de referência *OpenFog* apresenta um composto de perspetivas e múltiplas vistas das partes interessadas utilizadas para satisfazer uma determinada implementação ou cenário da *fog computing*. As três vistas incluem *Software*, Sistema e Nó:

- Vista do *software*, é representada pelas três primeiras camadas da figura 3.3, que inclui: *Application Services, Application Support, and Node Management* (IB) e *Software Backplane*.
- Vista do sistema, é representada pelas camadas intermédias ilustrada na figura 3.3, inclui *Hardware Virtualization* através do *Hardware Platform Infrastructure*.

- Vista do nó, é representada pelas duas últimas camadas ilustradas na figura 3.3, inclui a Camada *Protocol Abstraction Layer e Sensors, Actuators, and Control*.

Com base na arquitetura referência proposta pelo consórcio *OpenFog*, vários autores como: Sarkar, Chatterjee, & Misra (2018), Dastjerdi *et al.* (2016) e Gubbi *et al.* (2013), propuseram arquiteturas para a *fog computing*.

A figura 3.4 ilustra as propostas dos autores Sarkar, Chatterjee, & Misra (2018) e Dastjerdi *et al.* (2016), onde na parte inferior, estão disponibilizados objetos inteligentes, dispositivos da extremidade da rede e *gateways*. Esta camada possui aplicações que podem ser instalados nos dispositivos finais com objetivo de acrescentar a sua funcionalidade. Os dispositivos nesta camada podem utilizar a camada imediatamente acima para a comunicação com outros dispositivos e/ou com a *cloud*.

A camada de rede auxilia a conexão dos dispositivos, através de tecnologias de rede com ou sem fio. Além disso, providencia também, o acesso a recursos de rede virtualizados como instâncias da *fog* através de elementos inteligentes, capazes de processar e armazenar temporariamente dados recolhidos pelos *gateways* sobre dispositivos da camada inferior. Estes dispositivos da *fog* também são responsáveis por filtrar e enviar/atualizar informações periodicamente para a *cloud*.

Acima da camada de rede são processados serviços que disponibilizam suporte a execução de tarefas para aplicações que precisam de recursos "ilimitados virtualmente" disponibilizados pela *cloud*.

Acima da camada *cloud* encontra-se a camada gestão de recursos definida por software que coordena toda a infraestrutura e oferece qualidade de serviço às aplicações *fog*. Ela implementa diferentes serviços *middleware* com objetivo de otimizar a utilização dos recursos da *cloud* e *fog* em benefício das aplicações.

Finalmente, na camada superior estão as aplicações que utilizam a infraestrutura *fog* para disponibilizar soluções inovadoras e inteligentes aos utilizadores finais.

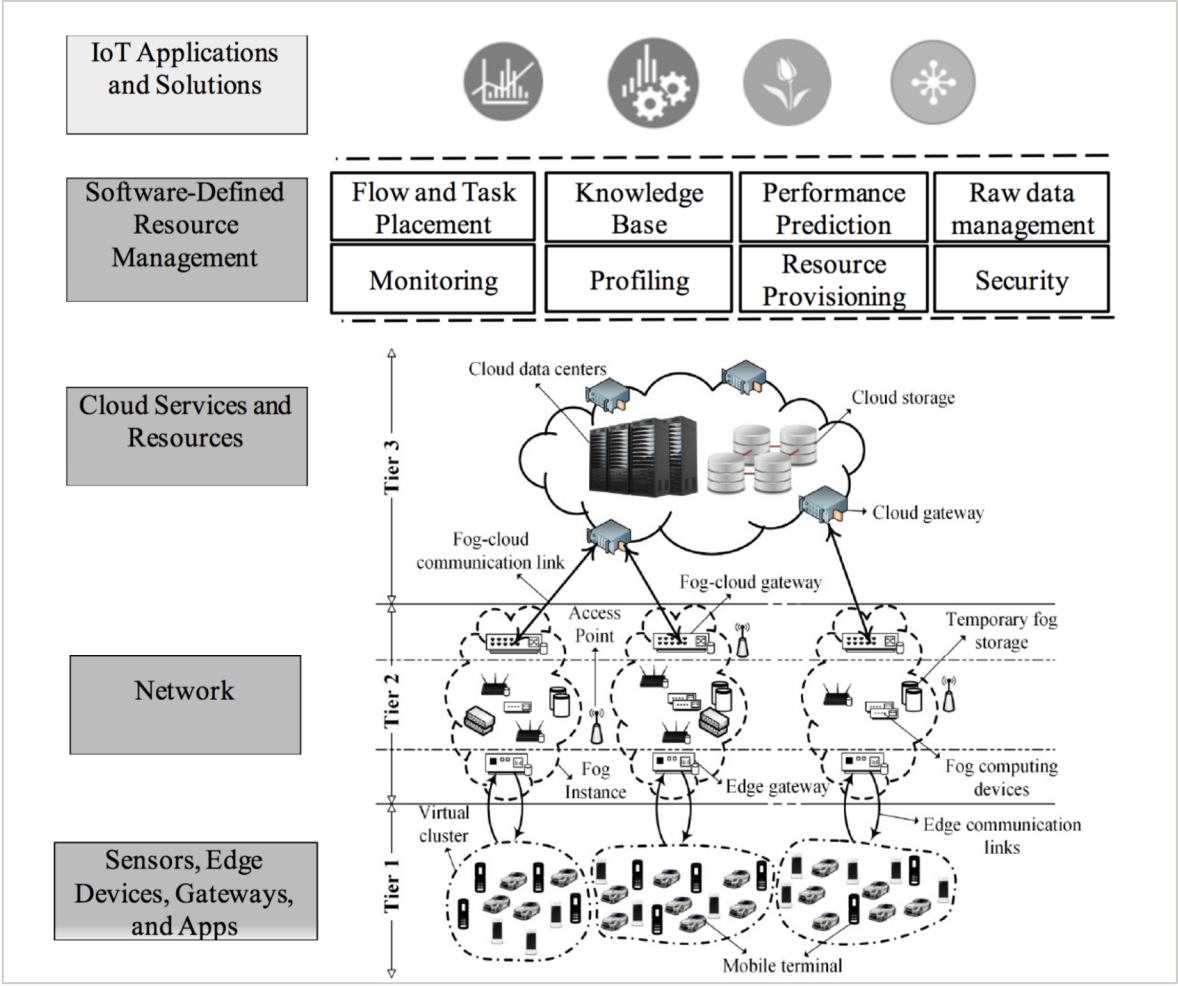


Figura 3.4: Arquitetura padrão para *fog computing*.
Fonte: (Sarkar, Chatterjee, & Misra, 2018) e (Dastjerdi *et al.*, 2016).

Segundo Dastjerdi *et al.* (2016), na camada gestão de recursos definida por software é executado um conjunto de serviços cujo objetivo é reduzir a utilização da *cloud* por forma a melhorar o desempenho das aplicações e reduzir a latência através da deslocação da execução de tarefas para nós da *fog*.

Os autores citados realçam ainda, que esse complexo processo é executado através de um trabalho conjunto que envolve serviços de:

- Localização de fluxo e tarefas, que preserva informações do estado dos recursos disponíveis na *cloud, fog* e na rede através de consultas ao serviço de monitorização. Visa identificar os melhores candidatos a receber tarefas e suportar os fluxos de execução. Este componente está em constante comunicação com o serviço de provisionamento de recursos a fim de indicar o número atual de fluxos e tarefas atribuídas.

- Base de dados de conhecimento, armazena o histórico dos pedidos de aplicações e recursos, que podem ser disponibilizados a outros serviços a fim de auxiliar o processo de tomada de decisão.
- Previsão de desempenho, estima o desempenho dos recursos disponíveis, com base na consulta de informações na base de conhecimento. Esta informação é utilizada pelo serviço de provisionamento de recursos por forma a decidir sobre a quantidade de recursos que serão disponibilizados a quando da sobrecarga de tarefas e fluxos alocados, ou quando o desempenho não é satisfatório.
- Gestão de dados, disponibiliza vistas sobre os dados para outros serviços através do acesso direto às fontes e ficheiros históricos. Essas vistas são obtidas através de consultas *Structured Query Language* (SQL) simples ou através de processamento mais complexos que envolvem técnicas de *Big Data* como *MapReduce*. No entanto, o método específico utilizado na criação das vistas é abstraído para os outros serviços.
- Monitorização, retém o controlo do desempenho e do estado das aplicações, recursos e outros serviços e disponibilizam essas informações quando solicitadas.
- Gestor de perfis, estabelece perfis de recursos e aplicações com base em informações obtidas a partir da base de conhecimento e monitorização de serviços.
- Provisionamento de recursos, é o responsável pela obtenção de recursos na *cloud*, *fog* e na rede a fim de alojar as aplicações. Esta afetação é dinâmica, uma vez que a quantidade de aplicações hospedadas e seus requisitos mudam ao longo do tempo. A decisão sobre os recursos é feita geralmente com base nos requisitos de latência, nas credenciais geridas pelo serviço de segurança, e nas informações disponibilizadas por outros serviços como gestor de perfis, previsão de desempenho e monitorização. Por exemplo, o componente transfere as tarefas com requisitos de baixa latência para a *edge network* logo que os recursos apropriados ficam disponíveis.
- Segurança, disponibiliza serviços de autenticação, autorização e criptografia para acesso e execução de serviços e aplicações, estendido a todos os elementos da *fog*.

Os elementos e serviços descritos são apenas referências. Os protocolos e aplicações *fog* podem ser criados sem a necessidade de todos esses elementos, ou podem integrar outros componentes e serviços não apresentados na figura 3.4.

3.3.2. Tecnologias e componentes de suporte a *fog computing*

Segundo Bonomi *et al.* (2012), plataformas *fog* aproveitam a sua proximidade às origens dos dados para suportar características como mobilidade, baixa latência, entre outros. Suas arquiteturas incluem requisitos da arquitetura *cloud* como adaptação, visualização, escalabilidade, multi-inquilino, entre outros. Vaquero & Rodero-Merino (2014), destacam algumas tecnologias que têm evoluído e ajudado na resolução da complexidade envolvida:

- Técnicas de rede definida por software (*network softwarization*), *fog* abrange infraestruturas, dispositivos que são geridas por diferentes organizações, há a necessidade dos serviços serem executados em conformidade ou automatizadas por software. Utilização de técnicas de *Network Functions Virtualization* (NFV) pelas operadoras podem disponibilizar implementação dinâmica de serviços sob demanda. As tecnologias *Software Defined Networking* (SDN) quando utilizadas possibilitam que a rede seja gerida por softwares, como resultado as operações serão mais ágeis e baratas em relação as soluções tradicionais que se baseiam em hardware.
- Técnicas declarativas e assintóticas, orientada a gestão em grande escala, estas soluções permitem especificação do estado desejado para o sistema de forma declarativa ao invés de serem utilizados comandos de configuração individuais. Entretanto, é assumido que o estado final atingido pode não ser alcançado. Dado que, o sistema pode ser modificado durante as configurações.
- Nós da *fog*, são grupos de dispositivos e/ou dispositivos *fog* que podem comportar como uma *cloud* em pequena escala.
- Abordagens baseadas em redes *Peer-to-Peer* (P2P) e de sensores, permitem cooperação entre nós permitindo maior escalabilidade, com resultados semelhantes as técnicas que exigem um único provedor responsável pela performance da rede e dos serviços.

Segundo Yi *et al.* (2015), a plataforma *fog* é constituído pelos componentes conforme a figura 3.5, alguns componentes são iguais aos da arquitetura de referência, ilustrado na figura 3.3 e descrito na subsecção 3.3.1.

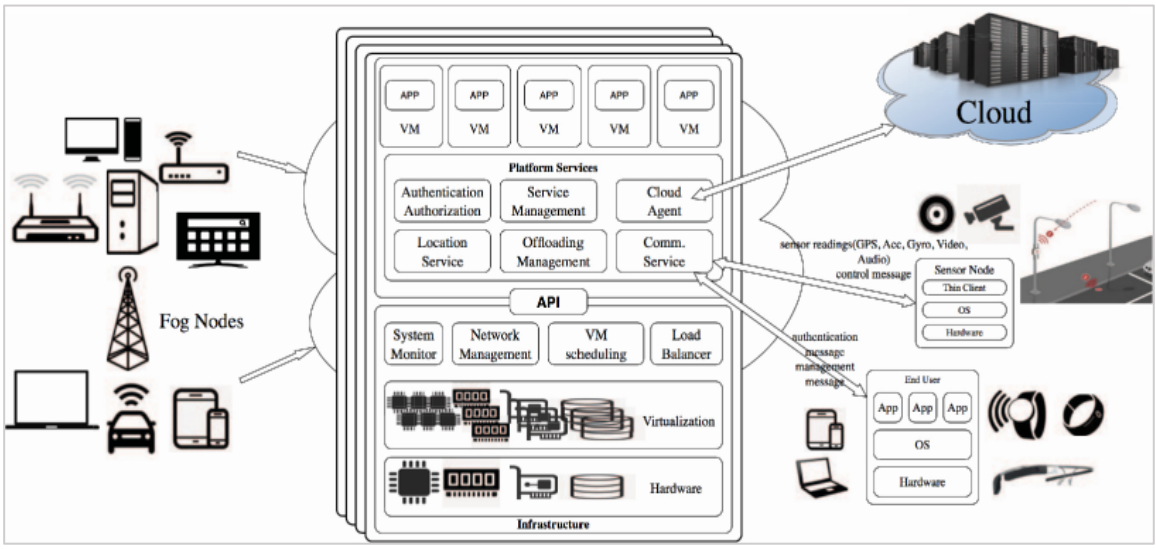


Figura 3.5: Componentes da plataforma *fog computing*.
Fonte: (Yi *et al.*, 2015)

Neste ambiente, as funcionalidades gestão de rede podem possuir bilhões de dispositivos heterogêneos que executam diferentes serviços. Esses elementos distribuídos precisam ser configurados e coordenados. As tecnologias definidas na subsecção 3.3.2 auxiliam esse processo.

Com o crescimento da *fog computing*, a disponibilização de uma arquitetura de hardware que a tornasse fácil e eficiente tornou-se premente. Como ela funciona de forma distribuída, a facilidade de utilização é um fator importante e que deve ser levado em consideração. Para isso, os sistemas *Advanced RISC Machine* (ARM) são utilizados amplamente. Visto que, os principais conceitos por trás desses sistemas são: a simplicidade, baixo custo e baixo consumo de energia.

Dastjerdi *et al.* (2016), consideram a *fog computing* como uma rede composta por um conjunto de nós dentro dos quais o processamento é distribuído. Por isso, ela pode ser menos eficiente energeticamente em comparação com o modelo *cloud*. Avançam ainda, que o hardware deve possuir uma unidade de poupança e gestão de energia cuja responsabilidade é resolver o problema de sobreaquecimento, que as unidades de refrigeração devem ser disponibilizadas pelos nós e servidores *fog*, que os subsistemas de armazenamento devem ser disponibilizados com a maior capacidade de armazenamento possível, e que quando se

pretende respostas céleres do processamento paralelo, devem ser instalados coprocessadores em servidores *fog*.

Bonomi *et al.* (2012), sugeriram uma perspetiva de arquitetura de software para a *fog* na qual foram propostas duas camadas para o sistema:

- Camada de abstração;
- Camada de orquestração.

Camada de abstração, virtualiza recursos heterogêneos com estrutura da *fog*.

Camada de orquestração é o responsável pelo planeamento da execução de todas as tarefas enviadas. As tarefas são analisadas em relação ao prazo, custo e outros fatores de QoS, o cronograma é preparado e, com base nesse gestor de execução do cronograma, as tarefas são processadas.

Segundo Yi *et al.* (2015), a plataforma *fog*, fora as tecnologias de suporte definidas na subsecção 3.3.2, é composta pelos seguintes componentes, conforme a figura 3.5:

- Monitor do sistema; é um elemento padrão em infraestruturas *cloud*, a sua principal função é disponibilizar informações úteis a outros componentes da infraestrutura. Por exemplo, pode ser utilizado em tarefas de gestão como: descobrir, afetar, provisionar e fazer a manutenção de um conjunto de recursos de forma distribuída. Também é utilizada pelo balanceamento de carga.
- Balanceamento de carga; visa distribuir as tarefas para os vários nós da *fog* por forma a possibilitar a redundância e aumentar a disponibilidade dos serviços.
- Escalonamento de máquinas virtuais; permite a distribuição das instâncias de MVs, aplicações e recursos em uma sequência lógica. Leva em consideração alguns parâmetros como: a utilização do sistema, as informações estatísticas das tarefas, as informações de localização, entre outros e seleciona a melhor estratégia para o escalonamento.

Referem ainda, que esses componentes descritos, através de API, colaboram e partilham infraestrutura de hardware virtualizado (nós, dispositivos, canais, servidores, entre outros) através da *Service-Oriented Architecture* (SOA).

Esta camada, disponibiliza as funcionalidades necessárias para a implementação dos serviços de plataforma, através dos seguintes componentes conforme ilustrado na figura 3.5:

- Gestão de serviços; disponibilizam funções que permitem: a descoberta dinâmica, a monitoração e a configuração dos serviços. Este componente permite ainda, a instalação remota de novos serviços em tempo de execução a fim de satisfazer as necessidades das aplicações. Um repositório pode ser criado para identificar o catálogo de serviços associados a cada objeto na rede para facilitar a composição de outros serviços mais complexos. Além disso, podem incluir outras funções relacionadas, por exemplo, com a QoS ou com questões semânticas como a gestão de contexto.
- Serviços de comunicação; permitem a interoperabilidade entre os níveis da plataforma e envolvem dispositivos inteligentes com sensores e atuadores, nós da *fog* e *cloud*. Em relação aos dispositivos, disponibiliza suporte a diferentes padrões de apresentação com o objetivo de garantir que os dados sejam difundidos pelas aplicações em diferentes tipos de sistemas. Da mesma forma, os agentes específicos devem garantir a comunicação com diferentes provedores *cloud*, resguardando as particularidades na operação sobre cada modelo de serviço disponibilizado.
- Mecanismos de autenticação e autorização; como a instalação da *fog* é próxima dos utilizadores finais, ela permite novas formas de autenticação e autorização através da utilização de padrões de mobilidade, de acesso, e dos dispositivos de segurança confiáveis. Dsouza, Ahn, & Taguinod (2014), propuseram uma plataforma de controlo de acesso e gestão de autorização para a *fog computing*.
- Gestão de *offloading*; tem forte impacto na plataforma porque permite transferir tarefas dos dispositivos finais para a *fog*. Os autores Shanhe Yi, Li, & Li (2015), relativamente ao *offloading* no paradigma *fog*, destacam três grandes preocupações: *(i)* quais informações são necessárias para decidir sobre *offloading*, *(ii)* como particionar uma aplicação para *offloading* e *(iii)* como projetar um esquema ideal de *offloading*, levando em consideração que o custo de transferência para *fog* pode ser superior ao tempo de processamento dos dados no próprio dispositivo.
- Serviços de localização; tem por objetivo a preservação de um conjunto de informações de localizações acerca dos nós vizinhos (fixos ou não), rastrear os

utilizadores móveis finais e partilhar as informações de localização entre os nós da *fog* envolvidos. Também faz o mapeamento da rede com localizações físicas além da adoção de um modelo de mobilidade que pode ser disponibilizado pelo utilizador ou definido de forma autônoma. O rastreamento e o mapeamento dos nós móveis precisam ter informações em diferentes níveis de comunicação como da camada física e de hardware (endereço físico, GPS, entre outros), da camada de rede (endereço IP) e da camada de aplicação (atividades sociais).

A simulação é geralmente utilizada para implementação de modelos e análise de desempenho de sistemas. A sua utilização deve-se sobretudo a dificuldade de acesso aos ambientes reais por forma a permitir aos pesquisadores desenvolver novas contribuições em diversas áreas.

Nesta tese foi utilizado o simulador *iFogSim,* descrito na secção 3.4, para implementação, análise de desempenho e comparação do modelo proposto sensível ao contexto com os algoritmos não sensível ao contexto (FCFS, SJF e *QoS-based*).

3.4. Visão geral do *iFogSim*

O *iFogSim* é uma ferramenta de simulação de código aberto escalável que possibilita modelação e simulação em infraestruturas *fog computing*. Permite simular tanto arquiteturas *fog* básicas, como as mais complexas compostas por entre outros: *data centers, brokers, hosts e* MVs. Além disso, permite fazer o aprimoramento das políticas de escalonamentos, afetação personalizada, modelação e simulação de diferentes topologias *fog*, recolhe informações sobre latência, utilização de memória, processamento, consumo de energia, tráfego na rede, entre outras. Segundo Mestre *et al.* (2019), estas funcionalidades permitem que ela seja uma ferramenta muito útil para análise de diferentes níveis hierárquicos de dispositivos em uma rede e investigações relacionadas com políticas de afetação de recursos. Autores como: Taneja & Davy (2017); Mahmud, Ramamohanarao, & Buyya, (2018); Mahmud & Buyya (2018); afirmam que ela é muito utilizada na literatura para estudos e simulações em ambientes *fog*.

Para Mestre *et al.* (2019), um simulador é geralmente composto por três componentes principais: componente físico, lógico e de gestão. Estes componentes são descritos no Anexo B desta tese.

A afetação dos módulos para serem executados no *iFogSim* é feito através das políticas *standard* também descritos no Anexo B desta tese. No entanto, podemos idealizar, melhorar e/ou propor outras abordagens.

Nesta tese, utilizamos a política *placement edgewards* e propomos uma abordagem sensível ao contexto com intervalos e escalonamentos estáticos e dinâmicos e realizamos uma análise comparativa em termos de desempenho com abordagens não sensível ao contexto.

3.5. Sensibilidade ao contexto e *fog computing*

Como descrito na subsecção 2.2.1, contexto pode ser definido como ambiente, estrutura, cenário ou situação que envolve uma entidade, evento ou ocorrência. Em computação móvel, contexto refere-se ao ambiente de processamento, ambiente do utilizador, ambiente físico, relevante para a interação entre um utilizador e uma aplicação, incluindo o utilizador e as próprias aplicações (Musumba & Nyongesa, 2013). Por outro lado, *fog Computing* é definido como um paradigma da *cloud* que visa suprir limitações deste através da disponibilização de serviços na extremidade da rede. Isto é, mais próximas dos utilizadores finais e do lugar onde os dados são produzidos (Stojmenovic, 2014).

Os dispositivos móveis quando comunicam com a *cloud*, enfrentam alta latência da rede e grande consumo de energia de transmissão (Jararweh *et al.,* 2013). Avançam ainda, que no paradigma *fog*, os dispositivos móveis enviam tarefas para os nós da *fog* a fim de serem processadas e devolvidas o resultado. Esse processo reduz, entre outros, o atraso na transmissão e o consumo de energia do dispositivo móvel. Devido a menor capacidade dos nós da *fog* quando comparada com a *cloud*, as tarefas, que não podem ser executadas na *fog*, são enviadas para serem executadas na *cloud*.

4. ESCALONAMENTO DE TAREFAS

Neste quarto capítulo, fizemos uma revisão da literatura sobre os algoritmos de escalonamento de tarefas na arquitetura cloud e no paradigma fog, estudámos e discutimos as suas limitações, explorámos e sugerimos algumas perspetivas de melhoria.

4.1. Escalonar tarefas

Escalonamento de tarefas refere-se à atribuição de recursos necessários para a execução de uma tarefa. Assume-se como um processo essencial para melhorar a confiabilidade e a flexibilidade dos sistemas. Exige algoritmos avançados capazes de escolher o recurso mais adequado disponível para executar a tarefa (Swaroop, 2019).

Pretende-se que os pedidos sejam executados tendo em consideração as restrições definidas. Estas restrições podem ser: tempo, custo, duração da bateria, níveis de sinal de redes, QoS da aplicação, entre outros (Swaroop, 2019). Avança ainda, que devido a sua complexidade, o escalonamento no paradigma *fog*, apresenta alguns desafios que nos leva a questionar a forma como as tarefas são encaminhadas entre dispositivos clientes, nós da *fog*, servidores *cloud*, entre outros.

Sistemas lidam com pedidos prioritários, tarefas prioritárias e com requisitos restritos de QoS. Para garantir a execução das tarefas dentro dos limites de tempo definidos, o escalonamento precisa ser tratado com ponderação. Escalonamento eficiente de tarefa deve garantir processamento simultâneo e eficiente de tarefas independentes do seu fluxo. No paradigma *fog*, o escalonamento constitui um desafio. Exige algoritmos avançados capazes de escolher o recurso mais adequado e disponível para executar a tarefa (Mahmud *et al.*, 2016). Avançam ainda, que um algoritmo de escalonamento deve tomar duas decisões importantes, que podem ser baseadas em alguns valores predefinidos ou através de dados dinâmicos obtidos durante a execução da tarefa:

- Determinar as tarefas que podem ser executadas em paralelo;
- Definir onde executar essas tarefas paralelas.

Avançam ainda, que no caso específico da *fog* existem dois principais estágios envolvidos:

- Fase de provisionamento de recursos, em que se deteta, classifica e provisiona os recursos necessários para a execução da tarefa;
- Fase de mapeamento de tarefas, fase na qual, um servidor/processador adequado é identificado e a tarefa é mapeada para esse servidor/processador.

As decisões em cada etapa, são tomadas tendo em conta algumas restrições como: custo e tempo de processamento, QoS da aplicação, informações do contexto, entre outros.

Alguns algoritmos discutidos na secção 4.2, focam apenas nas restrições do tempo, outros nas restrições de energias. Alguns têm como principais objetivos reduzir o tempo necessário para a execução das tarefas, outros abordam o problema da otimização na perspetiva dos provedores de serviço ou otimizam os níveis de QoS da aplicação. No entanto, ignoram informações contextuais ao nível do dispositivo e dos utilizadores finais e as suas experiências de utilização.

4.1.1. Conceção de algoritmos de escalonamento

Conforme Swaroop (2019), na conceção de um algoritmo de escalonamento, devem ser obedecidas restrições como custo das tarefas, dependências entre as tarefas e a sua localização. No que respeita ao custo das tarefas, devemos procurar respostas de questões como: que informações temos sobre o custo das tarefas utilizadas? Todas as tarefas têm o mesmo custo? Esse custo só é conhecido durante a execução?

Relativamente às dependências entre as tarefas, devemos questionar sobre: quais as tarefas que são dependentes e quais as que são independentes? Quais são os processos que possuem subprocessos?

Sobre a localização, devemos considerar as seguintes questões: onde executar as tarefas por forma a reduzir o custo total de execução? Como minimizar o custo de comunicação? E como ter informações sobre os requisitos de comunicação?

4.1.2. Decisões de escalonamento

As decisões de escalonamento podem ser estáticas ou dinâmicas (Swaroop, 2019). No escalonamento estático, a decisão sobre o escalonamento é tomada durante a compilação. Para isto, são utilizados alguns métodos da análise estática que permitam definir o tamanho da tarefa.

Não é fácil a obtenção dessas informações porque, muitas vezes, são disponibilizadas de forma incompleta. Avança ainda, que no escalonamento dinâmico, que também é conhecido como prática de partilha de tarefas adaptável, utiliza informações sobre o estado da tarefa num determinado instante durante a sua execução para tomar decisões. Constitui a melhor abordagem, visto que possibilita que vários problemas tenham solução. São, no entanto, exigentes computacionalmente e requerem estratégia de paralelização e balanceamento de carga dinâmico.

4.2. Algoritmos de escalonamentos na arquitetura *cloud* e *paradigma fog*

Nesta secção, começamos por descrever a metodologia utilizada para a revisão de literatura sobre os algoritmos de escalonamento de tarefas na arquitetura *cloud e* no paradigma *fog.* Seguidamente esses algoritmos foram categorizados, considerando suas descrições, propriedades e características, em: básicos; baseados em prioridade e orientado a QoS e sensível ao contexto.

4.2.1. Enquadramento

Com o objetivo de alcançar maior de rigor científico no levantamento feito, assegurámos o processo de investigação com base nas perspetivas da revisão sistemática da literatura, que segundo Kitchenham (2004), identifica, avalia e interpreta todas as pesquisas disponíveis relevantes para uma determinada área temática, uma questão específica, ou um fenómeno de interesse. Avança ainda, que a sua concretização deve considerar as seguintes fases: preparação da revisão; definição de metodologias para a execução e extração da revisão sistemática e procedimentos para a criação de relatórios da revisão efetuada.

Devido à sua rigorosidade e possibilidade de iterações, os resultados obtidos pela revisão sistémica são mais confiáveis comparativamente à revisão de literatura primária (Kitchenham, 2004).

A revisão feita nesta secção teve em conta a aplicação dos seguintes passos metodológicos:

1. Foi criada uma base de dados de propostas científicas utilizando os seguintes parâmetros: artigos completos disponíveis *online,* e publicados no período de 2015 a 2020 que tratam tópicos relevantes como: *task scheduling or scheduling algorithm and scheduling cloud computing or scheduling fog computing and cloud-fog computing.* Utilizaram-se bases de dados *ScienceDirect* (http://www.sciencedirect.com/), 371 artigos, *IEEE Explore* (http://ieeexplore.ieee.org/), 108 artigos, *Google Scholar* (https://scholar.google.pt/), 278 artigos, ACM (https://dl.acm.org/), 302 artigos e SCOPUS (https://www.scopus.com), 200 artigos, totalizando 1259 artigos.

O gráfico 4.1, ilustra os artigos identificados em diferentes bases de dados científicos.

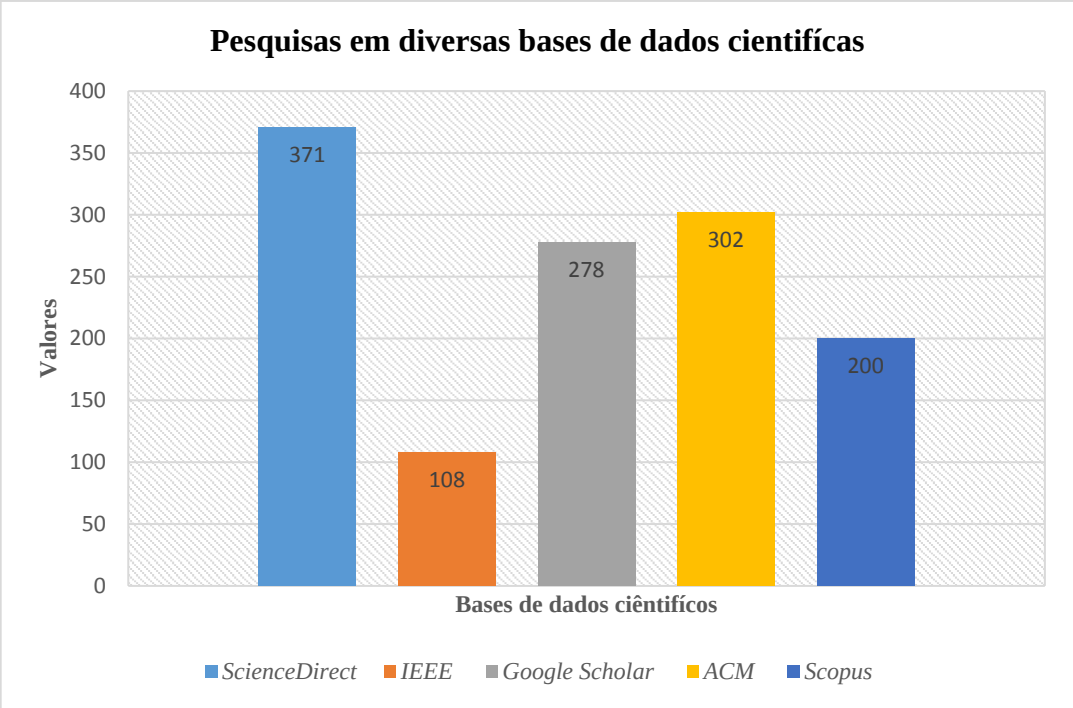


Gráfico 4.1: Artigos identificados
Fonte: o autor.

Os resultados foram importados para o software *Publish or Perish* (Harzing, 2020), uma aplicação gratuita que disponibiliza de forma simples métricas sobre os artigos, nomeadamente, ao nível de total de citações, ajustado ao ano, deteção automática de duplicados, entre outras.

2. Após a limpeza dos duplicados de 143 artigos, compilou-se uma lista de 1116 artigos, dos quais foram removidos 68 por não possuirmos o acesso ao documento.

O gráfico 4.2, ilustra o total dos artigos em que não tivemos acesso, duplicados e os analisados.

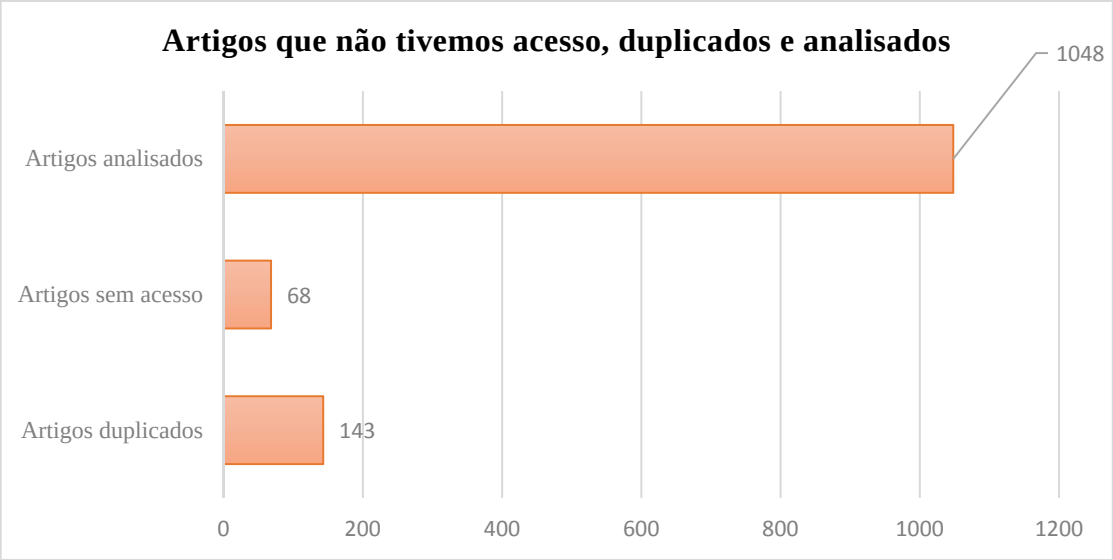


Gráfico 4.2: Filtragem dos artigos identificados
Fonte: o autor.

3. Com base na lista de 1048 artigos, procedemos à exclusão de propostas não essencialmente relevantes por "não ser uma proposta de escalonamento de tarefas, ou algoritmo de escalonamento de tarefas" e "não ser relacionada com *cloud* ou *fog computing*". Esta exclusão foi realizada por análise de títulos e resumos dos artigos e, em caso de dúvida, por análise de artigos. Foram removidos os artigos de tipo: *Survey* (53); Revisões teóricas ou *position papers* (78); Não ser uma proposta de escalonamento de tarefas, ou algoritmo de escalonamento de tarefas (184); Não relacionadas com *cloud* ou *fog computing* (450).

Após esta análise, obteve-se uma lista de 283 artigos, conforme o gráfico 4.3.

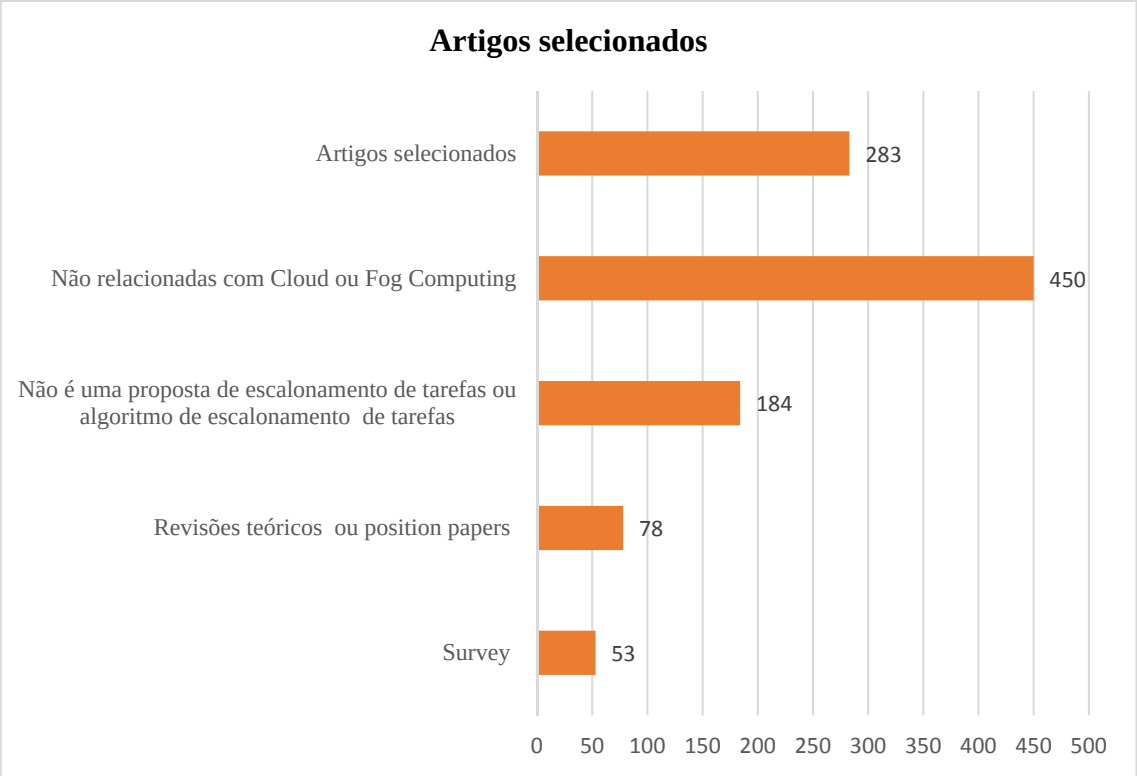


Gráfico 4.3: Seleção dos artigos
Fonte: o autor

4. Seleção, por inspeção manual, através da revisão integral dos artigos, de um conjunto reduzido de propostas, a partir de 283 artigos, com base nos seguintes critérios:

   - Ordenação por número de citações na literatura ajustada ao ano (calculada pelo *software Publish or Perish*), permitiu obter um conjunto de propostas que foi analisado e validado por outros autores e remover o enviesamento provocado pelo tempo de publicação;
   - A relevância para a temática, possibilitou obter um conjunto de propostas que corresponde às necessidades inseridas dentro da temática dos algoritmos de escalonamento de tarefas na arquitetura *cloud* ou paradigma *fog*, ou seja, propostas de escalonamento de tarefas na arquitetura *cloud* ou paradigma *fog*. Foram excluídos 247 artigos por não serem relevantes e seis por não possuirmos acesso ao texto integral.

Com base nesta metodologia de seleção, obteve-se uma amostra de 30 propostas, que foram categorizados, considerando suas descrições, propriedades e características, em: básicos;

baseados em prioridade e orientado a QoS e sensível ao contexto, conforme apresentada e analisada nas subsecções 4.2.2 à 4.2.4.

A figura 4.1 apresenta o fluxograma da sistematização da revisão da literatura efetuada.

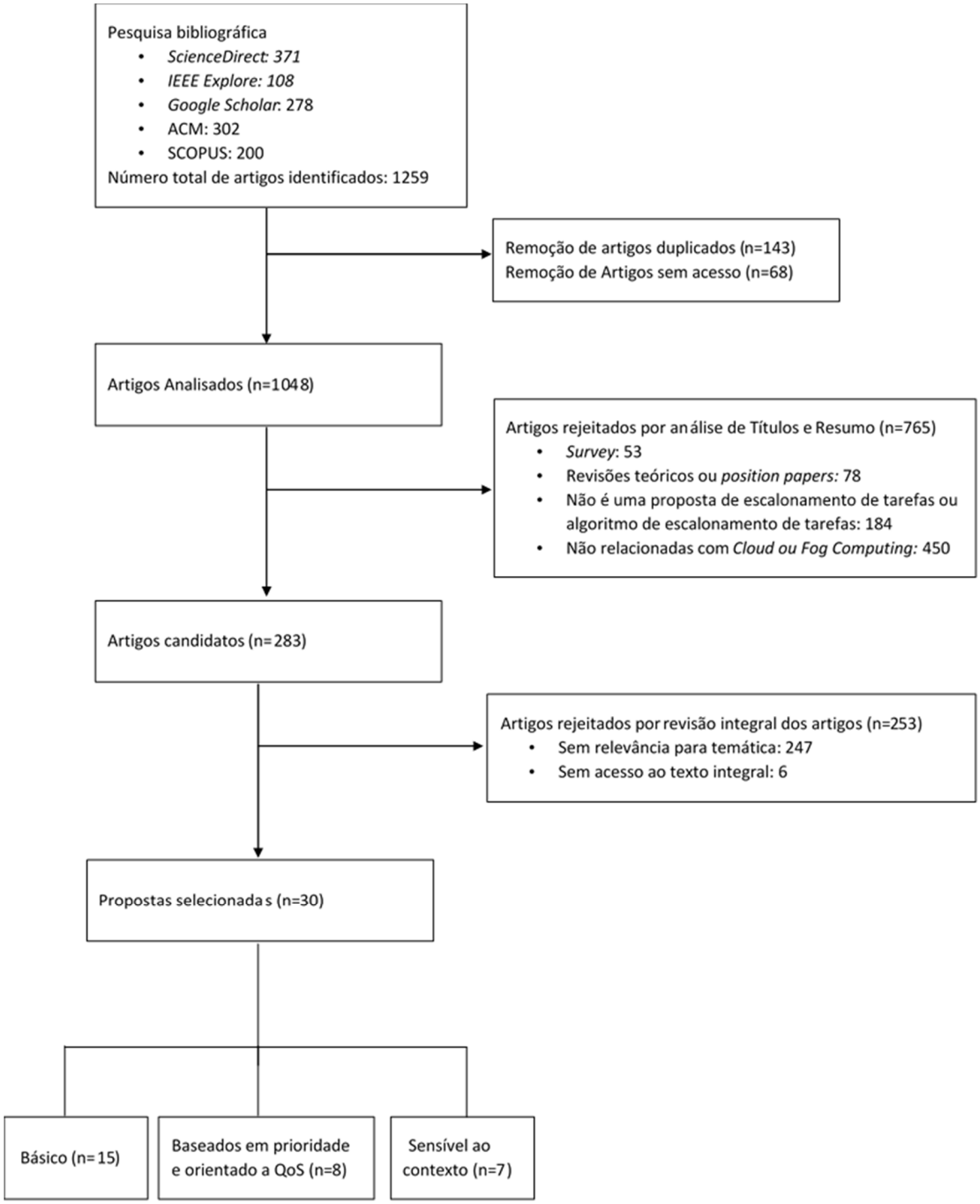


Figura 4.1: Fluxograma da sistematização da revisão da literatura.
Fonte: O autor

4.2.2. Algoritmos de escalonamentos básicos

Tanto na arquitetura *cloud* como no paradigma *fog*, encontrámos alguns algoritmos de escalonamento que, pelas suas características, foram definidos como básicos.

Deng *et al.* (2016), examinaram a relação entre o atraso na transmissão e o consumo de energia na *cloud* e avançaram que o paradigma *fog* permite mobilidade, distribuição geográfica, localização e mobilidade do utilizador suportadas com o apoio dos dispositivos da *fog*. A distribuição geográfica pode disponibilizar informações de contexto da rede, de forma a poder ser utilizada pela aplicação na *fog*.

Qiu *et al.* (2015), propuseram o *Energy-Aware Dynamic Task Scheduling* que visa minimizar o consumo de energia dos sistemas ciberfísicos, em que se destacam os *smartphones* que, devido as suas limitações, têm a necessidade de fazer muitas e frequentes interações com a *cloud* a fim de executarem tarefas como: solicitar recursos, enviar pedidos e receber respostas. Estas tarefas são simples, mas, pela sua frequência, consomem muita energia. Para alcançar o desempenho ótimo, propuseram resolver dois problemas: o primeiro visa diminuir ao mínimo o custo de energia e manter o desempenho; o segundo tenciona tornar as tarefas altamente confiáveis. Isto é, possibilitar que as tarefas terminem as suas execuções com alta probabilidade de sucesso.

Lawanyashri, Balusamy, & Subha (2017), apresentaram o *Fruity Fly Algorithm*, a abordagem proposta está voltada não só para a questão do balanceamento de carga, mas também enfatiza a redução de consumo de energia nos *data centers*. O método proposto conseguiu otimizar a utilização de recursos e permitiu reduzir o custo e o consumo de energia.

Tiwary *et al.* (2018), conceberam um modelo baseado em jogo em que o jogador maximiza a sua recompensa, por forma a minimizar o tempo de resposta. Isto é, há uma minimização do tempo de resposta na extremidade dos descarregadores de tarefas e uma maximização da vida útil da bateria nos recetores de tarefas.

Yang *et al.* (2018), propuseram o *Delay Energy Balanced Task Scheduling* (DEBTS), como uma estrutura analítica multicamadas para formular e estudar o equilíbrio entre o atraso do serviço e o consumo de energia no escalonamento de tarefas em redes *fog*.

Fan *et al*. (2017), apresentaram o *Deadline-Aware Task Scheduling Mechanism,* através dele, os provedores de serviços exploram a colaboração entre os nós da *fog* e os recursos da *cloud* alugados para executar com eficiência as tarefas dos utilizadores em grande escala geográfica. E como principal objetivo visa maximizar os lucros do provedor de serviços *fog* através da satisfação do tempo definido para a execução das tarefas.

Gawali & Shinde (2018), conceberam um método de otimização denominado BATS+BAR para executar o escalonamento de tarefas e atribuição de recursos em computação distribuída e paga conforme a utilização. A estrutura proposta classifica as tarefas e os atribuem de acordo com a restrição do tamanho e largura de banda em uma máquina virtual. Compararam ambos os algoritmos em termos da métrica de desempenho e averiguaram o aperfeiçoamento.

Wang, Wang, & Cui (2016), apresentaram o *Energy-Aware Multi-Job Scheduling Model* baseia-se no *Map Reduce* e tem por objetivo melhorar a eficiência energética através do ajuste do CPU do servidor.

Sahoo & Dehury (2018), realizaram uma pesquisa sobre as características da *cloud* e a forma como ela disponibiliza vários recursos aos clientes, através de diversos modelos de serviço. Propuseram um algoritmo que permite escalonar tarefas relacionadas com a saúde, geridas e armazenadas em *data center*.

Sarkar, Chatterjee, & Misra (2018), avaliaram a aplicabilidade do paradigma *fog* com o propósito de responder as demandas das aplicações sensível à latência no contexto da *IoT* e concluíram que o objetivo da *fog*, que visa proporcionar respostas aos pedidos em tempo real e com baixa latência, favorece a sua utilização no contexto da *IoT*. A adequação da *fog* no contexto dos pedidos em tempo real e o desempenho em relação ao histórico da computação distribuída foi avaliada e alcançou-se um reduzido consumo de energia, baixas taxas de emissão de gases de dióxido de carbono, latência e custo.

Bitam, Zeadally, & Mellouk (2018), apresentaram o *Bees Life Algorithm* (BLA) que visa encontrar o equilíbrio entre o tempo de execução de tarefas no CPU e a sua alocação na memória. O algoritmo proposto alcançou o objetivo de minimizar o tempo de execução comparado com os algoritmos *Particle Swarm Optimization* e *Genetic Algorithm*.

Uma abordagem híbrida, baseada na teoria *fuzzy* e no algoritmo genético chamado FUGE foi apresentada em Shojafar *et al.* (2015). Ela visa executar o balanceamento de carga ótimo considerando o tempo e o custo de execução.

O *Hybrid Fog and Cloud-Aware Heuristic* (Hybrid-EDF) foi proposto e descrito em Stavrinides & Karatza (2019). Esta abordagem utiliza espaços no escalonamento das máquinas virtuais da *cloud* e *fog* para escalonar tarefas exigentes com baixos requisitos de comunicação na *cloud* e tarefas intensivas de comunicação com baixos requisitos computacionais na *fog*. Durante o escalonamento, essa abordagem considera o custo da comunicação decorrente da transferência de dados dos sensores e dispositivos da camada da *fog*. A performance da heurística proposta foi avaliada e comparada com o algoritmo *Baseline Cloud-Unaware Strategy*, *Fog*-EDF.

Li *et al.* (2017), desenvolvera o *Cooperative-Based Model for Smart–Sensing Tasks* (CMST) que promove a qualidade da recolha de dados em regiões urbanas e suburbanas para a alimentação de plataformas através do paradigma *fog*. Utilizaram um algoritmo de escalonamento cooperativo focado em melhorar as recompensas associado à recolha de dados nas regiões suburbanas. As recompensas para cada utilizador com um sensor inteligente são distribuídas de acordo com a densidade da região. Além disso, para cada tarefa, há uma relação de cooperação entre os utilizadores, que cooperam entre si para alcançar o volume de dados que a plataforma necessite.

4.2.3. Algoritmos de escalonamentos baseados em prioridade e orientado a QoS

Pesquisas sobre escalonamento de tarefas baseados em prioridade e orientado a QoS no paradigma *fog* são recentes. Em Skarlat *et al.* (2017), foi definida uma abordagem de escalonamento que visa a colocação de aplicações sensível à QoS em recursos virtualizados da *fog*. A política proposta passa por uma orquestração baseada em colónia entre os nós da *fog* e concilia os requisitos de recursos das aplicações com os recursos disponíveis do sistema. Quando colónias necessitarem de recursos adicionais, conecta a *cloud* através de um *middleware*.

Gill, Garraghan, & Buyya (2019), avançaram que um dos principais desafios do paradigma *fog* é a gestão de recursos e que embora no paradigma *fog* padrão existam gerenciadores de recursos, eles concentram apenas em gerir um subconjunto importante de parâmetros que abrange o tempo de resposta do sistema, largura de banda da rede, consumo de energia e

latência e que, até ao momento, nenhum gestor de recursos da *fog* considerou todos esses parâmetros simultaneamente para a tomada de decisões. Propuseram uma nova técnica denominada ROUTER, que aproveita da *Particle Swarm Optimization* para aprimorar simultaneamente os parâmetros: tempo de resposta do sistema; largura de banda da rede; consumo de energia e latência. E possibilitar uma melhor tomada de decisões.

Em vez disso, Aazam *et al.* (2016), desenvolveram o *Media fog Resource Estimation* (MeFoRE), que considera o histórico de renúncia de serviço do utilizador e QoE na priorização dos pedidos de serviço e estima os recursos da *fog* com o objetivo de maximizar a utilização dos recursos e QoS. As violações do *Service Level Agreement* (SLA) são monitorizadas através de baixos valores de QoE. A quantidade de recursos aumenta com base no grau de violações das SLA e, com base nisso, a confiança do utilizador pode ser recuperada.

Oueis, Strinati, & Barbarossa (2015), abordaram a questão do balanceamento de carga no paradigma *fog*, com o objetivo de melhorar a QoE dos utilizadores. Os autores sugerem que todos os pedidos de diferentes utilizadores cuja computação necessitar de ser descarregada são executadas em *clusters* de recursos locais. Nesta proposta, foi introduzido um algoritmo de escalonamento de tarefas de complexidade reduzida para a *fog* em que os recursos são alocados com o fim de servir pequenas células (o nó da *fog*) com base em algumas regras de escalonamentos específicos. A principal regra define a afetação de recursos computacionais locais em cada pequena célula, para servirem os utilizadores internos. Essas pequenas células classificam os pedidos de descarregamento dos utilizadores de acordo com um parâmetro específico, como: hora de chegada; restrição de latência; entre outros. Esta classificação define também a política de escalonamento (por exemplo, *First In First Out* (FIFO), *Earliest Deadline First* (EDF)) a ser adotada para a afetação de recursos locais. Dessa forma, podem ser dadas diferentes prioridades aos distintos pedidos dos utilizadores, em função do parâmetro da ordenação.

Cardellini *et al.* (2015), avaliaram o escalonador distribuído de qualidade de serviço sensível ao QoS para processamento de fluxo de dados (DSP) em ambiente *fog*. Inovaram através da introdução de novos componentes como: monitor operário; monitor de QoS e um escalonador adaptativo nativo. O monitor operário, nessa arquitetura, desempenha a função de obter a taxa de entrada e saída de dados para cada executor, que é definido como uma

componente de computação que executa um conjunto de tarefas no nó da *fog*. Essa taxa de dados de entrada e saída é armazenada numa base de dados local para ser posteriormente utilizada pelo escalonador adaptativo. O monitor de QoS estima os parâmetros de QoS (por exemplo, latência de rede) e é responsável pela obtenção da utilização e disponibilidade intra-nó e informações inter-nós. Essas informações são enviadas para o escalonador adaptativo distribuído, que implementa a política de escalonamento do sistema. O escalonador adaptativo executa uma iteração de ciclo único periodicamente, com o intuito de verificar se a tarefa candidata será executada (chamado executor móvel). Se o executor for efetivamente alocado, o escalonador adaptável executa as ações correspondentes. Para isso, o escalonador determina um nó que executará o executor candidato apenas se esse nó melhorar o desempenho da aplicação em termos de recursos de tempo de execução.

Os autores Intharawijitr, Iida, & Koga (2016), definiram um modelo matemático de uma rede *fog* para auxiliar um sistema de computação a alcançar a eficiência máxima, garantindo um escalonamento ótimo de tarefas. Para o escalonamento de tarefas na *fog*, três políticas foram consideradas neste estudo: a política aleatória em que um nó da *fog* é selecionado aleatoriamente a partir de uma distribuição uniforme para executar uma tarefa; a política de menor latência, em que um nó da *fog* disponibiliza a tarefa com a menor latência com base no estado atual do sistema e, finalmente, a política de capacidade máxima disponível, que seleciona o nó da *fog*, com o máximo de recursos excedentes entre os nós candidatos. Os resultados das simulações mostraram que a política de menor latência proporciona um desempenho significativamente melhor, devido à disponibilidade de recursos.

Os autores concluíram que todas as políticas podem ser utilizadas para encontrar o nó mais adequado da *fog* para uma tarefa. A utilização de uma determinada política pode, no entanto, não ser a solução ideal para todo o sistema.

Bittencourt *et al.* (2017), analisaram o problema de escalonamento de tarefas na *fog* e focaram na forma como a mobilidade dos utilizadores influência o desempenho das aplicações. Avaliaram como os algoritmos de escalonamento: *Concurrent, First Come First Served e Delay-Priorit* podem ser utilizadas para melhorar a execução da tarefa, tendo por base as características das aplicações.

Contrariamente a *fog*, o escalonamento de tarefas baseados em prioridade e orientado a QoS foi amplamente estudado em *cloud* (Barros e*t al.,* 2020).

Em Xiaojiang *et al.* (2015) foi proposto um algoritmo com base nos atributos de tarefas como: privilégio do utilizador; expetativa; volume da tarefa e tempo suspenso na fila. A prioridade é calculada e seguidamente, as tarefas são escalonadas com base no tempo mínimo de conclusão que proporciona melhor desempenho e balanceamento de carga.

O *Offline Multi-Level Scheduling* foi proposto por Seddik & Hanzálek (2017), procura resolver o problema do nível de criticidade de cada tarefa, com base na sua prioridade, leva em consideração o tempo total de processamento das tarefas tendo em vista o objetivo final. O método proposto alcançou tarefas de maior criticidade, que necessitavam de maior tempo de execução e processamento.

4.2.4. Algoritmos de escalonamento sensíveis ao contexto

Segundo o conhecimento dos autores, pesquisas sobre algoritmos de escalonamento sensível ao contexto no paradigma *fog* são muito recentes e, na arquitetura *cloud,* encontramos algumas propostas.

Em Shi *et al.* (2016), é apresentado um escalonador probabilístico adaptativo, que otimiza o consumo de energia de tarefas com restrição de tempo. Em Zhou *et al* (2017), é proposto o *mcloud: Context-Aware Offloading Framework*, um algoritmo de decisão de descarga sensível ao contexto, que leva em consideração as mudanças de contexto como: níveis da rede e recursos da *mobile cloud* heterogêneos para disponibilizar decisões de descarga de código. Disponibilizou-se também um modelo de estimativa de custo para recursos de infraestrutura da *mobile cloud* que visa estimar o custo de execução da tarefa, incluindo tempo de execução e consumo de energia.

Em Mahmud *et al.* (2016), é apresentado uma política de escalonamento sensível ao contexto para *mobile cloud computing* e executada em uma *cloudlet.* Utiliza o algoritmo *Highest Priority Job First* (HPJF) para a ordenação, preempção e escalonamento de tarefas.

O *Multi-Capacity-Aware Resource Scheduling* é proposto em Sheikhalishahi *et al.* (2016), para resolver o problema de escalonamento multi-recursos. O plano de escalonamento permite escolher qualquer tarefa com base em requisitos de recursos semelhantes ao sistema disponível e com uma meta final específica que visa tratar a utilização de recursos num contexto de vários recursos.

Ghouma & Jaseemuddin (2015), propuseram o *Context Aware Cloud System*, política de escalonamento e afetação de recursos sensível ao contexto que monitoriza e utiliza as informações de contexto do *smartphone* do utilizador, para tomar decisões inteligentes sobre a afetação de recursos na *cloud* e escalonar tarefas. As informações de contexto do utilizador móvel como: qualidade da conexão do dispositivo e o nível de bateria, são utilizados na definição dos serviços para os fluxos de tarefas da aplicação que são enviadas e processados na *cloud*. O modelo proposto visa racionalizar a utilização de recursos em ambiente *cloud* e provê melhoria significativa da qualidade de serviço.

Mahmud *et al*. (2019), propuseram uma política de escalonamento de aplicações sensível ao QoE e que prioriza diferentes pedidos de escalonamento de acordo com as expectativas dos utilizadores e calcula os recursos das instâncias da *fog* levando em consideração o seus estado atual.

4.3. Discussão

Alguns autores (cf. Sheikhalishahi *et al.* (2016), Lawanyashri, Balusamy, & Subha (2017), Tiwary *et al.* (2018) e Gawali & Shinde (2018)), analisam o escalonamento sob a perspetiva dos provedores de serviços e ignoram questões contextuais dos utilizadores finais.

Os autores em Deng *et al.* (2016), Li *et al.* (2017), Lawanyashri, Balusamy, & Subha (2017) e Yang *et al.* (2018), propuseram abordagens que enfatizam a redução do consumo de energia dos *data centers* e nota-se uma grande preocupação com a eficiência energética.

Em Xiaojiang *et al.* (2015), é proposto um algoritmo de escalonamento de tarefas baseado em QoS para a *mobile cloud server* onde os atributos de tarefas como privilégios dos utilizadores; tamanho da tarefa; expetativa e o tempo suspenso na fila são utilizados para calcular a prioridade.

Cardellini *et al.* (2015), avaliaram o escalonador distribuído de qualidade de serviço sensível ao QoS para o processamento de fluxo de dados no paradigma *fog*. O algoritmo proposto permitiu mostrar que o escalonador distribuído sensível ao *QoS* supera o padrão centralizado, melhora o desempenho da aplicação e aprimora o sistema com recursos de adaptação em tempo de execução. No entanto, em topologias *fog* complexas, que envolvem muitos operadores podem causar alguma instabilidade o que pode limitar a disponibilidade da aplicação DSP.

O algoritmo de escalonamento proposto em Stavrinides & Karatza (2019), leva em consideração o custo da comunicação decorrente da transferência de dados dos sensores e dispositivos da camada da *fog* durante o processo de escalonamento.

Aazam *et al.* (2016), propuseram um algoritmo que visa otimizar a utilização dos recursos e QoS. Skarlat *et al.* (2017), definiram uma política de escalonamento que visa a colocação de aplicações sensível à QoS em nós da *fog*. Estes algoritmos de escalonamento não exploram os níveis de conectividade dos dispositivos para priorizar a execução de tarefas. Intharawijitr, Iida, & Koga (2016), tentam garantir a melhor utilização da largura de banda disponibilizada.

Em Sheikhalishahi *et al.* (2016), Deng *et al.* (2016), Lawanyashri, Balusamy, & Subha (2017) há uma evidente preocupação dos autores com a eficiência energética dos recursos.

Os algoritmos propostos em Shojafar *et al.* (2015) e Fan *et al.* (2017), pretendem cumprir o *deadline* das tarefas e aumentar os lucros dos provedores de serviços da *fog.* Visam ainda realizar um balanceamento de carga ótimo considerando o tempo e o custo de execução das tarefas.

Nestas cinco últimas propostas, há uma evidente preocupação dos autores em focar e defender principalmente os interesses dos provedores de serviços, ao invés de considerarem os contextos dos utilizadores finais e as suas experiências de utilização.

Li *et al.* (2017), desenvolveram um algoritmo de escalonamento cooperativo focado em melhorar as recompensas associado as recolhas de dados nas regiões suburbanas, onde os utilizadores cooperam entre si para alcançar o volume de dados que a plataforma requer.

O algoritmo proposto por Ghouma & Jaseemuddin (2015), explora contexto como níveis da conetividade da rede do dispositivo e o nível de bateria, para a definição das tarefas a serem escalonadas. Enquanto que o algoritmo proposto por Zhou *et al.* (2017), exploram os níveis da conectividade da rede e os recursos das MV alugadas para disponibilizar decisões de descarga de códigos. Ignoram, no entanto, os parâmetros importantes do contexto como os requisitos de QoS, tráfego na rede, largura de banda da rede, entre outros, que podem influenciar extensivamente as tomadas de decisões.

Mahmud *et al.* (2016), propõem uma política de escalonamento de aplicações sensível ao contexto para *mobile cloud computing* que utiliza o algoritmo HPJF para a ordenação,

preempção e escalonamento de tarefas, o que não é o mais adequado em muitos cenários, visto que ela é uma variante do algoritmo FCFS, não é preemptivo e em variadas situações é muito difícil a definição das prioridades das tarefas.

O algoritmo proposto em Zhu *et al.* (2015), é utilizado em tarefas agrupadas com objetivo de minimizar o tempo de execução. O proposto em Oueis, Strinati, & Barbarossa (2015), apesar de proporcionar ganho de latência e baixo consumo de energia, tal como os propostos em Cardellini *et al.* (2015) e Zhu *et al.* (2015), o desempenho degrada-se quando é utilizado no paradigma *fog* em grande escala.

Em Gill, Garraghan, & Buyya (2019), é proposta uma técnica de gestão de recursos para ambientes *cloud* e *fog* sensível ao QoS que, contrariamente ao escalonador padrão da *fog*, aproveita da otimização *Particle Swarm* para otimizar simultaneamente vários parâmetros do contexto. Enquanto que em Aazam *et al.* (2016), é proposto um algoritmo que objetiva estimar os recursos da *fog* com intuito de otimizar a QoE. Em vez disso, em Skarlat *et al.* (2017), a abordagem proposta visa escalonar aplicações sensível à QoS em recursos virtualizados da *fog*. Os autores consideraram a satisfação do tempo limite definido para a execução das aplicações como métrica de QoS. Concentraram na otimização do tempo de resposta entre os componentes da *fog* e não consideraram o tempo de serviço nem a QoE dos utilizadores em caso de vida útil de bateria ser reduzida e baixa conectividade da rede.

Contrariamente, em Zhu *et al.* (2015), Oueis, Strinati, & Barbarossa (2015), Mahmud *et al.* (2016) e Aazam *et al.* (2016), foram tidas em conta as questões relacionadas com o aprimoramento da QoE do utilizador final.

Muitas propostas, como as descritas em Sheikhalishahi *et al.* (2016), Lawanyashri, Balusamy, & Subha (2017), Tiwary *et al.* (2018), e Gawali & Shinde (2018), abordam o problema da otimização sob a perspetiva dos provedores de serviço e ignoram questões contextuais dos utilizadores finais e as suas experiências de utilização. Outras, como as definidas em: Cardellini *et al.* (2015), Aazam *et al.* (2016), Skarlat *et al.* (2017) e Gill Garraghan, & Buyya (2019), pretendem sobretudo otimizar os níveis de QoS da aplicação e alguns concentram apenas no escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog*. Outros, ainda, preocupam-se com a eficiência energética.

Apesar dos muitos esforços, ainda existem vários desafios relacionados com o escalonamento de tarefas na arquitetura *cloud* e no paradigma *fog*. Seguidamente, na

subsecção 4.3.1, apresentaremos algumas limitações dos escalonadores analisados na secção 4.2.

4.3.1. Limitações dos algoritmos de escalonamento estudados

Para Barros *et al.* (2020b), diferentes escalonadores têm as suas próprias deficiências, seguidamente destacaremos algumas limitações dos que foram apresentados na secção 4.2:

- *Análise de políticas na perspetiva de serviços*, a maioria dos escalonadores analisa as políticas apenas na perspetiva de serviço. A otimização dos custos para utilizadores, bem como o aperfeiçoamento da qualidade de experiência dos utilizadores não são tidas em consideração.
- *Desconhecimento do contexto do utilizador final*, nas técnicas de escalonamento sensível ao contexto estudados, os pedidos dos utilizadores finais são analisados num escopo restrito. A força do sinal associado a um pedido, por exemplo, não é tida em consideração. Como consequência, qualquer dispositivo pode ficar desconectado antes ou enquanto obtém resposta de um pedido. O nível da bateria do dispositivo dos utilizadores finais também é ignorado. Para garantir que um pedido sempre tenha respostas em tempo oportuno, um limite para o nível da bateria deve ser preservado.
- *Escalonamento deficiente de tarefas*, escalonadores de tarefas básico como as definidas em Li *et al.* (2017), Lawanyashri, Balusamy, & Subha (2017), Yang *et al.* (2018), Wang, Wang, & Cui (2016), e Qiu *et al.* (2015), são aqueles que privilegiam a eficiência energética e não consideram a sensibilidade ao contexto, a qualidade de serviço e a experiência dos utilizadores.
- *Priorização inadequada de tarefas*, alguns escalonadores baseados em prioridade foram estudados. No entanto, muitos não descrevem a forma como a prioridade é definida e outros não explicam claramente a metodologia utilizada para a priorização de tarefas.
- *Aumento do tempo médio de espera*, geralmente à medida que os pedidos aumentam, o tempo médio de espera tende também a aumentar proporcionalmente. Nos escalonadores analisados, nenhuma compensação para esse problema foi proposta.

- *Qualidade de experiência subtil*, apesar de existirem alguns algoritmos de escalonamentos na arquitetura *cloud* e *fog* que privilegiam a QoS para a priorização das tarefas, elas não se focalizam em otimizar a QoE do utilizador.
- *Supervisão para a preservação da qualidade de serviço*, os escalonadores analisados não fazem supervisão com objetivo de preservar adequadamente a qualidade de serviço. Isto é, o tempo máximo permitido para a obtenção de respostas não é considerado em algumas propostas e em outras, é considerado de forma inadequada.

Os vários escalonadores para a arquitetura *cloud* e paradigma *fog* existentes, permitem resolver vários problemas. No entanto, alguns aspetos podem ainda ser explorados a fim de se alcançar melhorias em relação às estratégias existentes (Barros *et al.*, 2020). Na subsecção 4.3.2 serão propostas algumas perspetivas de melhorias aos algoritmos de escalonamentos analisados na secção 4.2.

4.3.2. Perspetivas de melhoria dos algoritmos de escalonamento analisados

Segundo Barros *et al.* (2020b), alguns aspetos podem ainda ser explorados por forma a melhorar os algoritmos de escalonamento analisados:

- *Consciencialização do contexto no escalonamento de tarefas*, várias pesquisas asseguram que os escalonadores sensível ao contexto do utilizador são eficientes no aperfeiçoamento da qualidade de serviço e otimizam os custos, tanto do ponto de vista dos utilizadores como dos provedores de serviço.
- *Priorização de tarefas sensível ao contexto*, devem ser introduzidos modelos de escalonamento de aplicações onde as prioridades são definidas com base no contexto dos pedidos, onde os pedidos articulam informações do dispositivo e os contextos da aplicação. Considerando esses contextos associados a um pedido, o escalonador deve fazer o escalonamento com o objetivo de melhorar o desempenho.
- *Preservação da restrição de energia*, atendendo que muitos dispositivos possuem menor capacidade de processamento e autonomia reduzida da bateria, a restrição de energia deve ser levada em consideração durante a execução das tarefas. O nível da bateria do dispositivo do utilizador final associado a um pedido deve disponibilizar a um nível limite da bateria, para que o dispositivo solicitante seja preservado até ao fim da execução.

- *Conservação da força do sinal da rede*, a intensidade do sinal associada a um pedido deve assegurar a força mínima do sinal de forma a possibilitar ao utilizador solicitar recursos e obter respostas em tempo hábil.
- *Salvaguarda da qualidade de serviço*, as tarefas descarregadas na *fog* são heterogéneas em termos do contexto do dispositivo, do utilizador final e requisitos da aplicação, incluindo a QoS da aplicação. Assim, ao escalonar tarefas, é necessário ter em conta que o tempo necessário para a obtenção de resposta deve ser confinado dentro dos limites temporais previstos.
- *Redução do tempo médio de espera*, um mecanismo de compensação relativamente ao tempo médio de espera deve ser introduzido pelo escalonador de tarefas de forma a que o aumento do tempo de espera seja proporcionalmente menor em relação à chegada das tarefas.
- *Otimização da qualidade da experiência*, os algoritmos de escalonamentos devem também concentrar em otimizar tanto a QoE dos utilizadores finais como a QoS.

Um resumo dos algoritmos de escalonamento estudados, elucidando as suas principais vantagens, limitações e categoria é descrita na tabela 4.1.

Tabela 4.1: Síntese dos algoritmos de escalonamento analisados

(continua)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Gawali & Shinde, 2018)	*BATS+ BAR*	Facilita o aluguer de recurso na *cloud*.	Escalonamento e afetação de recursos são muito complexos	Básico
(Wang, Wang, & Cui, 2016)	*Energy-Aware Multi Job Scheduling*	Melhora a eficiência energética do servidor na afetação de dados	Possui limitações no escalonamento e otimização de tarefas;	Básico
(Bitam Zeadally, & Mellouk, 2018)	*Bees Life Algorithm* (BLA)	Maximiza o tempo de execução das tarefas	Existem poucos estudos relativamente a sua utilização em ambientes densamente distribuídas	Básico
(Aazam *et al.*, 2016)	*MEdia Fog Resource Estimation (MeFoRE)*	Estima os recursos da *fog* com base no QoE do utilizador; permite otimizar QoS e a gestão de recursos é descentralizada	Não respeita a expectativas do utilizador no acesso ao serviço nem no tempo de processamento; Velocidade de processamento também não satisfaz	Baseado em prioridade e orientado a QoS
(Woo, Jung, & Kim, 2017)	*Genetic Algorithm*	Permite reduzir o tempo de execução de tarefas	Possui pouco improviso o que dificulta o seu uso no paradigma *fog*	Básico
(Sarkar, Chantterjee, & Misra, 2018)	*Fog Computing in the Context of Internet of Things*	Permite melhorar o desempenho de aplicações sensíveis à latência	Possibilita grande tráfego de dados	Básico
(Deng *et al.*, 2016)	*Fog – Cloud Architecture*	Permite melhorar a mobilidade, a distribuição geográfica e a localização	A latência nas respostas podem levar à privação de serviços	Básico

(continuação)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Lawanyashri, Balusamy, & Subha, 2017)	*Fruity Fly Algorithm*	Permite melhorar a disponibilidade e escalabilidade; Enfatiza a redução de consumo de energia e consequentemente minimiza os custos	A afetação de recursos não é feita de forma equilibrada	Básico
(Intharawijitr, Iida, & Koga, 2016)	*Mathematical model of a Fog network*	Permite o escalonamento eficiente e ótimo das tarefas	Para o escalonamento são utilizadas as politicas: aleatória; menor latência e capacidade máxima disponível, o que torna o modelo muito complexo.	Baseado em prioridade e orientado a QoS
(Tiwary *et al.*, 2018)	*Non-Cooperative Game Model*	Melhora o tempo de resposta dos pedidos	Executa tarefas com tamanhos limitados	Básico
(Sheikhalishahi *et al.*, 2016)	*Multi-Capacity Aware Resource Scheduling*	Permite melhorar a utilização de recursos e prima pela eficiência energética	Utiliza menos CPU, o que faz com que o processamento das tarefas seja muito complexo	Sensível ao contexto
(Seddik & Hanzálek, 2017)	*Offline Multi-Level Scheduling*	Permite melhorar a utilização de recursos	A definição do tempo de processamento de uma tarefa é complexa	Baseado em prioridade e orientado a QoS
(Sahoo & Dehury, 2018)	*CPU Intensive Scheduling Data-Intensive Scheduling*	Permite melhorar a infraestrutura dos servidores e da rede	Pode ocasionar indisponibilidade de recurso e ter impacto negativo ao nível da QoS	Básico

(continuação)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Oueis, Strinati, & Barbarossa, 2015)	*Reduced Complexity Task Scheduling Algorithm for Fog*	Proporciona satisfação dos utilizadores em termos de baixa latência e consumo de energia	Algoritmos utilizados alcançam bons resultados apenas em infraestruturas de computação de baixa densidade	Baseado em prioridade e orientado a QoS
(Skarlat *et al.*, 2017)	*QoS-Aware Application Placement on Virtualized Fog Resources*	Observa as expectativas em termos de equilíbrio de recursos e tempo de processamento. Satisfaz o *status* de instâncias quanto à disponibilidade de recursos e velocidade de processamento;	Não respeita as expectativas em termos de acesso ao serviço; Não atende ao status de instância quanto à proximidade ou taxa de resposta	Baseado em prioridade e orientado a QoS
(Zhou *et al.*, 2015)	*QoE-Driven Video Cache Allocation Scheme for Mobile Cloud Server*	Observa as espectativas em termos do acesso ao serviço, satisfaz o *status* de instâncias quanto à proximidade ou taxa de resposta e disponibilidade de recursos; Permite a gestão descentralizada e prioriza tarefas no escalonamento	Não respeita as expetativas em termos de equilíbrio de recursos e tempo de processamento; Não atende ao *status* de instância quanto a velocidade de processamento	Sensível ao contexto

(continuação)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Cardellini *et al.*, 2015)	*Quality of Service Distributed Scheduler with QoS Awareness*	Em ambientes pequenos, melhora o desempenho da aplicação e aprimora o sistema com recursos de adaptação em tempo de execução	Em topologias *fog* complexas, com muitos operadores causam instabilidade o que diminui os níveis da disponibilidade da aplicação	Baseado em prioridade e orientado a QoS
(Mahmud *et al.*, 2016)	*QoE and Context-Aware Scheduling (QCASH)*	Observa as espectativas em termos de acesso ao serviço e tempo de processamento;	A gestão é centralizada, o que dificulta a sua implementação em ambientes densamente distribuídos como a *fog*	Sensível ao contexto
(Shi *et al.*, 2016)	*Adaptive Probabilistic Scheduler*	Consegue reduzir o consumo médio da energia em tarefa bem-sucedida e mantêm uma elevada taxa de execução da tarefa. Apresenta uma alta adaptabilidade tanto em rede fixa quanto em rede móvel.	Pode causar desequilíbrios na utilização de recursos	Sensível ao contexto
(Zhu *et al.*, 2015)	*Priority-Based Two-Phase Min–Min Scheduling Policy (PTMM)*	Pode ser aplicado em tarefas agrupadas por forma a minimizar o tempo de execução esperado;	Possui um mau desempenho caso os pedidos forem complexas o que pode resultar numa fraca QoE dos utilizadores	Baseado em prioridade e orientado a QoS

(continuação)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Qiu *et al.*, 2012)	*Energy-Aware Dynamic Task Scheduling*	Reduz o consumo de energia dos sistema ciberfísico, em comparação com o método de escalonamento do caminho crítico e o do escalonamento baseado no paralelismo	Abordagem ainda pouco estudada. Foi testada apenas nos dispositivos moveis Android. Há a necessidade de se fazer mais testes e combinar mais técnicas *Mobile Edge Computing*.	Básico
(Li *et al.*, 2017)	*Cooperative-Based Model for Smart-Sensing Tasks* (CMST)	Experiências extensivas mostram que o CMST melhora a cobertura e qualidade de dados e permite reduzir os custos.	Possui um bom desempenho, apenas em casos específicos como recolhas de dados com objetivo de alimentar plataformas.	Básico
(Shojafar *et al.*, 2015)	*FUzzy GEnetic Task Scheduling* (FUGE)	Resultados das experimentações mostram a eficiência dessa abordagem em termos de tempo, custo de execução e grau médio de desequilíbrio.	Não considera o consumo de energia e a migração de maquinas virtuais; Não é eficiente em termos energéticos; No escalonamento considera apenas o tempo de execução da tarefa (*makespan*)	Básico

(continuação)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Gill, Garragha n, & Buyya, 2019)	ROUTER	Permite a redução na utilização da largura de banda da rede, no tempo de resposta, da latência e no consumo de energia.	Foi testada em *Smart Home* de pequena escala; Não atende aos parâmetros como escalabilidade, confiabilidade, custo e disponibilidade.	Baseado em prioridade e orientado a QoS
Mahmud *et al*. (2019)	*QoE-aware application placement policy*	Observa as espectativas em termos do acesso ao serviço, satisfaz o *status* de instâncias quanto à proximidade ou taxa de resposta e disponibilidade de recursos; Permite a gestão descentralizada e prioriza tarefas no escalonamento	A priorização no escalonamento leva em consideração apenas o requisito de QoE	Sensível ao contexto
(Yang *et al.*, 2018)	*Delay Energy Balanced Task Scheduling (DEBTS)*	Minimiza o consumo de energia, reduz o atraso médio do serviço e de entrega de dados na rede (*jitter*) no escalonamento de tarefas;	Apenas foi testado no paradigma *fog* simples, há a necessidade de ser testado em ambientes *fog* mais complexas.	Básico
(Fan *et al.*, 2017)	*Deadline-Aware Task Scheduling Mechanism*	Melhora o desempenho do sistema quando comparado com os algoritmos *Min-Min* e *First Come First Served*.	Em ambientes densamente distribuídos como a *fog*, o desempenho se degrada e há a necessidade de se considerar algoritmos distribuídos e leves para a otimização.	Básico

(conclusão)

Autores	Algoritmos	Vantagens	Limitações	Categoria
(Stavrinides & Karatza, 2019)	*Hybrid Fog and Cloud-Aware Heuristic (Hybrid-EDF)*	Proporciona em média uma diminuição em relação a falhas na entrega porque utiliza recursos da *cloud*. sobretudo quando esta for composta por maquinas virtuais *on-demand multi-tenant*.	Quando a camada da *cloud* for constituída por maquinas virtuais em *hosts* reservados e dedicados tem um alto custo.	Básico
(Zhou *et al.*, 2017)	*mcloud: Context-Aware Offloading Framework*	Disponibiliza decisões de descarregamento com base no contexto atual dos dispositivos o que permite diminuir o custo do tempo de execução e o consumo de energia.	Por forma a ser mais eficiente e confiável, há necessidade de ampliar a estrutura por forma a que os recursos da *cloud* tenham a capacidade de intercomunicar com base nas mudanças de contexto	Sensível ao contexto
(Ghouma & Jaseemuddi, 2015)	*Context Aware Cloud System*	Reduz o custo de execução das tarefas e, ao mesmo tempo, satisfaz a duração da execução definida.	A elevada latência associada a *cloud* impossibilita a sua utilização em aplicações que requerem baixas latência.	Sensível ao contexto

Fonte: o autor

5. MODELO E ARQUITETURA PROPOSTOS

Neste quinto capítulo, descrevemos os contextos previstos, ilustramos e discutimos o nosso modelo e proposta de arquitetura e apresentamos um exemplo ilustrativo que nos ajudará a visualizar e a entender as funcionalidades da nossa proposta.

5.1. Contextos previstos e assunções

Os dispositivos móveis atuais executam vários formatos de dados e aplicações de computação intensiva. Devido à escassez de recursos, autonomia reduzida da bateria e menor capacidade de processamento, surgiram conceitos como *code offloading*[14] e *cloud computing*. Esses conceitos são utilizados para resolver as limitações da computação móvel, através da disponibilização de outros fornecedores de recursos, que não os próprios dispositivos móveis para alojar a execução de aplicações (Barros *et al.,* 2020c).

Para muitas aplicações, nas quais se incluem as de linguagem natural em tempo real, processamento de imagem, reconhecimento de voz e escrita, entre outros, que precisam manter requisitos rigorosos de tempo de execução e uma alta taxa de execução para os utilizadores móveis, a *cloud*, devido à centralização da execução, não tem sido a alternativa mais adequada, surgindo a necessidade da implementação de estratégias descentralizadas, mais próximas dos utilizadores ou dos lugares, onde os dados são produzidos. Neste quesito, a *fog computing* surge como um dos paradigmas mais promissores.

Segundo Wang *et al.* (2017), ao contrário da arquitetura *cloud*, que geralmente é composta por duas camadas, na qual um utilizador interage apenas e diretamente com a *cloud*, o paradigma *fog* é composta por, no mínimo, três camadas, isto é, *edge nodes* adicionais (por exemplo, estações base móveis, *routers* e *switches*, pontos de acessos, entre outros) são introduzidos entre o utilizador e a *cloud*.

Nesta tese, concentramo-nos no problema de escalonar pedidos heterogéneos na *fog*. Se por insuficiência de recursos, os pedidos não puderem ser escalonados, serão então encaminhadas para serem executadas na *cloud*, conforme ilustrado na figura 5.1.

Devido à sua heterogeneidade, é muito difícil explorar os contextos no escalonamento de aplicações. Para resolver este problema, Han, Kamber, & Jian (2012), propuseram uma solução que visa resolver a heterogeneidade de contexto, através do processamento de vários parâmetros do contexto em um intervalo normalizado, utilizando a normalização *Min-Max*, que utilizamos na nossa proposta. Cada pedido é priorizado com base nos seus valores de parâmetros do contexto. O escalonador resolve o problema de Otimização de Programação Não Linear Multiobjetivo da afetação de aplicações às máquinas virtuais com o objetivo de

[14]Tradução livre do autor: “descarregamento de código”.

minimizar o tempo de execução e garantir a execução prioritária dos pedidos com maiores prioridades.

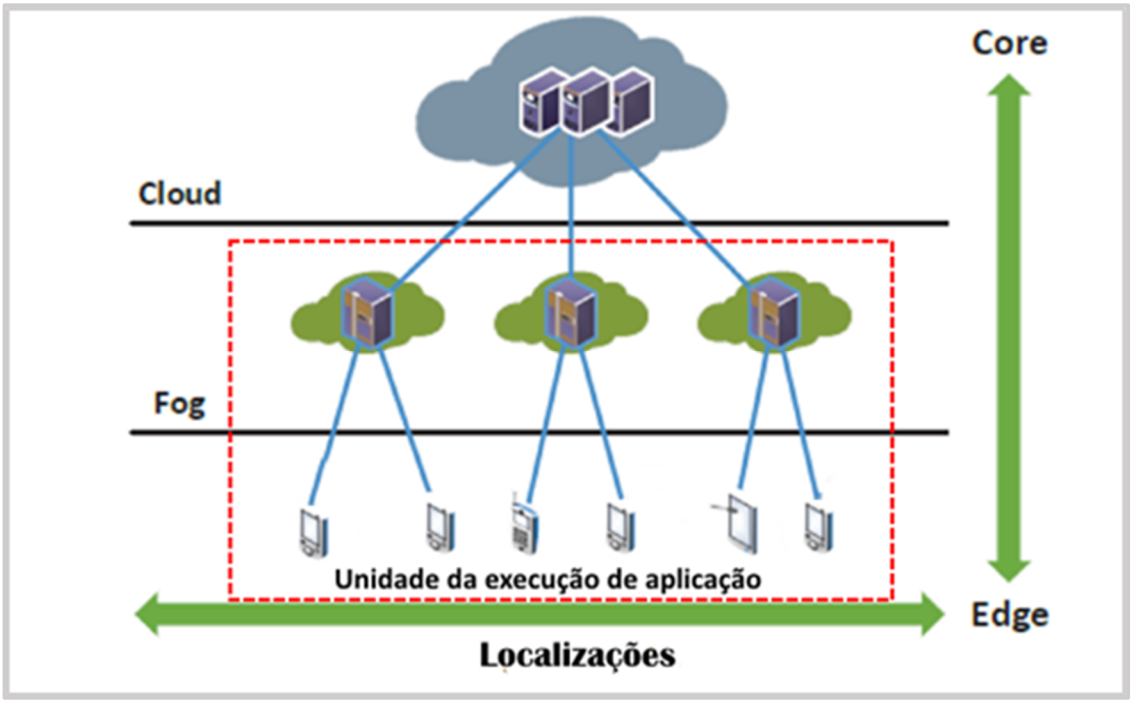


Figura 5.1: Ambiente de execução de aplicações no paradigma *fog*.
Fonte: adaptado de Swaroop (2019).

Assumimos que uma técnica de *code offloading* adequado (por exemplo, *ExTrade* definido em Khoda *et al.* (2016), *ENDA* estabelecido em Li *et al.* (2013), *ThinkAir* indicado em Kosta *et al.* (2012), *COMET* apresentado em Gordon *et al.* (2012), *CloneCloud* indicado em Chun *et al.* (2011), *MAUI* definido em Cuervo e*t al.* (2010), entre outros) esteja a ser executada nos dispositivos móveis a fim de tomar a melhor decisão em relação ao descarregamento ou não de códigos e em quais nós da *fog* (Berg, Dürr, & Rothermel, 2014).

Consideramos que um pedido de execução descarregado inclui o atraso máximo permitido (ou seja, exigência de QoS) para executar a aplicação, o nível de bateria do dispositivo e os valores da força do sinal da rede. Também assumimos, tal como em Deng *et al.* (2016), que a *fog* disponibiliza capacidades computacionais muito maiores que os dispositivos móveis e deve ter a capacidade de extrair os contextos associados aos pedidos e tomar a decisão de escalonamento em conformidade.

Na secção 2.4 do Capítulo 2, Musumba & Nyongesa (2013), definiram como os principais contextos que podem ser explorados em qualquer ambiente de computação móvel: a ligação

a rede; processadores disponíveis; nível de bateria, a localização; a largura de banda da rede; o tráfego na rede; as máquinas virtuais alugadas e requisitos de QoS da aplicação.

Como no nosso domínio do problema consideramos apenas os contextos do dispositivo do utilizador final, da rede e os do tempo de execução da aplicação, os contextos dos provedores de serviços (como por exemplo as máquinas virtuais alugadas) foram ignorados. Também, depois de descarregado a tarefa na *fog*, torna-se desnecessário considerar os processadores no dispositivo móvel. A localização do dispositivo também não afetará a tarefa do escalonamento, bem como o tráfego na rede e a largura de banda que é igual para todos os utilizadores.

Com base nestas considerações, nesta tese previmos três parâmetros de contexto nos dois seguintes componentes ambientais:

- *Componente da rede e comunicação:*
    - Contexto do dispositivo do utilizador: Nível da bateria;
    - Contexto da rede: Relação sinal-interferência-ruído da rede (SIN);
- *Componente de dados ou aplicação:*
    - Contexto de tempo de execução da aplicação: QoS da aplicação.

Também assumimos que, se um pedido descarregado possuir um sinal fraco da rede, significa que o dispositivo correspondente tem alta probabilidade de sair da cobertura do BS/AP, devido à sua mobilidade. O nível de bateria baixo corresponde à criticidade energética do dispositivo. Caso os valores de nível do sinal e de bateria de um pedido satisfizerem, respetivamente, a intensidade mínima do sinal da rede e do nível de bateria, para que a aplicação mantenha o tempo de resposta baixa (requisito de QoS) e prolongar a vida útil do dispositivo, até ao final da execução, optamos por priorizar pedidos provenientes de dispositivos com valores críticos. Caso contrário, o sistema simplesmente rejeita o pedido ou o encaminha para a *cloud*.

5.2. Modelo proposto

Os nós da *fog*, com a nossa proposta ativada, é constituída por várias unidades de trabalho:

- *Unidade de recuperação de informações de contexto* - compreende uma arquitetura, conforme definido em La e Kim (2010). Ela recupera as informações de contexto (C_i) de cada pedido ($r \in R$). As informações de contexto recuperadas são encaminhadas para a *unidade de priorização de tarefas sensível ao contexto*;
- *Unidade de priorização de tarefas sensível ao contexto* - estima o valor da prioridade de contexto (P_r) para cada pedido individual $r \in R$ e o encaminha para a *unidade de QoE e escalonamento sensível ao contexto*.
- *Unidade de QoE e escalonamento sensível ao contexto* - escalona as tarefas para serem executadas nas *máquinas virtuais (MVs) em servidores* de forma que a QoE do utilizador final seja otimizada.

A figura 5.2 ilustra o modelo proposto e as diferentes unidades de trabalho que o compõem.

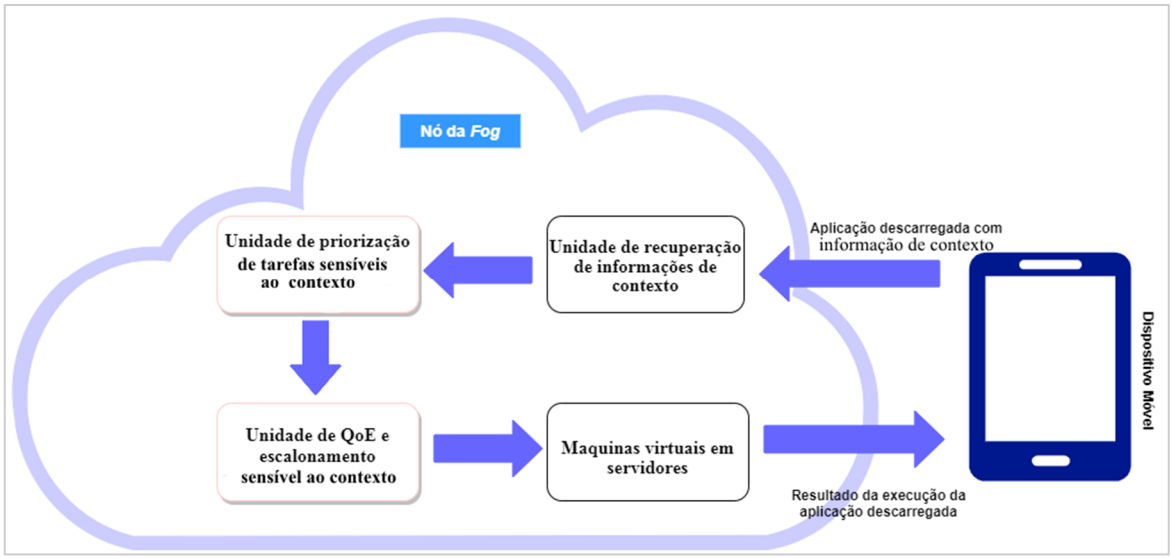


Figura 5.2: Trocas de informações entre as diferentes unidades do modelo proposto.
Fonte: o autor.

As notações e definições relevantes para a priorização de tarefas sensível ao contexto e as utilizadas na unidade de QoE e escalonamento sensível ao contexto estão listadas na tabela 5.1.

Tabela 5.1: Notações e definições da arquitetura do modelo proposto.

Símbolo	Definição
C	Conjunto de todos os parâmetros de contexto
α_i	Quantidade de diferentes tipos de níveis abstratos de um parâmetro de contexto, $C_i \in C$
γ_i	Diferença lógica entre dois níveis abstratos $C_i \in C$
η_i	Intervalo normalizado de valores $C_i \in C$
L_i	Conjunto de todos os níveis em concretos de um $C_i \in C$ no instante *t*
θ_i	Conjunto de valores tendenciosos de um $C_i \in C$
F	Conjunto de todas as possíveis combinações de cada um dos elementos L_i
M	Multiconjunto de valores mínimos $\forall \mu \in F$
R	Conjunto de pedido de execução num determinado instante *t*
V	Conjunto de máquinas virtuais disponíveis num determinado instante *t*
β	Conjunto dos coeficientes de *Multiple Linear Regression*
P_r	Prioridade de qualquer pedido $r \in R$
$\psi_{r,v}$	Tempo de execução de um pedido $r \in R$ em uma máquina virtual $v \in V$
I_r	Quantidade de interrupções de um pedido $r \in R$
r	Pedido
Q_r	Tempo de espera na fila de um pedido *r*
T_r	Representa o valor da QoS da aplicação. Isto é, ou o tempo máximo permitido para a obtenção de resposta
z	Índice dos níveis abstratos de um parâmetro de contexto, $C_i \in C$

Fonte: o autor.

Assumimos que algumas MVs já estejam criadas com diferentes configurações o que permite minimizar as sobrecargas relativas aos processos de criação e eliminação de MVs, conforme referido em Das *et al.* (2013), e Skarlat *et al.* (2017). A provisão ótima de MVs para os pedidos e as suas afetações energeticamente eficientes estão fora do escopo desta tese. Foram, no entanto, tratadas em Li *et al.* (2017) e Yang *et al.* (2018).

Nesta tese, abordamos o problema da otimização da qualidade de experiência dos utilizadores e escalonamento de aplicações sensível ao contexto no paradigma *fog*.

5.3. Arquitetura do modelo proposto

O fato dos parâmetros de contextos associados aos pedidos serem heterogéneos, torna difícil o processo de exploração das informações do contexto no escalonamento de aplicações. Para resolver este problema, em Han, Kember & Jian (2012), foi proposto um resolvedor de heterogeneidade de contexto, que processa vários parâmetros de contexto, num intervalo normalizado, através da normalização *Min-Max*, que utilizamos na nossa proposta, onde cada pedido é priorizado com base nos seus valores de parâmetros de contexto.

O escalonador de prioridade resolve o problema de otimização de programação não linear multiobjetivo que consiste em afetar aplicações às máquinas virtuais por forma a minimizar o tempo de execução (ou seja, otimiza a QoE) e a garantir a execução prioritária dos pedidos com maior prioridade.

Nas subsecções seguintes são definidas a arquitetura do modelo proposto, a começar pela unidade de priorização das tarefas sensível ao contexto.

5.3.1. Unidade de priorização de tarefas sensíveis ao contexto

A *unidade de priorização de tarefas sensível ao contexto* proposta é constituída por um *repositório de contexto* e uma *unidade de previsão de contexto*.

O *repositório de contexto* armazena as informações de contexto das tarefas atuais e anteriormente recebidas. Desta forma, beneficia de todos os pré-requisitos da normalização *Min-Max*. Associado ao *repositório de contexto*, a *unidade de previsão de contexto*, explora a informação de contexto num determinado instante de tempo e alimenta a *tabela de previsão*. Assim, conseguimos eliminar a heterogeneidade das informações de contexto na alimentação da *tabela de previsão.*

A tabela de previsão disponibiliza um conjunto de dados para a análise da MLR que visa definir a prioridade de contexto dos pedidos atuais. A figura 5.3 ilustra arquitetura da unidade de priorização de tarefas sensível ao contexto do modelo proposto.

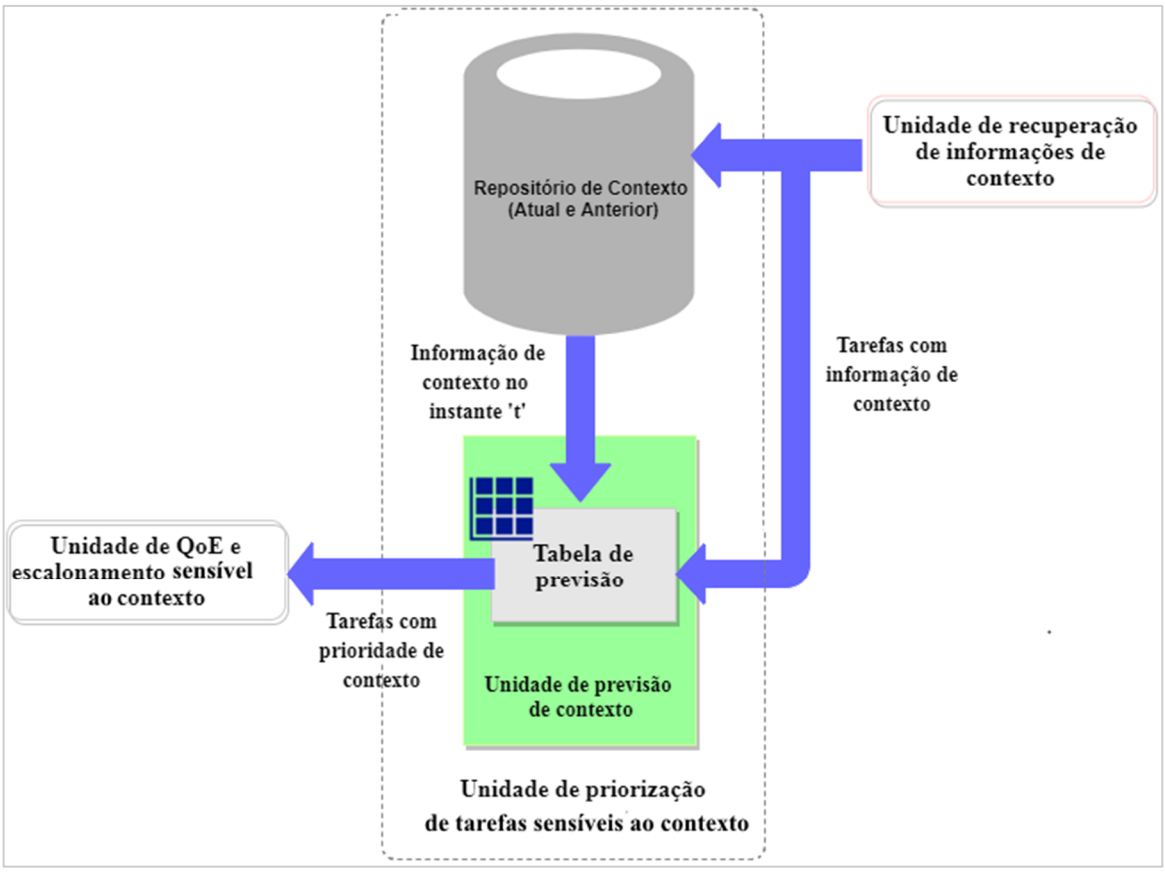


Figura 5.3: Arquitetura da unidade de priorização de tarefas do modelo proposto.
Fonte: o autor.

5.3.2. Criação da tabela de previsão de prioridade de contexto

Para a resolução de problemas relacionados com a heterogeneidade de informações do contexto, com base na normalização *Min-Max*, foi concebido um modelo que alimenta a tabela de previsão de prioridade de contexto. O repositório de contexto disponibiliza informações do contexto de tarefas previamente recebidas num determinado instante t. $\forall C_i \in C$, todas as informações do contexto são normalizadas em relação aos extremos (valores máximo e mínimo). Assim, os valores ficam concentrados num intervalo entre 0 e 1. Esta abordagem permite minimizar a heterogeneidade dos diferentes parâmetros do contexto em termos de valores e unidades.

Definimos o intervalo normalizado, η_i, para um parâmetro de contexto, $C_i \in C$, como:

$$\boldsymbol{\eta_i = \frac{\max(C_i) - \min(C_i)}{\max(C_i)}}. \quad (5.1)$$

Dentro deste intervalo normalizado de um $C_i \in C$, assumimos que existem α_i níveis abstratos. Estes níveis abstratos representam a medição qualitativa do parâmetro de contexto

correspondente. Permitem que as variações de valores de um determinado parâmetro de contexto possam ser classificadas internamente. As diferenças lógicas entre dois níveis abstratos consecutivos, γ_i, de um contexto $C_i \in C$ são definidas como:

$$\boldsymbol{\gamma_i = \frac{\eta_i}{\alpha_i}.} \tag{5.2}$$

A representação numérica desta abstração compreende um conjunto de níveis concretos, L_i, para qualquer $C_i \in C$. Esta abordagem transforma a medição qualitativa em quantitativa, compatível para cálculos posteriores. O L_i é definido como:

$$\boldsymbol{L_i = \left\{x: x = \frac{\min(C_i)}{\max(C_i)} + z\gamma_i\right\}, \forall C_i \in C.} \tag{5.3}$$

Onde $z = 0,1,2,\ldots,(\alpha_i - 1)$.

Utilizando o conjunto de produtos cartesianos, são criados os conjuntos combinatórios de todos os níveis de contexto a partir de diferentes parâmetros de contexto. Este produto cartesiano é formulado conforme indicado na equação 5.4.

$$\mathbf{F} = \prod_{\mathbf{i=1}}^{|\mathbf{C}|} \mathbf{L_i}\,. \tag{5.4}$$

Como resultado, todas as possíveis combinações de diferentes parâmetros do contexto de um pedido individual ($r \in R$) são determinadas.

Também é criado um multiconjunto *M* com os valores mínimos de cada combinação de *F*, definido conforme a equação 5.5.

$$\mathbf{M = \{q : q = \min(\mu)\},\ \forall \mu \in F.} \tag{5.5}$$

Este multiconjunto *M* identifica o estrangulamento de todas as possíveis combinações do contexto de um pedido $r \in R$. Este parâmetro de estrangulamento influência grandemente a priorização das combinações simbólicas.

Para além dos parâmetros de estrangulamento, os valores de enviesamento, associados aos diferentes níveis dos vários parâmetros de contexto, que são utilizados na priorização da combinação simbólica, também influenciam a priorização.

Este valor de enviesamento permite enfatizar os outros parâmetros. Basicamente, é um mapeamento *um-para-um* entre os elementos de L_i e o conjunto enviesado, θ_i para um determinado parâmetro de contexto, $C_i \in C$.

Seja, $\theta_{i,j}$ refere ao valor mínimo de enviesamento de C_i, no seu *j-ésimo* nível associado a uma combinação candidata em *F*. Para mapear, θ_i para L_i, $\theta_{i,0}$, presume-se, $\forall C_i \in C$.

$\forall C_i \in C, \theta_{i,j}$ deve satisfazer a seguinte condição:

$$\boldsymbol{\theta_{i,j} * max(q \in M) < \theta_{i,j+1} * min(q \in M)}. \quad (5.6)$$

As prioridades destas combinações são definidas de forma que a informação do contexto de qualquer pedido possa ser mapeada sobre ela mesma a fim de prever a prioridade desse pedido.

O cálculo da prioridade é feito utilizando os valores relevantes de enviesamento $\forall C_i \in C$ e o seu parâmetro de contexto de estrangulamento. Assumimos que $\delta_i\ (\mu_k)$ define a prioridade do contexto $\mu_k \in F$ enviesado em $C_i \in C$. $\delta_i\ (\mu_k)$ é representado conforme a equação 5.7.

$$\boldsymbol{\delta_i\ (\mu_k) = \frac{\theta_{ij} * q_k}{\sum q}, \forall q \in M}. \quad (5.7)$$

Onde, $0 \leq k\ \leq (|F| - 1)$ e $q_k \in M$ está associada à μ_k.

A prioridade estimada, $\hat{\delta}_i(\mu_k)$ de μ_k é calculada através da equação 5.8.

$$\boldsymbol{\widehat{\delta}_i(\mu_k) = \sum_{C_i \in C} \delta_i\ (\mu_k)}. \quad (5.8)$$

μ_k e $\hat{\delta}(\mu_k)$, $\forall \mu_k \in F$ vão alimentar a tabela de previsão com um conjunto de dados. Estes dados da tabela de previsão permitem prever a prioridade de qualquer pedido em fila de espera.

5.3.3. Previsão da prioridade do contexto

A MLR é um dos métodos estatísticos multivariados cuja a principal preocupação consiste em estabelecer as relações entre várias variáveis independentes ou preditoras e uma variável dependente ou critério. Ao identificar como estas múltiplas variáveis independentes se relacionam com a variável dependente, as informações sobre as variáveis independentes podem ser utilizadas para fazer previsões precisas e poderosas (Quinn & Keough, 2002).

Consideremos, *m* observações de um conjunto de *p* número do variável preditor *Xs* e um variável critério *Y* associado a elas. O modelo de MLR, que geralmente se ajusta a este cenário, é definido conforme a equação 5.9.

$$\boldsymbol{y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_j x_{ij} + \cdots + \beta_p x_{ip} + \epsilon_i,} \quad (5.9)$$

onde, para a *i-ésima* observação, $y_i = Y$ e $x_{ji} = X_j$, $\forall X_j \in X$. O coeficiente do MLR, β_0, é a intercetação da população e β_j, é a variação de *Y* em uma unidade de variação em X_j, $\forall X_j \in X$ *e* $1 \leq j \leq p$, mantendo as outras variáveis independentes constantes, ϵ_i é o erro aleatório ou inexplicável associado à *i-ésima* observação. Os valores estimados dos coeficientes da MLR são calculados através dos valores conhecidos da equação (5.8) de cada observação, e resolvidos algebricamente. Estimando $\beta = \{\beta_0, \beta_1, \ldots, \beta_p\}$, $\forall \beta_0 \in \beta$ e σ_ϵ^2 (erro da variância), a reta de regressão ajustada, que prevê *Y* para qualquer observação desconhecida, é expressa conforme a equação 5.10:

$$\boldsymbol{\hat{y}_i = b_0 + b_1 x_{i1} + \cdots + b_j x_{ij} + \cdots + b_p x_{ip}} \quad (5.10)$$

onde, $\hat{y}_i$ é o valor previsto para qualquer observação desconhecida, b_j é a estimativa da amostra de $\beta_j \, \forall \beta_j \in \beta$.

Ao invés de calcular as regressões para cada variável preditora individualmente, a MLR utiliza informações de todas as variáveis independentes simultaneamente para prever uma única variável critério. Como resultado, ela é naturalmente mais rápida do que os outros métodos de análise multivariada (Mertler & Reinhart, 2016).

Fazemos o mapeamento da tabela de previsão de contexto para o modelo MLR, considerando cada *tuplo* da tabela de previsão como uma observação de um conjunto de dados da MLR em que, o conjunto de variáveis independentes, $X = C_i$, $\forall C_i \in C$ de cada combinação $\mu \in F$, a variável dependente, $Y = \hat{\delta}_i(\mu_k)$ é a prioridade estimada de qualquer pedido desconhecido, $P_r = \hat{y}_i$.

Para qualquer pedido $r \in R$, quanto menor for o valor P_r, maior será a prioridade. A MLR é um método de análise multivariada, amplamente utilizado para prever uma variável dependente, em função de várias variáveis independentes, sendo ambas de natureza quantitativa (Mertler & Reinhart, 2016). A tabela de previsão de prioridade de contexto é criada através dos valores quantitativas de $\forall C_i \in C$ e dos parâmetros de contexto que são

independentes entre si. Neste sentido, é mais adequada a utilização da MLR para prever a prioridade de qualquer pedido desconhecido da tabela de previsão de prioridade do contexto. Após a definição da prioridade do contexto, as informações do contexto desta tarefa são armazenadas no repositório de contexto.

Na fase de priorização do contexto, um valor de prioridade é atribuído às tarefas. O gráfico 5.1 ilustra a sequência de execução das tarefas em relação à definição de prioridades com base nas informações do contexto. As tarefas, com parâmetros de contexto mais baixos, apresentam menores prioridades de contexto. Quanto menor for o valor da prioridade do contexto, maior será a prioridade da tarefa.

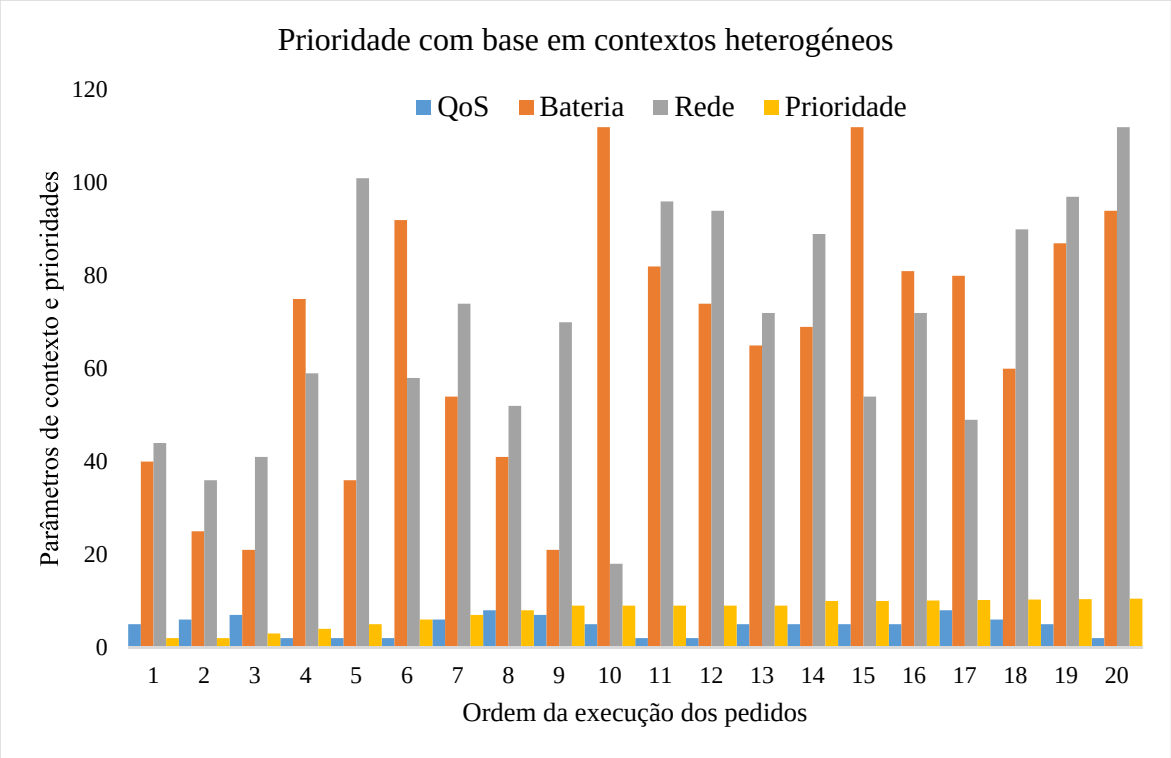


Gráfico 5.1: Prioridade em função da heterogeneidade dos contextos dos pedidos
Fonte: o autor.

5.4. Otimização do escalonamento das aplicações

A *unidade de QoE e escalonamento sensível ao contexto* do modelo proposto escalona as tarefas prioritárias para serem executadas em máquinas virtuais na *fog* após um intervalo de escalonamento (IE), τ.

De modo a otimizar a QoE dos utilizadores, o escalonador proposto explora a prioridade de contexto (P_r) de um pedido $r \in R$ e a sua duração estimada do tempo de execução ($T_{r,v}$) para determinar o escalonamento desse pedido $r \in R$ em uma máquina virtual, $v \in V$. Se em

virtude da quantidade limitada de MVs e à baixa prioridade, um pedido $r \in R$ não poder ser escalonada num intervalo de escalonamento, ela deverá ser escalonada nos próximos intervalos. Na nossa proposta, a execução de tarefa com menor prioridade é interrompida em virtude da chegada de tarefa com maior prioridade. Considerando que essa preempção pode propiciar a espera por tempo indeterminado para a execução de um determinado pedido $r \in R$, exploramos também o número de intervalos de escalonamento (I_r), em que um pedido de escalonamento numa MV é adiado desde a sua chegada, com o objetivo de evitar a condição de *starvation*.

Com vista a otimizar a QoE dos utilizadores foi utilizada a técnica de otimização de programação não linear multiobjetivo para escalonar os pedidos.

Segundo Miettinen (1998), a otimização não linear multiobjetivo é geralmente aplicado aos problemas de otimização onde existem mais de uma função objetivo não lineares a serem otimizada simultaneamente. Geralmente, é utilizado quando se pretende minimizar, ou maximizar, um conjunto de objetivos que geralmente são conflituantes, ou seja, objetivos que não possuem a mesma solução ótima sob determinadas restrições.

Ela possui alguns elementos que o caraterizam como: conjunto de funções objetivos não lineares que devem ser otimizadas; variáveis de decisão que constituem o domínio no qual cada função objetivo deve operar; restrições que delimitam o espaço de pesquisa.

Nas subsecções seguintes, definiremos as funções objetivos e as restrições para o escalonamento otimizado de aplicações.

5.4.1. Definição de função objetivo

Em programação não linear multiobjectivo a função objetivo é constituída por dois ou mais funções objetivo representadas por funções não lineares e composta por uma ou mais variáveis de projeto que se quer minimizar ou maximizar.

Um dos objetivos desta tese consiste em escalonar pedidos, $r \in R$ em uma máquina virtual, $v \in V$, com o objetivo de otimizar a QoE para todos os pedidos num determinado intervalo de escalonamento.

Assumindo que a execução de todas os pedidos, $r \in R$ não são preemptivos e que todas as MVs, $v \in V$ estão criadas na *fog*, a função objetivo é definida conforme a equação 5.11.

$$\boldsymbol{min}_{r,v} \sum_{r \in R} \sum_{v \in V} \frac{\boldsymbol{P}_r * \boldsymbol{\psi}_{r,v}}{\boldsymbol{I}_r} \tag{5.11}$$

E está sujeita a algumas restrições, (inequações (5.12) à (5.17)).

A equação (5.11) indica que a QoE de todos os pedidos de aplicações dos utilizadores podem ser otimizadas através da minimização da soma dos seus tempos de execução. Ela também leva em consideração a execução prioritária das tarefas com maiores prioridades através da minimização da soma das prioridades de todas os pedidos, dado que, quanto menor for o resultado, maior será a prioridade obtida. Além disso, a soma dos valores inversos do (I_r), $\forall r \in R$ mostra que os pedidos, em que foram adiados os seus escalonamentos num determinado intervalo, terão maior prioridade para serem escalonadas nos atuais intervalos, atenuando assim a situação de *starvation*.

5.4.2. Definição das restrições

As restrições em otimização não linear multiobjetivo, são representadas através de funções de igualdade ou desigualdade sobre as variáveis de projeto e descrevem situações de projeto consideradas como não desejáveis (Miettinen, 1998).

Para o escalonamento otimizado definimos as seguintes restrições:

- Restrição de capacidade: a restrição de capacidade é representada como a inequação 5.12.

$$\sum_{v \in V} \boldsymbol{\eta}_v \leq \boldsymbol{\eta} \tag{5.12}$$

  onde, η_v é a capacidade da máquina virtual e η é a capacidade total do nó da *fog*.

- Restrição de afetação de MV: a restrição de afetação de MV é representada conforme a inequação 5.13.

$$\sum_{v \in V} \boldsymbol{K}_{v,r} \leq \boldsymbol{1}, \forall \boldsymbol{v} \in \boldsymbol{V}, \forall \boldsymbol{r} \in \boldsymbol{R} \tag{5.13}$$

  onde, $K_{v,r}$ é uma variável booleana, que é igual a um, se uma MV $v \in V$ for afetado a um pedido $r \in R$; caso contrário é zero.

- Restrição de escalonamento dos pedidos: a restrição de escalonamento dos pedidos é escrita como a inequação 5.14.

$$\sum_{r \in R} \chi_{r,v} \leq \mathbf{1}, \forall r \in R, \forall v \in V. \quad (5.14)$$

onde, $\chi_{r,v}$ é uma variável booleana, é igual a um, se um pedido $r \in R$ estiver escalonada em uma máquina virtual $v \in V$; caso contrário, é zero.

- Restrição de QoS: a restrição de QoS é representada conforme a inequação 5.15.

$$\psi_{r,v} + Q_r \leq T_r, \forall r \in R. \quad (5.15)$$

onde, $\psi_{r,v}$ representa o tempo necessário para executar um pedido $r \in R$ em uma máquina virtual $v \in V$, Q_r é o tempo de espera na fila do pedido *r*. e T_r corresponde à QoS da aplicação ou o tempo máximo admissível para a obtenção de resposta.

- Restrição de consumo de energia: a restrição de consumo de energia é representada conforme a inequação 5.16:

$$B_r \geq B_{th}. \quad (5.16)$$

onde, B_r representa o nível de bateria do dispositivo do utilizador final associado a um pedido, $r \in R$ e B_{th} indica o nível mínimo de bateria, para que o dispositivo requisitante se permaneça ligado até ao final da execução.

- Restrição da qualidade do sinal: a restrição da qualidade do sinal é escrita conforme a inequação 5.17:

$$SIN_r \geq SIN_{th}, \quad (5.17)$$

onde, SIN_r representa a intensidade do sinal associado a um pedido $r \in R$ e SIN_{th} indica a intensidade mínima do sinal necessário para a submissão de um pedido do dispositivo do utilizador final.

5.5. Definição dos parâmetros do sistema

Comparado com os outros parâmetros de contextos, o requisito de QoS da aplicação tem maior impacto no aperfeiçoamento da QoE dos utilizadores finais. Por isso, definimos os valores de enviesamento, θ associados principalmente a este parâmetro.

A MLR, conforme descrito na subsecção 5.3.3, explora a relação entre uma variável dependente e um conjunto de variáveis independentes, em que os valores de todas as variáveis são definidos para *m* casos (Berger, Tirari, & Tille, 2003). Quanto maior o número de observações, com mais exatidão se consegue prever a situação. Portanto, é sensato aumentar as diferentes categorias dos níveis abstratos $\alpha_i, \forall\, C_i \in C$.

O desempenho da solução proposta é influenciado grandemente pelo intervalo de escalonamento, τ. Atendendo que o tempo médio de execução de diferentes pedidos $r \in R_t$ nas diferentes MVs $v \in V_t$ são conhecidos a partir das experiências anteriores. Podemos calcular a previsão do escalonamento estático (τ_s) da seguinte forma:

$$\boldsymbol{\tau_s = \frac{1}{|R_t|} \sum_{r \in R_t} \sum_{v \in V_t} \psi_{r,v}.} \tag{5.18}$$

O valor τ estático nem sempre permite uma boa utilização dos recursos. Um valor de τ alto reduz a sobrecarga do escalonamento, contudo, poderá não conseguir manter a exigência de QoS da aplicação do utilizador nem garantir uma melhor utilização dos recursos computacionais. Do mesmo modo, um valor de τ muito baixo pode provocar uma grande sobrecarga no escalonamento, assim como poderá possibilitar a existência de alguns intervalos de escalonamento, sem a chegada de pedidos. Portanto, um valor τ dinâmico permite aumentar o desempenho da nossa proposta, assumindo que a taxa média de chegada dos pedidos varia muito ao longo do tempo. No entanto, na prática, nem a sequência da chegada dos pedidos, nem os respetivos tempos de execução nas MVs obedecem o intervalo de escalonamento estático (τ_s), que é calculada como média aritmética dos valores das experiências anteriores. Por isso, visando aumentar o desempenho do modelo proposto optamos, pelo cálculo do intervalo de escalonamento dinâmico (τ_d), que utiliza a fórmula do *Exponentially Weighted Moving Average* (EWMA) (Wold, 1994), conforme a equação 5.19:

$$\boldsymbol{\hat{\tau}_{d(m)} = (1 - \omega)\hat{\tau}_{d(m-1)} + \omega\tau_{d(m)}} \tag{5.19}$$

onde, ω é a constante de ponderação, ou o fator de alisamento, onde $0 < \omega < 1$ e $\tau_{d(m)}$, é a média aritmética dos tempos de execução dos pedidos em todas as MVs no intervalo de escalonamento *m-éssimo*. Conforme a equação 5.20.

$$\boldsymbol{\tau_{d(m)} = \frac{1}{|S|} \sum_{s \in S} \frac{1}{|V_s|} \left(\sum_{v \in S} \frac{N_{v,r}}{\phi_v} \right).} \tag{5.20}$$

onde, S e V_s significam respetivamente o conjunto de servidores e de MVs em cada servidor. $N_{v,r}$ representa o número de instruções de um pedido $r \in R$ executada numa MV $v \in V_s$ com velocidade de processamento ϕ_v.

Na secção seguinte, é apresentado um exemplo ilustrativo que ajuda a visualizar as funcionalidades da nossa proposta de priorização de contexto e da unidade de aprimoramento da QoE numa dimensão concreta.

5.6. Exemplo ilustrativo

Nesta secção, propomos apresentar um exemplo ilustrativo que nos ajudará a visualizar as funcionalidades do modelo proposto. Nele, três parâmetros de contexto, $C_i \in C$ foram considerados: Bateria ($C_{bateria}$), Intensidade de Sinal da rede (C_{Sin}) e QoS da Aplicação (C_{QoS}).

5.6.1. Criação da tabela de contexto

Seja o valor de níveis abstratos para diferentes contextos, onde:

$$\boldsymbol{\alpha_{\text{Bateria}} = 3, \alpha_{\text{Sin}} = 2, \alpha_{QoS} = 4.}$$

Num determinado período de tempo *t*, o repositório de contexto disponibiliza os valores *Max-Min* e as equações (5.1) e (5.2) calculam o intervalo normalizado η_i e as diferenças lógicas entre dois níveis abstratos consecutivos, γ_i para os diferentes parâmetros de contexto $C_i \in C$, como ilustra a tabela 5.2.

Tabela 5.2: Intervalo normalizado e diferenças gerais das informações de contexto

Parâmetros	**Valores *Min-Max***	$\boldsymbol{\eta_i}$	$\boldsymbol{\gamma_i}$
$C_{Bateria}$	Max – 99,75 %; Min – 16,65 %	0,83	0,28
C_{Sin}	Max – 59,83 dB; Min – 23,63 dB	0,61	0,3
C_{QoS}	Max – 12,71s; Min – 3,17s	0,75	0,19

Fonte: o autor.

Utilizando a equação (5.3), o conjunto dos níveis quantitativos, L_i para os diferentes parâmetros de contexto $C_i \in C$, é calculado da seguinte forma:

$$\boldsymbol{L_{Bateria} = \{0,17;\ 0,44;\ 0,72\},}$$

$$\boldsymbol{L_{Sin} = \{0,39;\ 0,7\},}$$

$$\boldsymbol{L_{QoS} = \{0,25;\ 0,44;\ 0,62;\ 0,81\}}$$

Em seguida, é calculado o produto cartesiano *F* para os diferentes níveis quantitativos, L_i através da equação (5.4), conforme descrito na tabela 5.3.

Tabela 5.3: produto cartesiano para os diferentes níveis quantitativos

F =

0,17	**0,39**	**0,25**
0,17	**0,39**	**0,44**
0,17	**0,39**	**0,62**
0,17	**0,39**	**0,81**
0,17	**0,7**	**0,25**
0,17	**0,7**	**0,44**
0,17	**0,7**	**0,62**
0,17	**0,7**	**0,81**
0,44	**0,39**	**0,25**
0,44	**0,39**	**0,44**
0,44	**0,39**	**0,62**
0,44	**0,39**	**0,81**
0,44	**0,7**	**0,25**
0,44	**0,7**	**0,44**
0,44	**0,7**	**0,62**
0,44	**0,7**	**0,81**
0,72	**0,39**	**0,25**
0,72	**0,39**	**0,44**
0,72	**0,39**	**0,62**
0,72	**0,39**	**0,81**
0,72	**0,7**	**0,25**
0,72	**0,7**	**0,44**
0,72	**0,7**	**0,62**
0,72	**0,7**	**0,81**

Fonte: o autor.

O multiconjunto correspondente, *M*, é calculado conforme a equação 5.5.

$$\boldsymbol{M = \{0,17;\ \ldots;\ 0,17;\ 0,25;\ 0,39;\ \ldots;\ 0,25;\ 0,44;\ 0,62;\ 0,7\}}$$

Considerando os valores iniciais do θ_i, como: $\theta_{\text{Bateria},0} = 3$, $\theta_{Sin,0} = 2$ e $\theta_{QoS,0} = 8$, o conjunto enviesado $\theta_i, \forall C_i \in C$, que satisfaz a condição em (5.6), é indicado a seguir:

$$\boldsymbol{\theta_{Bateria} = \{3;\ 13;\ 55\}, \theta_{Sin} = \{2;\ 9\}, \theta_{QoS} = \{8;\ 34;\ 143;\ 598\}}$$

Ao aplicar a equação (5.7) a qualquer elemento candidato $\boldsymbol{\mu(0,17;\ 0,39;\ 0,25) \in F}$, a prioridade de contexto estimada, $\delta, \forall C_i \in C$, é calculada como:

$$\boldsymbol{\delta_{Bateria}(\mu) = 0,06;\ \delta_{Sin}(\mu) = 0,04,\ \delta_{QoS}\ (\mu) = 0,17}$$

Através da equação (5.8), obtemos a prioridade estimada, $\hat{\delta}_i\ (\mu), \forall \mu \in F$, conforme descrito na tabela de previsão (tabela 5.4).

Tabela 5.4: Tabela de previsão

Ordem	$C_{Bateria}(\mu)$	$C_{Sin}(\mu)$	$C_{QoS}(\mu)$	$\hat{\delta}_i\ (\mu_k)$
1	0,17	0,39	0,25	0,27
2	0,17	0,39	0,44	0,83
3	0,17	0,39	0,62	3,16
4	0,17	0,39	0,81	12,90
5	0,17	0,70	0,25	0,40
6	0,17	0,70	0,44	1,00
7	0,17	0,70	0,62	3,30
8	0,17	0,70	0,81	13,10
9	0,44	0,39	0,25	0,70
10	0,44	0,39	0,44	2,48
…	…	…	…	…
22	0,72	0,70	0,44	5,5
23	0,72	0,70	0,62	16,6
24	0,72	0,70	0,81	53,3

Fonte: o autor.

5.6.2. Previsão da prioridade do contexto

Na *unidade de previsão do contexto* é aplicada a análise de *Multiple Linear Regression* na tabela de previsão de contexto.

Neste exemplo, consideramos $\epsilon_i = 0$, os seguintes coeficientes de regressão foram calculados através da equação (5.9), $\beta_0 = -3{,}21;\ \beta_1 = 2{,}24;\ \beta_2 = 1{,}1;\ \beta_3 = 5{,}09$.

Estes coeficientes foram utilizados para prever a prioridade do contexto para qualquer tarefa, através da equação (5.10), durante um período de tempo *t = 3* segundos, em um nó da *fog*, composta por $|V| = 2$ máquinas virtuais com as mesmas configurações.

Neste tempo, $|R| = 8$ pedidos com os mesmos comprimentos chegam.

As informações do contexto são normalizadas de acordo com os valores máximos e mínimos de $C_i \in C$ e, através da Equação (5.9), a previsão da prioridade $P_r, \forall r \in R$, é calculada como se mostra na tabela 5.5, com a informação do contexto atual (A) e o seu respetivo valor normalizado (N).

Tabela 5.5: Previsão de prioridade de contexto

ID	Tempo de chegada	$C_{Bateria}$	C_{Sin}	C_{QoS}	P_r
1	0	A:52,60;N:0,53	A:33,48;N:0,56	A:3,56;N:0,28	0,02
2	0	A:18,42;N:0,18	A:57,58;N:0,97	A:5,75;N:0,45	0,57
3	0	A:35,05;N:0,35	A:41,34;N:0,69	A:11,56;N:0,91	2,97
4	0,22	A:84,88;N:0,85	A:35,56;N:0,59	A:8,31;N:0,65	2,68
5	0,72	A:39,73;N:0,40	A:47,69;N:0,80	A:7,94;N:0,63	1,74
6	0,88	A:32,22;N:0,32	A:29,45;N:0,49	A:10,1;N:0,79	2,10
7	1,63	A:39,28;N:0,39	A:38,92;N:0,65	A:6,95;N:0,55	1,17
8	1,77	A:44,00;N:0,44	A:47,54;N:0,79	A: 5,50;N:0,43	0,86

Fonte: o autor.

5.6.3. Otimização do escalonamento de tarefas

De acordo com a função objetivo, equação (5.11), as tarefas são escalonadas em MVs nos servidores. O conjunto de pedidos (*r1* à *r8*) que tenham satisfeito as restrições, são escalonadas na *fog* a cada um segundos.

A tabela 5.6, ilustra como o modelo proposto prioriza os pedidos com base nas informações de contexto e visando a otimização da QoE.

Tabela 5.6: Escalonamento de tarefas

Instante de tempo	ID & Saída da FO	Informações sobre o escalonamento
0,0	*r1 - 0,015* *r2 - 0,573* *r3 - 2,971*	$r1 \rightarrow v0$ $r2 \rightarrow v1$
1,0	*r3 - 1,485* *r5 - 1,743* *r6 - 2,103* *r4 - 2,681*	$r3 \rightarrow v0$ $r5 \rightarrow v1$
2,0	*r8 - 0,857* *r6 - 1,051* *r7 - 1,173* *r4 - 1,340*	$r8 \rightarrow v0$ $r6 \rightarrow v1$
3,0	*r7* - 0,586 *r4*-0,893	$r7 \rightarrow v0$ $r4 \rightarrow v1$

Fonte: o autor.

Este exemplo ilustrativo apresenta o escalonamento de tarefas no modelo proposto.

Às tarefas com menores valores de contexto são atribuídas maior prioridade a fim de otimizar a qualidade da experiência tanto do utilizador como do provedor de serviços.

6. AVALIAÇÃO DO DESEMPENHO

Neste capitulo, analisamos o desempenho da nossa proposta (com IE estático e dinâmico) comparando-a com os algoritmos de escalonamentos não sensível ao contexto: FCFS tal como definido em Zhu et al. (2015), SJF tal como descrito em Heo & Park (2017), e QoS-based tal como referenciado em Feng et al. (2017). As métricas da avaliação definidas são: percentagem de execução dos pedidos bem-sucedidos; tempo médio de espera e QoE dos utilizadores em relação ao aumento dos pedidos, de MVs e de requisitos QoS da aplicação.

6.1. Ambiente de simulação

Existem algumas ferramentas de simulação disponíveis para investigações em ambientes *fog*. Estas ferramentas disponibilizam funcionalidades que permitem simular ambientes *fog* reais.

Nesta tese, utilizamos o *kit* de ferramentas de simulação *iFogSim*, descrito na secção 3.4 do Capítulo 3, para simular e comparar a nossa proposta de escalonamento sensível ao contexto com as outras propostas não sensível ao contexto (*First Come First Served, Shortest Job First e QoS based priority scheduling*). A escolha do *kit* de ferramentas *iFogSim* deve-se ao facto de ela ser uma ferramenta de simulação de código aberto, escalável, permitir a personalização de diferentes classes, proporcionar simulações realísticas e permitir avaliar a eficiência dos algoritmos de escalonamentos em ambientes *fog*.

Diversos autores, tais como, Gupta *et al.* (2016); Bittencourt *et al.* (2017); Skarlat *et al.* (2017); Taneja & Davy (2017); Mahmud, Ramamohanarao, & Buyya (2018); Mestre *et al.* (2019) e Mahmud & Buyya (2018), utilizam-na nas suas investigações e consideram-na como uma importante ferramenta para os investigadores da *fog computing*.

Nas subsecções seguintes, descreveremos as personalizações feitas de forma a acomodar a nossa simulação.

6.1.1. Configuração do ambiente de simulação

O ambiente de simulação do modelo proposto foi desenvolvido no *kit* de ferramentas de simulação *iFogSim*. O ambiente de simulação modela uma *fog* composta por três *hosts* que podem acomodar um máximo de 15 MVs, sendo que cada uma delas possui como caraterísticas: 10000 MB de disco, 512 MB de memória e as velocidades de processamento variarem entre: 1000 a 1500 Milhões de instruções por segundos (MIPS). O tamanho de cada pedido varia entre: 1000 a 2000 milhões de instruções (MI).

Seguindo as inequações 5.16 e 5.17, a *fog* aceita pedidos dos dispositivos com valores mínimos de B_{th} = 15% e SIN_{th} = 15dB. É permitido que as tarefas solicitadas tenham um requisito mínimo de QoS de um segundo. Os parâmetros de contexto heterogêneos ($C_{Bateria}$, C_{Sin} e C_{QoS}) associados a cada pedido podem ter valores aleatórios. Se o nó da *fog* não conseguir disponibilizar os recursos necessários, os pedidos correspondentes são encaminhadas para a *cloud*, conforme descrito em Soyata *et al.* (2013).

A simulação é executada durante 500 segundos. Cada ponto no gráfico corresponde à média aritmética dos resultados obtidos através de 100 execuções de simulação. Os valores dos parâmetros do ambiente de simulação podem ser resumidos conforme a tabela 6.1.

Tabela 6.1: Parâmetros de simulação

Parâmetros	Valores
Número de MV	1 ~ 15
Capacidade do disco de cada MV	10000 MB
Memória de cada MV	512 MB
QoS da Aplicação	6 ~ 15 s
Tamanho do pedido	1000 ~ 2000 MI
B_{th}	15%
SIN_{th}	15 dB
$\alpha_{Bateria}$	3
α_{sin}	2
α_{QoS}	4
Duração da Simulação	500 s

Fonte: o autor.

6.2. Métricas de desempenho

As seguintes métricas de desempenho são utilizadas para comparar a nossa proposta de escalonamento (com IE estático e dinâmico) com abordagens de escalonamento não sensível ao contexto como: *FCFS*, *SJF* e *QoS-based*:

- *Percentagem de execução dos pedidos bem-sucedidos:* é calculada através da razão entre a quantidade de tarefas que preservam os diferentes parâmetros de contexto e o total de tarefas solicitadas. Quanto maior for esta percentagem, maior será a quantidade de tarefas que preservam os diferentes parâmetros de contexto.
- *Tempo médio de espera de uma tarefa na fila:* é o tempo decorrido desde a sua chegada até a sua disponibilização numa MV. Q_r indica o tempo de espera de um pedido $r \in R$. O tempo médio de espera de uma tarefa é calculado conforme a equação 6.1:

$$\widehat{\boldsymbol{Q}} = \frac{\mathbf{1}}{|\boldsymbol{R}|}\sum_{\boldsymbol{r}\in\boldsymbol{R}}(\boldsymbol{Q_r}) \qquad 6.1$$

onde *R* representa a totalidade dos pedidos recebidos durante o período de simulação. Quanto menor for o valor de $\hat{Q}$ maior será o desempenho.

- *Qualidade da Experiência (QoE):* conforme definido na secção 2.6 do capítulo 2, é o grau de satisfação global dos utilizadores relativamente à utilização de um produto ou serviço. Pode ser aperfeiçoada através da diferença entre o tempo máximo permitido para a obtenção da resposta (requisito QoS da aplicação), T_r e o tempo de resposta de um pedido.

  Calculamos a média de QoE de todos os pedidos *R* expedidos durante a simulação da seguinte forma:

$$\boldsymbol{QoE} = \frac{\mathbf{1}}{|\boldsymbol{R}|}\sum_{\boldsymbol{r}\in\boldsymbol{R}}\sum_{\boldsymbol{v}\in\boldsymbol{V}}(\boldsymbol{T_r} - (\boldsymbol{\psi_{r,v}} + \boldsymbol{Q_r})) \qquad 6.2$$

Quanto maior for este valor, melhor será a capacidade do sistema em otimizar a QoE do utilizador.

6.3. Resultados e discussões

Os resultados da avaliação, obtidos através da simulação implementada com base nos parâmetros da tabela 6.1, são descritos nesta secção.

Uma análise detalhada ao ficheiro rastreio da simulação permite-nos concluir que a percentagens de execução dos pedidos com sucesso no modelo sensível ao contexto é superior em comparação com os escalonamentos não sensível ao contexto, onde os contextos do utilizador final não são considerados.

No modelo proposto, os pedidos provenientes dos dispositivos em situações de vulnerabilidades, com baixos valores de contextos, possuem maior precedência de execução. Isto é, é-lhes assegurado o funcionamento prioritário por forma a poderem receber o *feedback* da *fog*. No entanto, como a *fog* possui recursos limitados, o que afeta a performance de serviço, a percentagem de execução dos pedidos com sucesso diminui quando aumentam a quantidade dos pedidos.

Seguidamente, apresentamos e discutimos a variação da percentagem de execução dos pedidos com sucesso em relação ao aumento da quantidade de pedidos, MVs e requisitos QoS da aplicação.

6.3.1. Impactos do aumento de pedidos de aplicação

As percentagens de execução dos pedidos com sucesso preservando os diferentes parâmetros de contexto (mantendo constante a quantidade de MVs), são apresentadas nos gráficos 6.1, 6.2 e 6.3. O tempo médio de espera e a QoE neste cenário são apresentadas nos gráficos 6.4 e 6.5, respetivamente.

Na nossa proposta, os pedidos provenientes de dispositivos com nível de bateria baixo têm maior prioridade. São executadas com maior precedência. Como resultado, os dispositivos obtêm as respostas aos pedidos antes do dispositivo se desligar devido a insuficiência da bateria. Consequentemente, o modelo proposto possui um melhor desempenho em comparação com as outras abordagens em estudo, conforme ilustrado no gráfico 6.1. Isto deve-se ao facto de que nenhum dos outros algoritmos analisados se preocupar com o contexto do nível da bateria. Além disso, o modelo proposto (com IE dinâmico) diminui o tempo de inatividade dos recursos da MV nos servidores o que permite aumentar as hipóteses para os pedidos com contextos críticos serem executadas com maior prioridade.

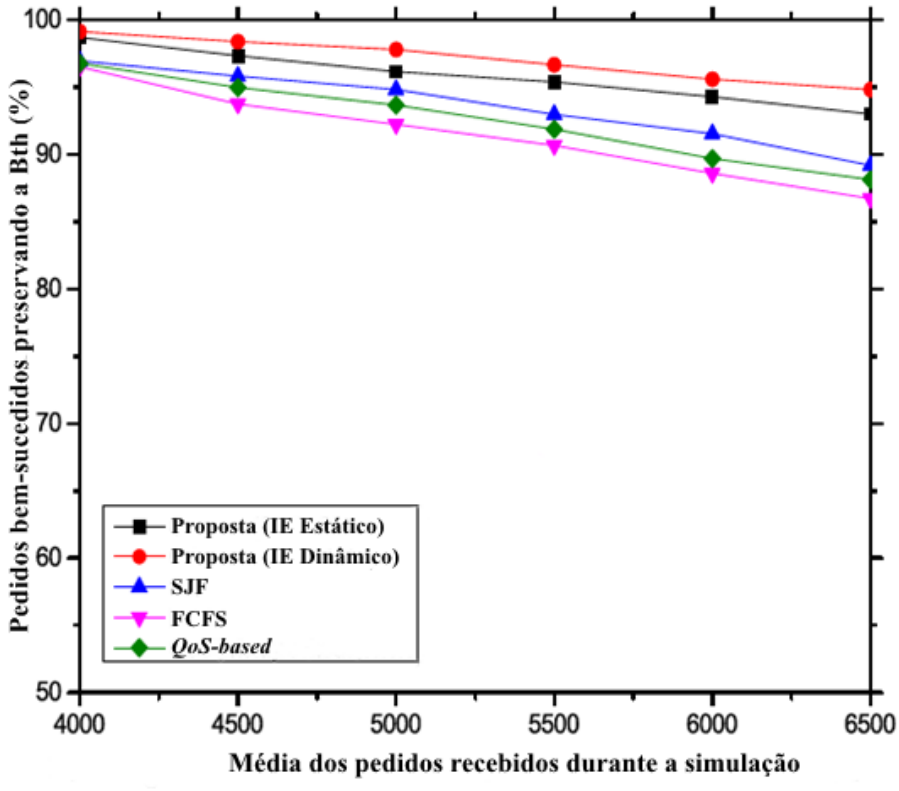


Gráfico 6.1: Tarefas executadas com sucesso preservando o nível da bateria em relação ao aumento da quantidade de pedidos.
Fonte: o autor

Da mesma forma, podemos observar através do gráfico 6.2 que em todas as abordagens estudadas, a percentagem de pedidos de aplicações executados com sucesso preservando o nível do sinal, também diminui gradualmente devido ao aumento de receção dos pedidos. Como teoricamente esperado, a quantidade de pedidos com valores críticos do SIN aumentam em função do aumento da quantidade de pedidos e, como consequência, o desempenho total diminui ao longo do tempo.

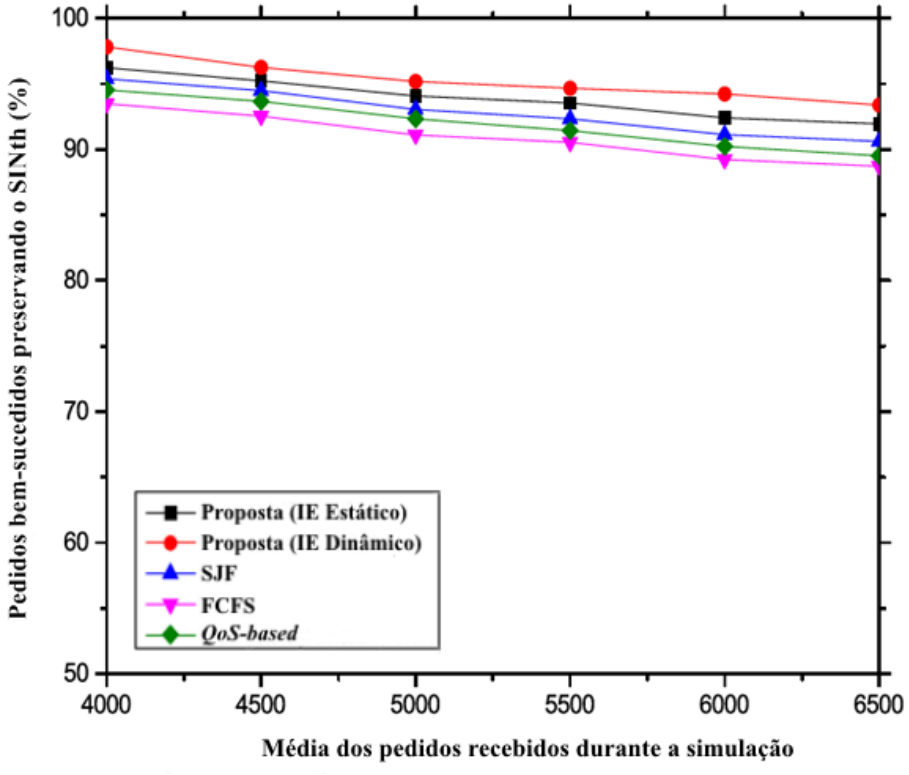


Gráfico 6.2: Tarefas executadas com sucesso preservando o nível do sinal em relação ao aumento da quantidade de pedidos.
Fonte: o autor

O gráfico 6.3 ilustra que a percentagem dos pedidos executados com sucesso, preservando as exigências da QoS, diminui drasticamente em todas as abordagens de escalonamento estudadas em função do aumento da quantidade de pedidos. A razão subjacente a este resultado é explicada pelo facto de que o aumento dos pedidos também permite aumentar o tempo de espera, o que obriga muitas tarefas a violarem os requisitos de QoS. No entanto, o desempenho da nossa proposta supera o dos outros algoritmos (*FCFS, SJF, QoS-based*).

No modelo proposto, o impacto dessa diminuição é significativamente menor, dado que ela privilegia o escalonamento de acordo com os seus requisitos de QoS. Por fim, o resultado demonstra que, para todos os casos, o intervalo de escalonamento dinâmico produz um

impacto ainda melhor de desempenho em comparação com o seu equivalente estático. Uma vez que ela tem a capacidade de responder adequadamente aos pedidos críticos através da utilização eficiente dos recursos computacionais da *fog*.

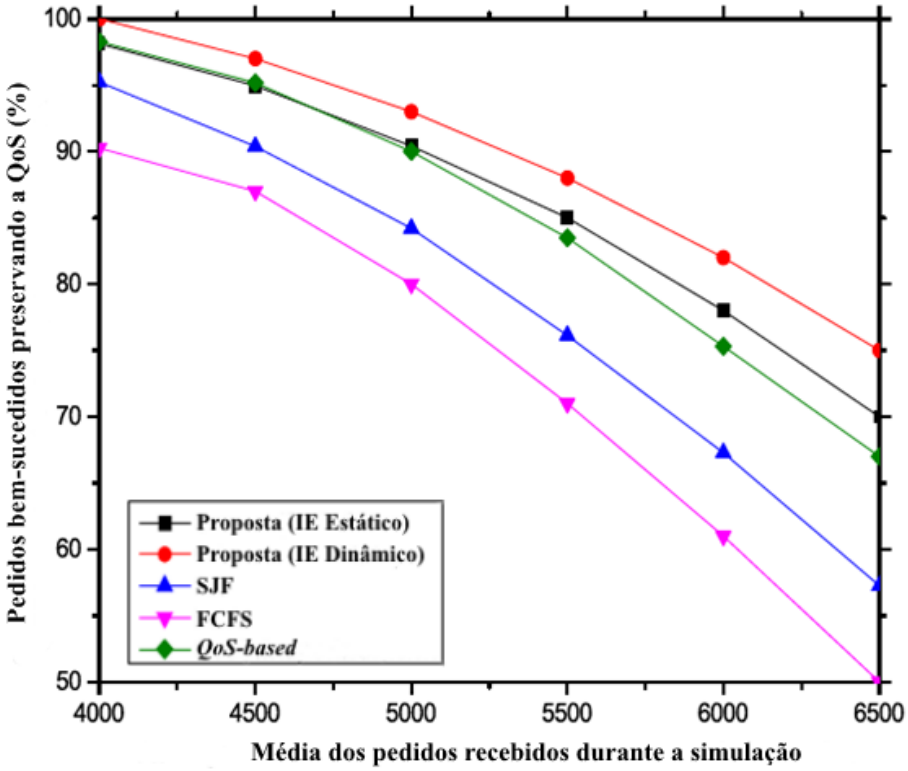


Gráfico 6.3: Tarefas executadas com sucesso preservando o nível da QoS em relação ao aumento da quantidade de pedidos.
Fonte: o autor

O gráfico 6.4 ilustra que o tempo médio de espera no modelo proposto é inferior em comparação com a maioria das outras técnicas não sensível ao contexto. Isto deve-se ao facto de o escalonamento no modelo proposto não considerar apenas o contexto das tarefas, mas também as interrupções dos mesmos enquanto estão na fila de espera para serem escalonadas e ao seu tamanho individual. Por conseguinte, uma tarefa deve ser penalizada por um período curto devido a outros fatores e uma tarefa trivial não deve esperar muito tempo para ser executada, devido à execução de uma tarefa maior. Apesar de o escalonador *SJF* apresentar um melhor desempenho neste cenário, ele possui uma taxa de sucesso de preservação da QoS menor. O tempo médio de espera dos pedidos em todas as abordagens aumentam exponencialmente em função do aumento das quantidades de pedidos recebidos, conforme previsto teoricamente.

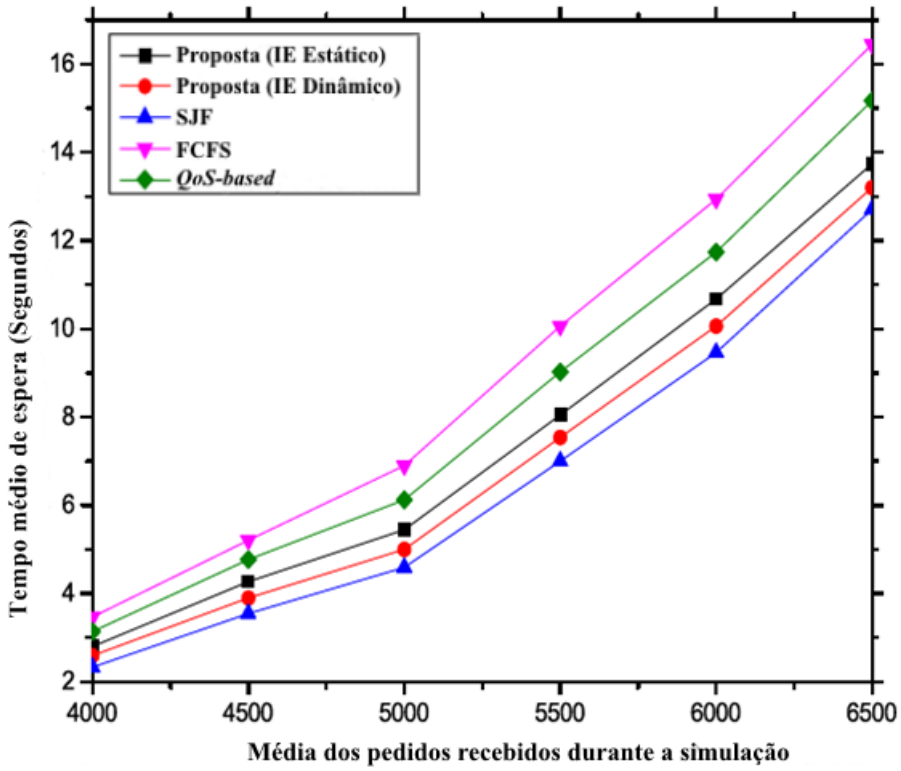


Gráfico 6.4: Tempo médio de espera em relação ao aumento da quantidade de pedidos.
Fonte: o autor

O gráfico 6.5 ilustra a otimização da QoE dos utilizadores finais. O modelo proposto privilegia a execução de tarefas com valores de contexto mais baixos. Como resultado elas são executadas respeitando um tempo mínimo e atraso de resposta tolerável. Relativamente às outras tarefas com parâmetros de contexto adequados, este cenário é suportado de forma intrínseca. Assim, a QoE dos utilizadores é otimizada em todos os cenários possíveis. Em comparação com a maioria das outras técnicas não sensível ao contexto, elas muitas vezes não conseguem otimizar a QoE, porque de acordo com os seus critérios de escalonamento, enquanto satisfazem um pedido com maior prioridade, o pedido com menor prioridade pode não conseguir manter a sua tolerância de QoS. Nos piores cenários, poderá resultar em valores de QoE negativos, enquanto que, no modelo proposto, o valor de QoE é sempre positivo.

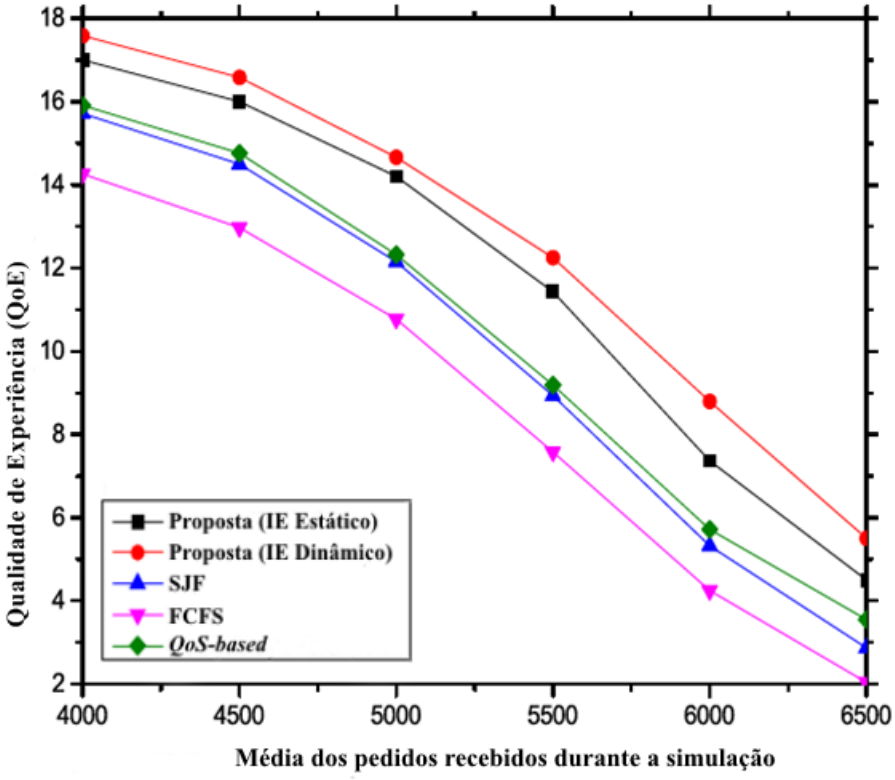


Gráfico 6.5: QoE dos utilizadores em relação ao aumento da quantidade de pedidos.
Fonte: o autor

6.3.2. Impactos do aumento de MVs na *fog*

As percentagens das tarefas executadas com sucesso preservando os diferentes parâmetros de contexto, assumindo que a quantidade de pedidos recebidos durante a simulação é constante em relação ao aumento da quantidade de MVs, são ilustrados nos gráficos 6.6, 6.7 e 6.8. Os gráficos 6.9 e 6.10 ilustram respetivamente o tempo médio de espera e QoE dos utilizadores no critério impactos do aumento das MVs.

Se o número de MVs aumentar, também aumenta a percentagem de execução dos pedidos com sucesso que satisfazem os vários condicionalismos contextuais. Este cenário é apresentado nos gráficos 6.6 e 6.7 tanto em termos de níveis da bateria como em relação à intensidade do sinal da rede.

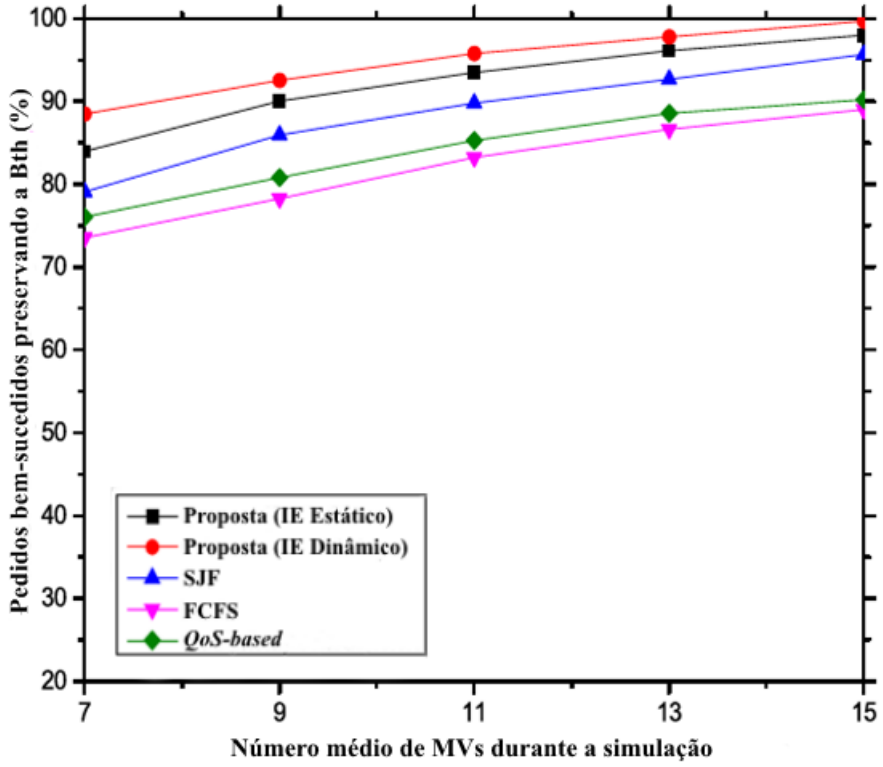


Gráfico 6.6: Pedidos executados preservando o nível de bateria em relação ao aumento de MV
Fonte: o autor

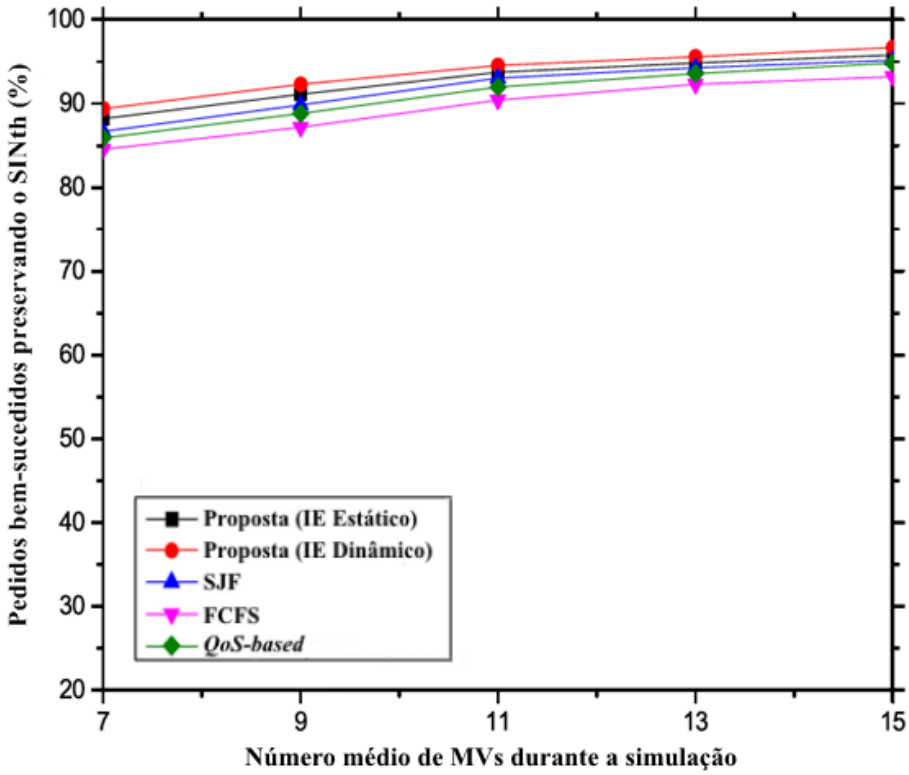


Gráfico 6.7: Pedidos executados preservando a intensidade do sinal em relação ao aumento de MV
Fonte: o autor

O gráfico 6.8 ilustra que a percentagem dos pedidos executados com sucesso preservando o nível de QoS em relação ao aumento das máquinas virtuais. Excetuando o comportamento normal para valores mais altos no eixo X, a percentagem de execução dos pedidos com sucesso é consideravelmente mais alta no modelo proposto em comparação com os outros algoritmos (*FCFS, SJF, QoS-based*), o mesmo também se verifica em relação aos valores mais baixos do eixo X.

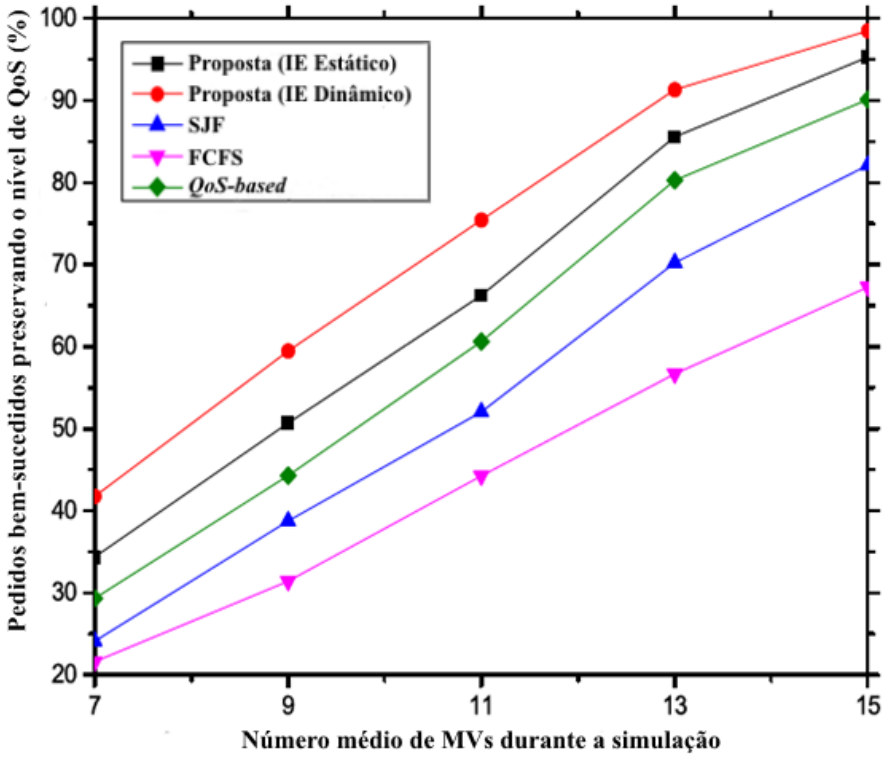


Gráfico 6.8: Pedidos executados preservando a QoS em relação ao aumento de MV.
Fonte: o autor

Quanto ao tempo de espera e QoE em relação ao aumento de MV, conforme os gráficos 6.9 e 6.10, apresentam cenários teoricamente comprovados. Mesmo assim, o desempenho do modelo proposto é sempre superior em comparação com as outras políticas, com exceção do gráfico 6.9, onde o *Shortest Job First* apresenta um melhor desempenho.

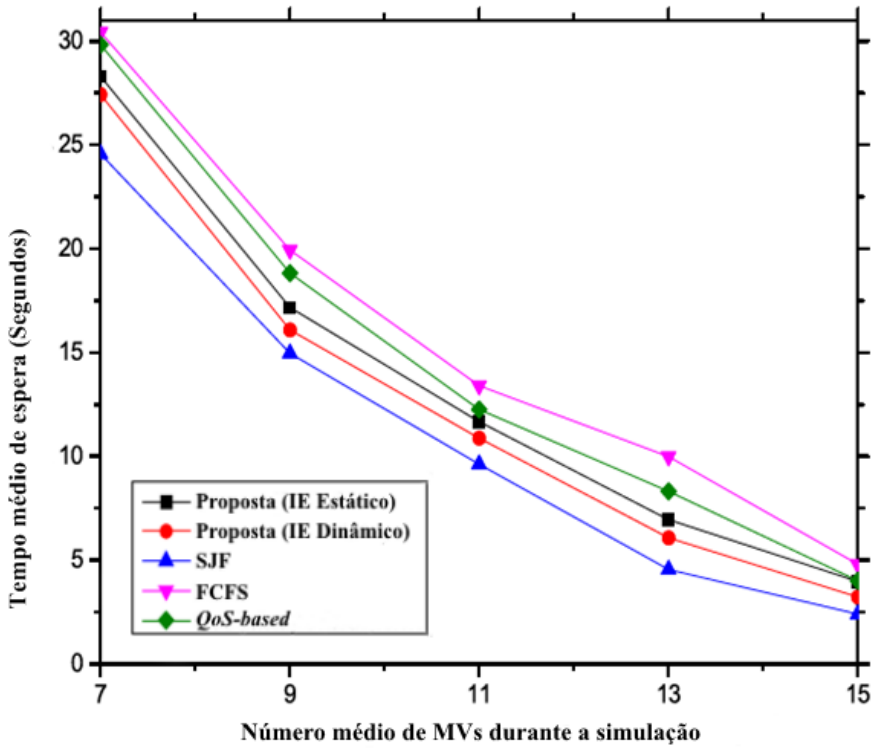


Gráfico 6.9: Tempo médio de espera em relação ao aumento de MV.
Fonte: o autor

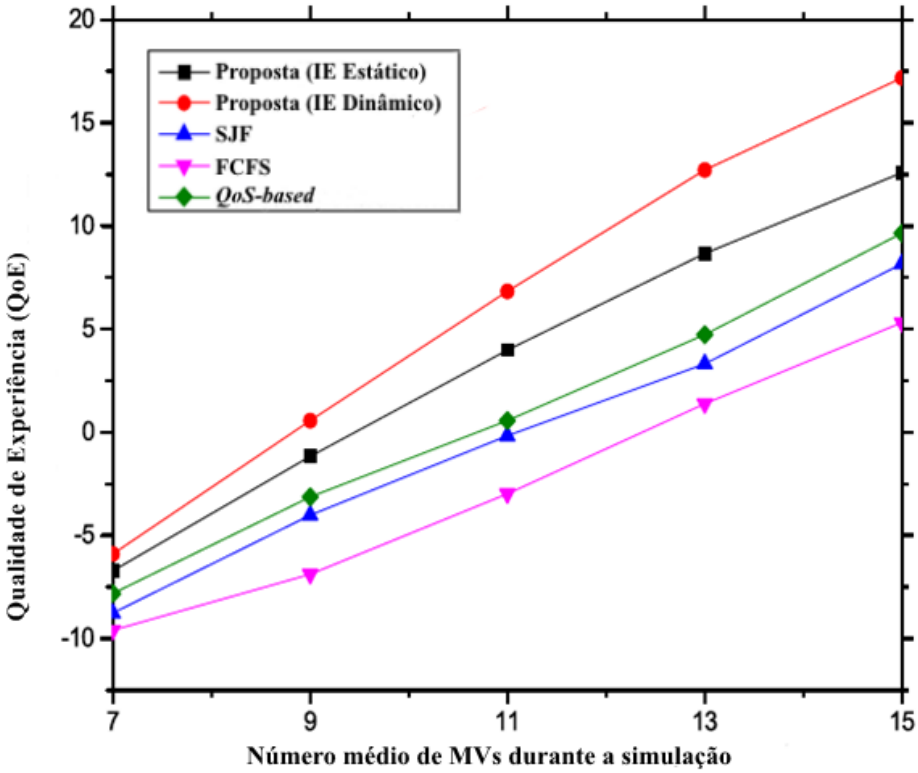


Gráfico 6.10: QoE do utilizador em relação ao aumento de MV.
Fonte: o autor

6.3.3. Impactos do requisito QoS da aplicação

Nos gráficos 6.11 e 6.12 é ilustrado o impacto do requisito QoS dos pedidos no desempenho do modelo proposto. Ilustram que à medida que aumenta o tempo médio admissível de resposta aos pedidos, aumentam também as percentagens das tarefas satisfeitas em termos de QoS e QoE. Isto acontece porque o tempo de execução dos pedidos diminui. Consequentemente, é possível executar tarefas de acordo com a sua exigência de QoS e dentro do tempo mínimo estabelecido. Este facto permite o aperfeiçoamento da QoE dos utilizadores. As outras técnicas de escalonamento não sensível ao contexto, em comparação com o modelo proposto, revelam menos eficiência neste cenário.

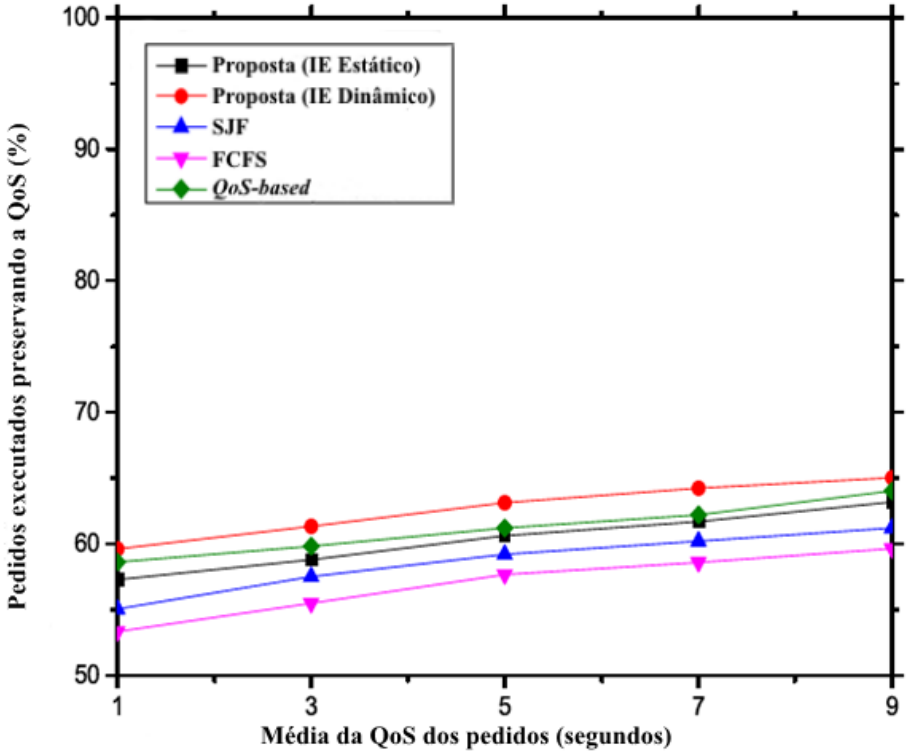


Gráfico 6.11: Pedidos executados em relação ao aumento da QoS da aplicação.
Fonte: o autor

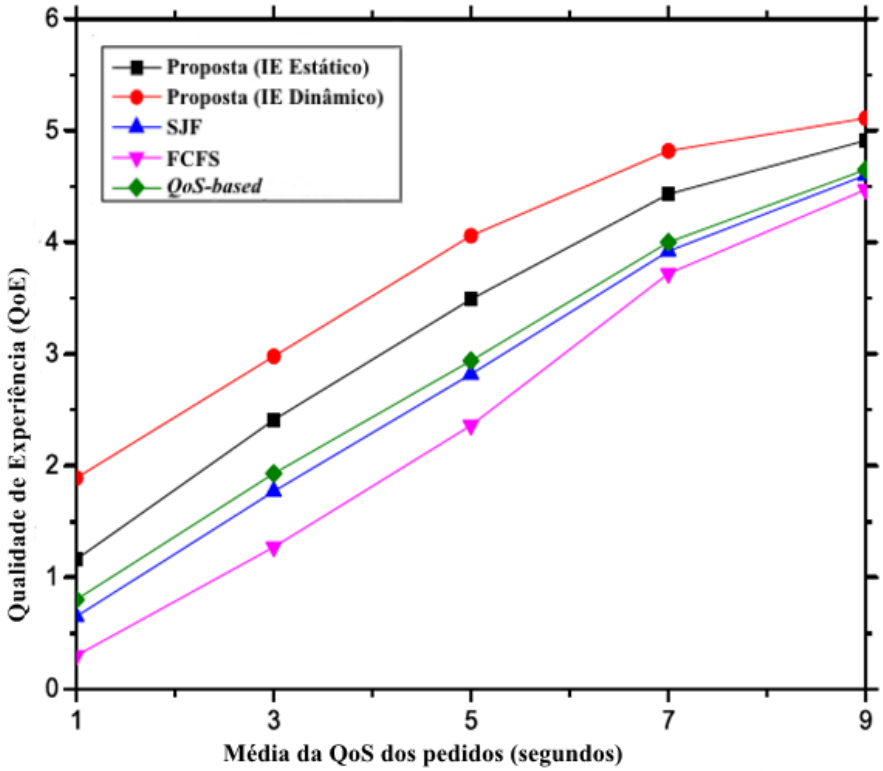


Gráfico 6.12: QoE do utilizador em relação ao aumento da QoS da aplicação.
Fonte: o autor

6.4. Resumo das simulações

Em síntese a esta experiência, afirmamos sem dúvida que o modelo de escalonamento sensível ao contexto proposta, é eficiente em relação à quantidade de tarefas escalonadas e que satisfazem a QoS, diminuição do tempo de resposta, otimização de QoE, redução do tempo médio de espera, entre outros, em comparação com as políticas de escalonamentos não sensível ao contexto. Tendo em consideração as observações fundamentadas apresentadas, podemos afirmar que a nossa proposta de escalonamento possui condições para ser implementada em ambientes práticos.

7. CONCLUSÕES E TRABALHO FUTURO

Neste capítulo final, é apresentada uma síntese dos pontos mais relevantes do trabalho desenvolvido, são tecidas algumas conclusões sobre o trabalho, com particular destaque para a validação dos objetivos propostos inicialmente. Serão também exploradas potenciais linhas de investigação futuras, de forma a permitir a continuidade do trabalho realizado.

7.1. Principais conclusões

O objetivo principal desta tese consiste em definir uma arquitetura de escalonamento de tarefas das aplicações móveis sensível ao contexto para o paradigma *fog computing*. Como apresentada nesta tese, a arquitetura proposta é suficientemente eficiente para priorizar tarefas independentemente da heterogeneidade das suas informações de contextos. Ademais, consegue otimizar a QoE dos utilizadores finais e permitir situações vantajosas para todos em termos de utilização de recursos e tempo de execução.

A validação do modelo e da arquitetura proposta foi feita com base em simulações realizadas no *kit* de ferramentas *iFogSim*, que possibilitou fazer comparações experimentais e teóricas da nossa proposta sensível ao contexto, tanto para intervalos de escalonamento estáticos como dinâmicos com escalonadores não sensível ao contexto com base nas seguintes métricas: percentagem de execução dos pedidos com sucesso; tempo de espera e QoE dos utilizadores.

A revisão da literatura sobre algoritmos de escalonamento na arquitetura *cloud* e no paradigma *fog* permitiu identificar os trabalhos relevantes, discutir as suas limitações, explorar e sugerir algumas perspetivas de melhorias e propor uma arquitetura de escalonamento de tarefas das aplicações móveis sensível ao contexto para o paradigma *fog computing*.

No sentido de atingir o objetivo principal, apresentado no Capítulo 1, foram realizados os seguintes objetivos parcelares:

- Realizámos um estudo sobre o contexto no âmbito da aplicação aos dispositivos móveis, os seus principais desafios, incluindo a definição de sensibilidade ao contexto e ciclo de vida contextual. Abordamos conceitos como a QoS e QoE dos utilizadores.
- Definimos a *cloud computing* e as suas tecnologias emergentes, destacamos o paradigma *fog computing*, a sua necessidade, benefícios, caraterísticas e arquitetura referência proposta pelo consórcio *OpenFog*. Também foi descrito o *kit* de ferramentas *iFogSim*. Pretendemos com esse objetivo parcelar abordar as principais teorias e conceitos relacionados com o nosso trabalho.

- Fizemos um levantamento bibliográfico sobre os algoritmos de escalonamento na arquitetura *cloud* e no paradigma *fog*, identificámos os trabalhos relevantes, discutimos as suas limitações, explorámos e sugerimos algumas perspetivas de melhorias. Neste objetivo parcelar, concluímos que o escalonamento na *cloud*, foi amplamente estudado. No entanto, na *fog*, devido à densidade e heterogeneidade de dispositivos, o escalonamento é complexo e ainda existem poucos estudos, tanto quanto é do nosso conhecimento; muitos dos algoritmos de escalonamento identificados não descrevem a forma como a prioridade é definida, não explicam o método utilizado na priorização de tarefas e nem definem a priorização de tarefas com base em informações do contexto, muitos defendem a perspetiva dos provedores de serviços. Outros são aplicados em tarefas agrupadas para diminuir o tempo de execução. Alguns otimizam apenas a QoS. Outros exploram apenas alguns contextos. Escalonadores para a arquiteturas *cloud* e paradigma *fog* identificados permitem resolver muitos problemas. Contudo, alguns aspetos como a utilização do contexto no escalonamento, priorização de tarefas sensível ao contexto, consideração da restrição energética no escalonamento, preservação da força do sinal da rede, preservação da QoS, redução do tempo médio de espera e otimização da QoE podem ser explorados e/ou melhorados;
- Com base nas limitações e nas sugestões de melhorias dos trabalhos relevantes analisados, propusemos um modelo de escalonamento de tarefas das aplicações móveis sensível ao contexto para o paradigma *fog computing* e apresentamos um exemplo ilustrativo que ajuda a visualizar as funcionalidades do modelo proposto, que utiliza a normalização *Min-Max*, para normalizar os diferentes parâmetros do contexto e resolver o problema de heterogeneidade dos contextos dos dispositivos e das aplicações. Para a definição da prioridade do contexto dos pedidos é utilizada a análise da MLR, que permite disponibilizar um conjunto de dados hipotéticos. O escalonamento ótimo dos pedidos visando otimizar a QoE, tendo em conta às várias restrições ao nível da aplicação e do serviço foi solucionado através da utilização da técnica MONLP. A definição dos parâmetros do sistema permitiu um escalonamento mais eficiente, que garante a boa utilização dos recursos, bem como a execução eficaz das tarefas.

- Implementámos a arquitetura proposta, especificada nesta tese, no *kit* de ferramentas *iFogSim* o que possibilitou comparar o desempenho da arquitetura proposta com escalonadores não sensível ao contexto (FCFS, SJF e *QoS-based*).
- Avaliámos o desempenho da arquitetura proposta, que permitiu assegurar que as funcionalidades implementas seguem as especificações definidas. O desempenho foi avaliado em termos de percentagem de execução dos pedidos bem-sucedidos; tempo médio de espera e QoE dos utilizadores em relação ao aumento dos pedidos, de MVs e dos requisitos QoS da aplicação.

Deste modo, os objetivos parcelares, focados no capítulo introdutório, foram atingidos com sucesso, tendo sido apresentadas contribuições inovadoras nas áreas de escalonamento de tarefas e computação distribuída.

Ao nível das questões de investigação, o levantamento do estado da arte sobre os algoritmos de escalonamento na arquitetura *cloud* e no paradigma *fog* e o modelo proposto permitiu-nos dar resposta às mesmas: através da identificação dos diferentes tipos de contexto em computação móvel, conforme descrito na secção 2.4 do Capítulo 2 e da análise do domínio de problema descrito na secção 5.1 do Capítulo 5 desta tese (QI-1);

A revisão da literatura efetuada sobre os principais algoritmos de escalonamento na arquitetura *cloud* e paradigma *fog* permitiu identificar os algoritmos de escalonamento de tarefas na arquitetura *cloud* e paradigma *fog*, conforme descrito na secção 4.2 do Capítulo 4 desta tese (QI-2);

A mesma revisão da literatura permitiu discutir as limitações, explorar e sugerir algumas perspetivas de melhorias dos trabalhos relevantes, conforme descrito nas subsecções 4.3.1 e 4.3.2 do capítulo 4 desta tese (QI-3);

Através da arquitetura proposta, a unidade de recuperação de informações de contexto, para normalizar os diferentes parâmetros do contexto e resolver os problemas de heterogeneidade, utiliza a normalização *Min-Max* conforme referenciado na secção 5.3 do capítulo 5 desta tese (QI-4);

Atendendo a arquitetura proposta, a unidade de priorização de tarefas sensível ao contexto, para a definição da prioridade do contexto dos pedidos utiliza a análise da MLR (QI-5);

Na unidade de QoE e escalonamento sensível ao contexto da arquitetura proposta é feito o escalonamento ótimo dos pedidos visando otimizar a QoE, tendo em conta às várias restrições ao nível da aplicação e do serviço, através da utilização da técnica MONLP (QI-6);

7.1.1. Limitações do estudo

Embora o modelo proposto tenha demonstrado as suas potencialidades relativamente aos resultados em termos de desempenho e precisão, ainda existem alguns aspetos a melhorar, que podem ser sintetizados da seguinte forma:

- O *kit* de ferramentas de simulação *iFogSim* proporciona um ambiente *fog computing* com uma sensação de utilização real. Poderá, no entanto, ser possível alcançar melhores desempenhos se a arquitetura proposta for implementada num ambiente *fog* real;
- A utilização de conjunto de dados reais ajuda a tomar melhores decisões;
- A extensão das métricas de desempenho utilizadas pode vir a permitir uma avaliação mais precisa do sistema proposto;
- Os parâmetros de contexto considerados são limitados para simplificação do modelo (nível da bateria, conectividade e requisitos de QoS), podendo ser estendidos a outros parâmetros ao nível do dispositivo e da aplicação, bem como avaliada a sua influência e impacto nos resultados. A focalização desta correlação pode ser aprofundada em trabalhos futuros.
- Ao nível da infraestrutura, a utilização de máquinas virtuais para a execução das tarefas, por ser dispendiosa e pesada, constitui outra limitação.

7.1.2. Linhas de orientação futura

O trabalho apresentado nesta tese, apesar de ser completo por si só, não esgota totalmente as potencialidades do tema, tal como o modelo e a arquitetura proposta, que não representa uma solução definitiva para o problema, mas antes uma contribuição valiosa para o debate da temática e do problema na comunidade científica. Assim, são apresentadas em seguida algumas linhas de orientação futura que ambicionam inspirar a continuidade desta investigação por forma que as limitações identificadas na subsecção 7.1.1 possam ser ultrapassadas e melhoradas.

Alguns futuros pontos de interesse relevantes que poderão ser desenvolvidos na continuidade deste trabalho são:

- Os parâmetros de contexto ao nível do dispositivo e da aplicação considerados podem ser aperfeiçoados;
- Apesar dos resultados de avaliação mostrarem que o desempenho do modelo proposto melhora com o aumento dos níveis dos parâmetros de contexto, uma análise estatística para um adequado nivelamento dos parâmetros de contexto pode ser uma dimensão de investigação futura;
- Na nossa proposta utilizamos as maquinas virtuais para a execução das tarefas. No entanto, estamos cientes que existem tecnologias mais "leves" e menos "dispendiosas" como os *Containers*, *Docker*, *Kubernetes*, entre outros. Sugeríamos a utilização destas tecnologias aplicado ao nosso modelo como trabalho futuro.
- O modelo e a arquitetura propostos podem ser implementados em infraestrutura *fog* real e o desempenho ser analisado nesse ambiente.

REFERÊNCIAS BIBLIOGRÁFICAS

Aazam, M., St-Hilaire, M., Lung, C., & Lambadaris, I. (2016). MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT. In *2016 23rd International Conference on Telecommunications (ICT)* (pp. 1–5). https://doi.org/10.1109/ICT.2016.7500362

Ameyed, D., Miraoui, M., & Tadj, C. (2015). A survey of prediction approach in pervasive computing. *International Journal of Scientific and Engineering Research*, *6*(5), 306–316.

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *2*(4), 263–277. https://doi.org/10.1504/IJAHUC.2007.014070

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020a). Scheduling in Cloud and Fog Architecture: Identification of Limitations and Suggestion of Improvement Perspectives. *Journal of Information Systems Engineering and Management*, *5*(3), em0121. https://doi.org/10.29333/jisem/8429

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020b). Survey on Job Scheduling in Cloud-Fog Architecture. In *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–7). https://doi.org/10.23919/CISTI49556.2020.9141156.

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020c). Uma Proposta de Algoritmos de Escalonamento de Aplicações Móveis Sensível ao Contexto para o Paradigma Fog Computing. *Revista Das Ciências Da Computação Da Universidade Aberta*, (15), 11. https://doi.org/https://doi.org/10.34627/rcc.v15i0

Barros, C., Rosio, V., Sousa, A., & Paredes, H. (2019). Architecture and platform for Fog Computing. *International Workshop on ADVANCEs in ICT Infrastructures and Services*, (7th), 7. Retrieved January 1, 2020, from https://easychair.org/publications/preprint_open/RvJ9

Barros, C., Rosio, V., Sousa, A., & Paredes, H. (2020). Survey on Job Scheduling in Cloud-Fog Architecture. In *Iberian Conference on Information Systems and Technologies,*

CISTI (Vol. 2020-June). https://doi.org/10.23919/CISTI49556.2020.9141156

Bazire, M., & Brézillon, P. (2005). Understanding Context Before Using It. In A. Dey, B. Kokinov, D. Leake, & R. Turner (Eds.), *Modeling and Using Context* (pp. 29–40). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/11508373_3

Bellavista, P., Corradi, A., Fanelli, M., & Foschini, L. (2012). A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM Comput. Surv.*, *44*(4), 1–45. https://doi.org/10.1145/2333112.2333119

Berg, F., Dürr, F., & Rothermel, K. (2014). Increasing the Efficiency and Responsiveness of Mobile Applications with Preemptable Code Offloading. In *2014 IEEE International Conference on Mobile Services* (pp. 76–83). https://doi.org/10.1109/MobServ.2014.20

Berger, Y. G., Tirari, M. E. H., & Tille, Y. (2003). Towards optimal regression estimation in sample surveys. *Australian & New Zealand Journal of Statistics*, *45*(3), 319–329. https://doi.org/10.1111/1467-842X.00286

Bitam, S., Zeadally, S., & Mellouk, A. (2018). Fog computing job scheduling optimization based on bees swarm. *Enterprise Information Systems*, *12*(4), 373–397. https://doi.org/10.1080/17517575.2017.1304579

Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., & Parashar, M. (2017). Mobility-Aware Application Scheduling in Fog Computing. *IEEE Cloud Computing*, *4*(2), 26–35. https://doi.org/10.1109/MCC.2017.27

Bonomi, F., Milito, R., Natarajan, P., & Zhu, J. (2014). Fog Computing: A Platform for Internet of Things and Analytics. In N. Bessis & C. Dobre (Eds.), *Big Data and Internet of Things: A Roadmap for Smart Environments* (pp. 169–186). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-05029-4_7

Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing* (pp. 13–16). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2342509.2342513

Brooks, P., & Hestnes, B. (2010). User measures of quality of experience: why being objective and quantitative is important. *IEEE Network*, *24*(2), 8–13. https://doi.org/10.1109/MNET.2010.5430138

Buchholz, T., Kupper, A., & Schiffers, M. (2003). Quality of Context: What It Is And Why We Need It. In *In Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03*.

Cardellini, V., Grassi, V., Presti, F. Lo, & Nardelli, M. (2015). On QoS-aware scheduling of data stream applications over fog computing infrastructures. In *2015 IEEE Symposium on Computers and Communication (ISCC)* (pp. 271–276). IEEE. https://doi.org/10.1109/ISCC.2015.7405527

Chen, H. (2004). *An Intelligent Broker Architecture for Context-Aware Systems*. (Ph.D), University of Maryland, Baltimore County. Retrieved June 20, 2020, from http://ebiquity.umbc.edu/paper/html/id/212/An-Intelligent-Broker-Architecture-for-Pervasive-Context-Aware-Systems

Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). CloneCloud: Elastic execution between mobile device and cloud. In *EuroSys'11 - Proceedings of the EuroSys 2011 Conference* (pp. 301–314). https://doi.org/10.1145/1966445.1966473

Crowley, J. L., Coutaz, J., Rey, G., & Reignier, P. (2002). Perceptual Components for Context Aware Computing. In G. Borriello & L. E. Holmquist (Eds.), *UbiComp 2002: Ubiquitous Computing* (pp. 117–134). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45809-3_9

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI: Making Smartphones Last Longer with Code Offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services* (pp. 49–62). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1814433.1814441

Das, A. K., Adhikary, T., Razzaque, M. A., & Choong Seon Hong. (2013). An intelligent approach for virtual machine and QoS provisioning in cloud computing. In *The International Conference on Information Networking 2013 (ICOIN)* (pp. 462–467). IEEE. https://doi.org/10.1109/ICOIN.2013.6496423

Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., & Buyya, R. (2016). Fog Computing: principles, architectures, and applications. In R. Buyya & A. Vahid Dastjerdi (Eds.), *Internet of Things* (pp. 61–75). Elsevier.

https://doi.org/10.1016/B978-0-12-805395-9.00004-6

Deng, R., Lu, R., Lai, C., Luan, T. H., & Liang, H. (2016). Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption. *IEEE Internet of Things Journal*, *3*(6), 1–1. https://doi.org/10.1109/JIOT.2016.2565516

Devaraju, A., Hoh, S., & Hartley, M. (2007). A context gathering framework for context-aware mobile solutions. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology - Mobility '07* (p. 39). New York, New York, USA: ACM Press. https://doi.org/10.1145/1378063.1378070

Dey, A.K., Salber, D., Abowd, G. D., & Futakawa, M. (1999). The Conference Assistant: combining context-awareness with wearable computing. In *Digest of Papers. Third International Symposium on Wearable Computers* (pp. 21–28). IEEE Comput. Soc. https://doi.org/10.1109/ISWC.1999.806639

Dey, Anind K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, *5*(1), 4–7. https://doi.org/10.1007/s007790170019

Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, *13*(18), 1587–1611. https://doi.org/10.1002/wcm.1203

Dsouza, C., Ahn, G.-J., & Taguinod, M. (2014). Policy-driven security management for fog computing: Preliminary framework and a case study. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)* (pp. 16–23). IEEE. https://doi.org/10.1109/IRI.2014.7051866

ETISI. (2003). Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia; IP Interworking over satellite; Performance, Availability and Quality of Service. Retrieved June 12, 2020, from https://www.etsi.org/deliver/etsi_tr/102100_102199/102157/01.01.01_60/tr_102157v010101p.pdf

Fan, J., Wei, X., Wang, T., Lan, T., & Subramaniam, S. (2017). Deadline-Aware Task Scheduling in a Tiered IoT Infrastructure. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference* (pp. 1–7). IEEE.

https://doi.org/10.1109/GLOCOM.2017.8255037

Feng, F., Peng, F., Yan, B., Lin, S., & Zhang, J. (2017). QoS-based LTE downlink scheduling algorithm for smart grid communication. In *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)* (pp. 548–552). IEEE. https://doi.org/10.1109/ICCSN.2017.8230172

Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems*, *29*(1), 84–106. https://doi.org/10.1016/j.future.2012.05.023

Fritsch, J., & Walker, C. (2014). The Problem with Data. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing* (pp. 708–713). IEEE. https://doi.org/10.1109/UCC.2014.115

Gawali, M. B., & Shinde, S. K. (2018). Task scheduling and resource allocation in cloud computing using a heuristic approach. *Journal of Cloud Computing*, *7*(1), 4. https://doi.org/10.1186/s13677-018-0105-8

Ghouma, H., & Jaseemuddin, M. (2015). Context aware resource allocation and scheduling for mobile cloud. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)* (pp. 67–70). IEEE. https://doi.org/10.1109/CloudNet.2015.7335282

Gill, S. S., Garraghan, P., & Buyya, R. (2019). ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices. *Journal of Systems and Software*, *154*, 125–138. https://doi.org/10.1016/j.jss.2019.04.058

Google trends. (2020). Google Trends comparar. Retrieved January 22, 2020, from https://trends.google.com/trends/explore?date=today 5-y&q=Edge Computing,Fog Computing,Mobile Cloud,Cloudlets,Micro Data Centers

Gordon, M., Jamshidi, D., Mahlke, S., Mao, Z., & Chen, X. (2012). COMET: code offload by migrating execution transparently. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation* (pp. 93–106).

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly*, *37*(2), 337–355. https://doi.org/10.25300/MISQ/2013/37.2.01

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, *47*(9), 1275–1296. https://doi.org/10.1002/spe.2509

Han, J., Kamber, M., & Jian, P. (2012). *Data Mining: Concepts and Techniques*. Elsevier. https://doi.org/10.1016/C2009-0-61819-5

Han, M. (2013). *An Integrative Human Activity Recognition Framework based on Smartphone Multimodal Sensors*. (Ph.D), Kyung Hee University Seoul. Retrieved February 20, 2020, from http://uclab.khu.ac.kr/resources/thesis/PhD_Thesis_Manhyung.pdf?PHPSESSID=4cb021dcab4cabdacd8c711fd52a6a9c

Harzing. (2020). Publish or Perish, Research in International Management. Retrieved July 6, 2020, from https://harzing.com/resources/publish-or-perish

Henricksen, K. (2003). *A framework for context-aware pervasive computing application*. (Ph.D), University of Queensland. Retrieved February 10, 2020, from http://henricksen.id.au/publications/phd-thesis.pdf

Heo, J., & Park, M. (2017). Shortest Job First Scheduling for Reducing Jitter in Cyber Physical Systems. In *Proceedings of the International Conference on Embedded Systems, Cyber-physical Systems, and Applications (ESCS)* (pp. 23–28).

Heuer, R., & CIA. (1999). *Psychology of intelligence analysis (Vol. 1): Center for the Study of Intelligence*. Central Intelligence Agency. Retrieved September 11, 2020, from https://www.cia.gov/static/9a5f1162fd0932c29bfed1c030edf4ae/Pyschology-of-Intelligence-Analysis.pdf

Hevner, A., & Chatterjee, S. (2010). Design Science Research in Information Systems. In *Management Information Systems Quarterly - MISQ* (Vol. 28, pp. 9–22). https://doi.org/10.1007/978-1-4419-5653-8_2

Hoareau, C., & Satoh, I. (2009). Modeling and Processing Information for Context-Aware Computing: A Survey. *New Generation Computing*, *27*(3), 177–196. https://doi.org/10.1007/s00354-009-0060-5

Intharawijitr, K., Iida, K., & Koga, H. (2016). Analysis of fog model considering computing and communication latency in 5G cellular networks. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)* (pp. 1–4). IEEE. https://doi.org/10.1109/PERCOMW.2016.7457059

ISO/IEC/IEEE 42010:2011. (2011). ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description. Retrieved May 5, 2020, from https://www.iso.org/standard/50508.html

ISO 8402:1994. (1994). Quality management and quality assurance - Vocabulary. Retrieved May 5, 2020, from https://kupdf.net/download/iso-8402-1994-iso-definitions_58fa42e8dc0d607b44959eaa_pdf

ISO 9000:2015. (2015). Quality management systems - Fundamentals and vocabulary. Retrieved May 6, 2020, from https://kupdf.net/download/np-en-iso-9000-2015_59ce78f908bbc5765a686f67_pdf

ITU-T Rec. E.800. (2008). Definition of terms related to Quality of Service. Retrieved July 14, 2020, from https://www.itu.int/rec/T-REC-E.800-200809-I/en

ITU. (2008). Subjective Video Quality Assessment Methods for Multimedia Applications. Retrieved July 14, 2020, from https://www.itu.int/rec/T-REC-P.910-200804-I

ITU. (2015). Method for the Subjective Assessment of Intermediate Quality Level of Coding Systems. Retrieved July 14, 2020, from https://www.itu.int/rec/R-REC-BS.1534-3-201510-I/en

ITU. (2017). *Quality of Service - Regulation Manual*. Switzerland. Retrieved July 14, 2020, from https://www.itu.int/dms_pub/itu-d/opb/pref/D-PREF-BB.QOS_REG01-2017-PDF-E.pdf

ITU. (2019). New definitions for inclusion in Rec. ITU-T P10/G100No Title. Retrieved July 20, 2020, from https://www.itu.int/rec/T-REC-P.10-201906-I!Amd1/en

Jararweh, Y., Tawalbeh, L., Ababneh, F., & Dosari, F. (2013). Resource Efficient Mobile Computing Using Cloudlet Infrastructure. In *2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks* (pp. 373–377). IEEE. https://doi.org/10.1109/MSN.2013.75

Khoda, M. E., Razzaque, M. A., Almogren, A., Hassan, M. M., Alamri, A., & Alelaiwi, A. (2016). Efficient Computation Offloading Decision in Mobile Cloud Computing over 5G Network. *Mobile Networks and Applications*, *21*(5), 777–792. https://doi.org/10.1007/s11036-016-0688-6

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews Kitchenham, B., 2004. Keele, UK, Keele University* (Vol. 33). Retrieved July 14, 2020, from https://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf

Klas, G. (2016). Edge Cloud to Cloud Integration for IoT. Retrieved June 12, 2020, from https://yucianga.info/?p=1008

Knappmeyer, M., Kiani, S. L., Reetz, E. S., Baker, N., & Tonjes, R. (2013). Survey of Context Provisioning Middleware. *IEEE Communications Surveys & Tutorials*, *15*(3), 1492–1519. https://doi.org/10.1109/SURV.2013.010413.00207

Kofod-Petersen, A., & Cassens, J. (2006). Using Activity Theory to Model Context Awareness. In *FLAIRS Conference* (Vol. 2006, pp. 1–17). https://doi.org/10.1007/11740674_1

Kosta, S., Aucinas, A., Pan Hui, Mortier, R., & Xinwen Zhang. (2012). ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *2012 Proceedings IEEE INFOCOM* (pp. 945–953). IEEE. https://doi.org/10.1109/INFCOM.2012.6195845

Kuhn, T., & Hacking, I. (2012). *The structure of scientific revolutions* (Fourth Edi). University of Chicago Press.

La, H. J., & Kim, S. D. (2010). A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services. In *2010 IEEE 3rd International Conference on Cloud Computing* (pp. 466–473). IEEE. https://doi.org/10.1109/CLOUD.2010.78

Lawanyashri, M., Balusamy, B., & Subha, S. (2017). Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications.

Informatics in Medicine Unlocked, *8*, 42–50. https://doi.org/10.1016/j.imu.2017.02.005

Li, J., Bu, K., Liu, X., & Xiao, B. (2013). ENDA. In *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing - MCC '13* (p. 39). New York, New York, USA: ACM Press. https://doi.org/10.1145/2491266.2491274

Li, T., Liu, Y., Gao, L., & Liu, A. (2017). A Cooperative-Based Model for Smart-Sensing Tasks in Fog Computing. *IEEE Access*, *5*, 21296–21311. https://doi.org/10.1109/ACCESS.2017.2756826

Luan, T. H., Gao, L., Li, Z., Xiang, Y., We, G., & Sun, L. (2016). A View of Fog Computing from Networking Perspective. Retrieved November 12, 2019, from http://arxiv.org/abs/1602.01509

Mahmud, M. R., Afrin, M., Razzaque, M. A., Hassan, M. M., Alelaiwi, A., & Alrubaian, M. (2016). Maximizing quality of experience through context-aware mobile application scheduling in cloudlet infrastructure. *Software: Practice and Experience*, *46*(11), 1525–1545. https://doi.org/10.1002/spe.2392

Mahmud, R., & Buyya, R. (2019). Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit. In *Fog and Edge Computing* (pp. 433–465). Hoboken, NJ, USA: John Wiley & Sons, Inc. https://doi.org/10.1002/9781119525080.ch17

Mahmud, R., Ramamohanarao, K., & Buyya, R. (2019). Latency-Aware Application Module Management for Fog Computing Environments. *ACM Transactions on Internet Technology*, *19*(1), 1–21. https://doi.org/10.1145/3186592

Mahmud, R., Srirama, S. N., Ramamohanarao, K., & Buyya, R. (2019). Quality of Experience (QoE)-aware placement of applications in Fog computing environments. *Journal of Parallel and Distributed Computing*, *132*, 190–203. https://doi.org/10.1016/j.jpdc.2018.03.004

Mertler, C., & Reinhart, V. (2016). *Advanced and Multivariate Statistical Methods* (6th Editio). CA, USA: Routledge. https://doi.org/10.4324/9781315266978

Mestre A., Charântola D., Zane R., Bittencourt, F. (2019). Gerenciamento de Recursos em sistemas distribuídos. Retrieved May 23, 2020, from https://www.ic.unicamp.br/~reltech/PFG/2019/PFG-19-10.pdf

Miettinen, K. (1998). *Nonlinear Multiobjective Optimization*. (S. Editors., Ed.) (Vol. 12). Boston, MA: Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-5563-6

Moore, P., Hu, B., Zhu, X., Campbell, W., & Ratcliffe, M. (2007). A Survey of Context Modeling for Pervasive Cooperative Learning. In *2007 First IEEE International Symposium on Information Technologies and Applications in Education* (pp. K5-1-K5-6). IEEE. https://doi.org/10.1109/ISITAE.2007.4409367

Musumba, G. W., & Nyongesa, H. O. (2013). Context awareness in mobile computing: A review. *International Journal of Machine Learning and Applications*, *2*(1). https://doi.org/10.4102/ijmla.v2i1.5

NetworkWorld. (2015). Network World: 2015 State of the Network survey. Retrieved January 14, 2020, from https://www.networkworld.com/article/2901257/network-world-2015-state-of-the-network-survey.html

OpenFog. (2017). OpenFog Reference Architecture for Fog Computing. Retrieved February 24, 2020, from https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf

Oueis, J., Strinati, E. C., & Barbarossa, S. (2015). The Fog Balancing: Load Distribution for Small Cell Cloud Computing. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)* (pp. 1–6). IEEE. https://doi.org/10.1109/VTCSpring.2015.7146129

Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In *Digest of Papers. Second International Symposium on Wearable Computers (Cat. No.98EX215)* (pp. 92–99). IEEE Comput. Soc. https://doi.org/10.1109/ISWC.1998.729534

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, *16*(1), 414–454. https://doi.org/10.1109/SURV.2013.042313.00197

Qiu, M., Chen, Z., Yang, L. T., Qin, X., & Wang, B. (2012). Towards Power-Efficient Smartphones by Energy-Aware Dynamic Task Scheduling. In *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems* (pp. 1466–1472). IEEE. https://doi.org/10.1109/HPCC.2012.214

Quinn, G. P., & Keough, M. J. (2002). *Experimental Design and Data Analysis for Biologists*. Cambridge University Press. https://doi.org/10.1017/CBO9780511806384

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, *30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Sahoo, P. K., & Dehury, C. K. (2018). Efficient data and CPU-intensive job scheduling algorithms for healthcare cloud. *Computers & Electrical Engineering*, *68*, 119–139. https://doi.org/10.1016/j.compeleceng.2018.04.001

Sarkar, S., Chatterjee, S., & Misra, S. (2018). Assessment of the Suitability of Fog Computing in the Context of Internet of Things. *IEEE Transactions on Cloud Computing*, *6*(1), 46–59. https://doi.org/10.1109/TCC.2015.2485206

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, *8*(4), 14–23. https://doi.org/10.1109/MPRV.2009.82

Schilit, B. N., & Theimer, M. M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, *8*(5), 22–32. https://doi.org/10.1109/65.313011

Schmohl, R., & Baumgarten, U. (2008). A Generalized Context-aware Architecture in Heterogeneous Mobile Computing Environments. In *2008 The Fourth International Conference on Wireless and Mobile Communications* (pp. 118–124). IEEE. https://doi.org/10.1109/ICWMC.2008.59

Seddik, Y., & Hanzálek, Z. (2017). Match-up scheduling of mixed-criticality jobs: Maximizing the probability of jobs execution. *European Journal of Operational Research*, *262*(1), 46–59. https://doi.org/10.1016/j.ejor.2017.03.054

Sheikhalishahi, M., Wallace, R. M., Grandinetti, L., Vazquez-Poletti, J. L., & Guerriero, F. (2016). A multi-dimensional job scheduling. *Future Generation Computer Systems*, *54*, 123–131. https://doi.org/10.1016/j.future.2015.03.014

Shi, T., Yang, M., Li, X., Lei, Q., & Jiang, Y. (2016). An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. *Pervasive and Mobile Computing*, *27*, 90–105. https://doi.org/10.1016/j.pmcj.2015.07.005

Shojafar, M., Javanmardi, S., Abolfazli, S., & Cordeschi, N. (2015). FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Cluster Computing*, *18*(2), 829–844. https://doi.org/10.1007/s10586-014-0420-x

Silva, M. (2017). *Um modelo de gerenciamento da qualidade de experiência para a provisão de serviços cientes de contexto*. (Ph.D), Universidade Federal de Santa Catarina. Retrieved February 24, 2020, from https://repositorio.ufsc.br/handle/123456789/188666

Skarlat, O., Nardelli, M., Schulte, S., & Dustdar, S. (2017). Towards QoS-Aware Fog Service Placement. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)* (pp. 89–96). IEEE. https://doi.org/10.1109/ICFEC.2017.12

Soldani, D., Chiavelli, D., Laiho, J., Li, M., Muhammad, N., Giambiasi, G., & Rodriquez, C. (2006). QoE and QoS Monitoring. In *QoS and QoE Management in UMTS Cellular Systems* (pp. 315–384). Chichester, UK: John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470034057.ch9

Sousa, A. (2017). *Adaptação dinâmica e sensível ao contexto de interfaces móveis em ambiente ubíquo*. (Ph.D), Universidade de Trás-os-Montes e Alto Douro. Retrieved December 10, 2019, from https://catalogo.biblioteca.utad.pt/cgi-bin/koha/opac-detail.pl?biblionumber=72842

Soyata, T., Ba, H., Heinzelman, W., Kwon, M., & Shi, J. (2013). Accelerating Mobile-Cloud Computing. In *Communication Infrastructures for Cloud Computing* (Vol. 4, pp. 175–197). https://doi.org/10.4018/978-1-4666-4522-6.ch008

Stavrinides, G. L., & Karatza, H. D. (2019). A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments. *Multimedia Tools and Applications*, *78*(17), 24639–24655. https://doi.org/10.1007/s11042-018-7051-9

Stojmenovic, I. (2014). Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)* (pp. 117–122). IEEE. https://doi.org/10.1109/ATNAC.2014.7020884

Swaroop, P. (2019). *Cost Based Job Scheduling in Fog Computing*. (Ph.D), Delhi Technological University. Retrieved February 20, 2020, from http://dspace.dtu.ac.in:8080/jspui/handle/repository/16722

Taneja, M., & Davy, A. (2017). Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (pp. 1222–1228). IEEE. https://doi.org/10.23919/INM.2017.7987464

Taurion, C. (2009). *Computação Em Nuvem: Transformando o Mundo da Tecnologia da Informação*. (Brasport, Ed.). BRASPORT.

Tiwary, M., Puthal, D., Sahoo, K. S., Sahoo, B., & Yang, L. T. (2018). Response time optimization for cloudlets in Mobile Edge Computing. *Journal of Parallel and Distributed Computing*, *119*, 81–91. https://doi.org/10.1016/j.jpdc.2018.04.004

Vaquero, L. M., & Rodero-Merino, L. (2014). Finding your Way in the Fog. *ACM SIGCOMM Computer Communication Review*, *44*(5), 27–32. https://doi.org/10.1145/2677046.2677052

Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds. *ACM SIGCOMM Computer Communication Review*, *39*(1), 50–55. https://doi.org/10.1145/1496091.1496100

Wang, N., Varghese, B., Matthaiou, M., & Nikolopoulos, D. S. (2019). ENORM: A Framework For Edge NOde Resource Management. *IEEE Transactions on Services Computing*, 1–1. https://doi.org/10.1109/TSC.2017.2753775

Wang, X., Wang, Y., & Cui, Y. (2016). An energy-aware bi-level optimization model for multi-job scheduling problems under cloud computing. *Soft Computing*, *20*(1), 303–317. https://doi.org/10.1007/s00500-014-1506-3

Wold, S. (1994). Exponentially weighted moving principal components analysis and projections to latent structures. *Chemometrics and Intelligent Laboratory Systems*, *23*(1), 149–161. https://doi.org/10.1016/0169-7439(93)E0075-F

Woo, Y.-B., Jung, S., & Kim, B. S. (2017). A rule-based genetic algorithm with an improvement heuristic for unrelated parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Computers & Industrial*

Engineering, *109*, 179–190. https://doi.org/10.1016/j.cie.2017.05.007

Yang, Y., Zhao, S., Zhang, W., Chen, Y., Luo, X., & Wang, J. (2018). DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks. *IEEE Internet of Things Journal*, *5*(3), 2094–2106. https://doi.org/10.1109/JIOT.2018.2823000

Ye, Jihang, Zhu, Z., & Cheng, H. (2013). What's Your Next Move: User Activity Prediction in Location-based Social Networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining* (pp. 171–179). Philadelphia, PA: Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972832.19

Ye, Juan, Stevenson, G., & Dobson, S. (2011). A top-level ontology for smart environments. *Pervasive and Mobile Computing*, *7*(3), 359–378. https://doi.org/10.1016/j.pmcj.2011.02.002

Yi, S., Hao, Z., Qin, Z., & Li, Q. (2015). Fog Computing: Platform and Applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)* (pp. 73–78). IEEE. https://doi.org/10.1109/HotWeb.2015.22

Yi, S., Li, C., & Li, Q. (2015). A Survey of Fog Computing. In *Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata '15* (pp. 37–42). New York, New York, USA: ACM Press. https://doi.org/10.1145/2757384.2757397

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18. https://doi.org/10.1007/s13174-010-0007-6

Zhou, B., Dastjerdi, A. V., Calheiros, R. N., Srirama, S. N., & Buyya, R. (2017). mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud. *IEEE Transactions on Services Computing*, *10*(5), 797–810. https://doi.org/10.1109/TSC.2015.2511002

Zhou, X., Sun, M., Wang, Y., & Wu, X. (2015). A New QoE-Driven Video Cache Allocation Scheme for Mobile Cloud Server. In *Proceedings of the 11th EAI International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness* (pp. 122–126). IEEE. https://doi.org/10.4108/eai.19-8-2015.2260126

Zhu, C, Li, X., Leung, M., Hu, X., & Yang, T. (2015). Message from the Cloud Computing Association. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. xv–xv). IEEE. https://doi.org/10.1109/CloudCom.2015.106

Zhu, Chunsheng, Li, X., Ji, H., & Leung, V. C. M. (2015). Towards Integration of Wireless Sensor Networks and Cloud Computing. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 491–494). IEEE. https://doi.org/10.1109/CloudCom.2015.27

Zielinski, S. (2016). On Some Biases Encountered in Modern Audio Quality Listening Tests (Part 2): Selected Graphical Examples and Discussion. *Journal of the Audio Engineering Society*, *64*(1/2), 55–74. https://doi.org/10.17743/jaes.2015.0094

Zimmermann, A., Lorenz, A., & Oppermann, R. (2007). An Operational Definition of Context. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, & L. Vieu (Eds.), *Modeling and Using Context* (pp. 558–571). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74255-5_42

ANEXOS

ANEXO A: DESCRIÇÃO DOS PILARES DA *OPENFOG*

A arquitetura de referência *OpenFog* foi impulsionada por um conjunto de oito pilares:

Pilar da segurança, muitas aplicações suportadas pela plataforma *OpenFog*, tem aspetos críticos de privacidade, missão crítica e até vida crítica. Como tal, qualquer problema de segurança pode originar consequências graves. A arquitetura de referência *OpenFog*, sintetiza essas tecnologias, e possibilita a criação flexível de ambientes computacionais que abordam um amplo espetro de preocupações de segurança, desde dispositivos *IoT* a *cloud* e *fog*.

A conformidade com os requisitos da arquitetura de referência *OpenFog* garante que uma implementação será construída sobre um ambiente computacional que inclui a segurança do nó *OpenFog*, da rede *OpenFog*, e a gestão e segurança da orquestração *OpenFog*. Isto permitirá aos arquitetos e *designers* focalizarem nos problemas de segurança e privacidade de alto valor específicos para os tipos de dispositivos utilizados na sua aplicação.

Esse pilar começa com uma definição clara dos componentes de base. Todos os nós da *fog* devem utilizar uma raiz de confiança imutável com base em hardware. A raiz da confiança do hardware é um componente de hardware confiável que recebe controlo na ativação. Ela amplia a cadeia de confiança para os outros componentes de *hardware, firmware e software*. A raiz da confiança deve ser atestada por agentes de software que executam dentro e através da infraestrutura. Dada a proximidade à *edge network*, os nós da *fog* muitas vezes agem como o primeiro nó de controlo de acesso e encriptação. Isso significa, que eles devem proporcionar a integridade contextual e isolamento e controlar a agregação de dados sensível à privacidade antes que eles saiam da *edge network*.

Atendendo que topologias mais complexas possam ser criadas em implementações *fog*, como serviço (*Fog-as-a-Service*), a certificação continua como uma cadeia de confiança do nó da *fog*, para os outros nós da *fog* e para a *cloud*. Como esses nós podem ser instanciados ou eliminados dinamicamente, os recursos de *hardware* e *software* devem ser certificados. Os componentes que não foram certificados não devem ser totalmente autorizados a participar no nó da *fog* ou podem não ser considerados como tendo dados totalmente confiáveis.

Pilar da escalabilidade, este pilar responde às necessidades técnicas e comerciais dinâmicas das implementações *fog*. A escalabilidade elástica é transversal a todas as aplicações *fog*. A

sua propriedade hierárquica e a sua localização na *edge network* permitem oportunidades adicionais de escalabilidade:

- Nós individuais da *fog* podem escalar internamente, através do aumento de hardware ou software;
- Redes *fog* podem escalar verticalmente e horizontalmente, através da inserção de mais nós com objetivo de auxiliar os nós sobrecarregados, seja no mesmo nível da hierarquia ou em níveis adjacentes;
- Uma rede de nós da *fog* pode ser escalada tanto horizontalmente como verticalmente num ambiente elástico conforme a demanda;
- Armazenamento, conectividade da rede e serviços analíticos podem ser escalados com a infraestrutura da *fog*.

Devido à variabilidade de casos de utilização da *fog*, a arquitetura de referência *OpenFog* possibilita tanto implementações elásticas de pequena escala como grandes implementações de missão crítica conforme a procura. Os recursos podem ser configurados em instalações iniciais e reajustadas em nós da *fog* existente, conforme a necessidade.

Pilar da abertura, ser aberto é essencial para o sucesso do ecossistema ubíquo da *fog*. Soluções proprietárias ou em regime do monopólio podem levar a uma diversidade limitada de fornecedores, o que pode impactar negativamente no custo, qualidade e inovação do sistema. Segundo a OpenFog (2017), a importância do pilar aberto é salientada pelo desejo por sistemas totalmente interoperáveis, suportados por um ecossistema de fornecedores dinâmicos. Avançam ainda, que ser aberto como princípio, permite a existência de nós da *fog* em qualquer ponto de uma rede, redes de extensão e possibilita o agrupamento através de descobertas, o que significa que novos nós da *fog* definidas por software podem ser dinamicamente criados a fim de solucionar problemas de negócios. Este pilar, está intimamente relacionado com o pilar de segurança.

Pilar da autonomia, permite que nós da *fog* continuem o seu normal funcionamento em caso de falhas de serviços externos. Nesta arquitetura, a autonomia é suportada em toda a hierarquia. A tomada de decisão é feita em todos os níveis da hierarquia de uma implementação, tanto perto do dispositivo como em camadas superiores.

A arquitetura de referência *OpenFog* suporta autonomia para várias funcionalidades, ela não depende de uma entidade centralizada para operação (por exemplo, *backend* da *cloud*) e abarca algumas tarefas como: descoberta de registo de recursos, orquestração e gestão, segurança, operação de suporte a tomada de decisão, entre outros.

Pilar de Fiabilidade, Disponibilidade e Manutenibilidade (RAS), todas as arquiteturas de sistema bem-sucedidas garantem o RAS. Também, elas assumem uma grande importância na arquitetura de referência *OpenFog*.

O Hardware, software e operações são as três principais áreas do pilar RAS, que é especialmente importante em implementações da arquitetura de referência *OpenFog* em ambientes adversos e remotos. Por isso, aspetos do RAS são transversais a arquitetura.

Uma implementação é confiável se continuar a disponibilizar as suas funcionalidades mesmo em condições de funcionamento adversas. A fiabilidade do pilar RAS inclui, entre outros, às seguintes propriedades:

- Iniciar pedidos de manutenção preventiva, incluindo novos *patches* de hardware e software, redireccionamento de rede, entre outros;
- Garantir o funcionamento fiável do hardware sobre o qual o software opera, permitindo assim, o funcionamento seguro e resiliente do software. Uma *fog* fiável é normalmente medida com base no seu *uptime*;
- Proteger a disponibilidade e integridade dos dados a serem processados no *edge gateways* utilizando *hardware, software* e redes aprimorados;
- Aumentar a satisfação do cliente, através da simplificação do suporte e da auto-optimização e recuperação do dispositivo;
- Validar a plataforma e a arquitetura dos sistemas através de um conjunto de testes de certificação de interoperabilidade;
- Testar e validar componentes do sistema, incluindo *drivers* dos dispositivos e ferramentas de diagnósticos relacionados com um conjunto de condições do ambiente.
- Disponibilizar alertas, relatórios, *logs*, entre outros, que auxiliam na resolução de problemas.

A disponibilidade do pilar RAS, garante uma gestão e orquestração contínua, normalmente é medida em função do *uptime* e inclui propriedades como:

- Acesso seguro a todos os níveis de uma hierarquia *fog* para orquestração, gestão e controlo, o que inclui a possibilidade de atualização, diagnóstico e alteração do *firmware* de forma segura;
- Conceito de suporte *backend* baseado em *cloud* com disponibilidade de interfaces em todo o sistema,
- Suporte para configurações redundantes por forma a minimizar o *downtime*, isolamento de falhas, deteção de sintomas que conduzem as falhas por forma a melhorar o tempo médio de manutenção;
- Capacidade do arranque remoto da plataforma;

A manutenibilidade do pilar RAS, assegura a disponibilidade do serviço de uma implementação *fog* e possibilita o seu correto e normal funcionamento. Inclui, entre outros, as seguintes propriedades:

- Instalação, atualização e manutenção altamente automatizada a fim de possibilitar implementações *fog* eficiente e escalar;
- Capacidade do hardware ou software em caso de erros poder ser corrigido autonomamente ou por vários fabricantes;
- Fácil utilização e manutenção;
- Suporte as configurações redundantes visando produtividade contínua.

Pilar da agilidade, aborda decisões operacionais do negócio para uma implementação da arquitetura de referência *OpenFog*. A IoT, proporcionou disponibilidade de dados cujas análises para as tomadas de decisões em tempos oportunos ultrapassam as capacidades humanas (OpenFog, 2017). Esse pilar se concentra na transformação dessa grande quantidade de dados em *insights* acionáveis. Ela lida também com a natureza altamente dinâmica das implementações *fog* e com a necessidade de responder rapidamente às mudanças.

Pilar da hierarquia, a hierarquia computacional e do sistema não são obrigatórias em todas arquiteturas de referência *OpenFog*. No entanto, elas são expressas em maioria das

implementações. Segundo OpenFog (2017), a arquitetura *OpenFog* complementa as tradicionais arquiteturas *cloud,* em parte, devido ao pilar hierárquico da *OpenFog*.

Os recursos de processamento da arquitetura de referência *OpenFog* podem ser vistos como uma hierarquia lógica que se baseia nos requisitos funcionais de um sistema. Dependendo da escala e natureza do cenário a ser abordado, a hierarquia pode ser uma rede de sistemas particionados inteligentes e conectados dispostos em camadas físicas ou lógicas, ou pode colapsar num único sistema físico (pilar de escalabilidade).

Pilar de programabilidade, este pilar permite implementações altamente adaptativas que incluem o suporte para programação nas camadas de software e hardware. Isto possibilita que a reprogramação de um nó *fog* ou cluster de nós *fog* para a acomodação da dinâmica operacional possa ser completamente automatizada. A reprogramação pode ser feita com a ajuda dos nós da *fog* inerente às interfaces de programabilidade, que descrevemos utilizando interfaces computacionais de propósito geral ou aceleradores. A programabilidade permite benefícios como: infraestrutura adaptável, implementações com mais eficiência de recursos, operações mais económicas, segurança mais aprimorada, entre outros.

ANEXO B: Componentes do *iFogSim* e suas políticas de afetação de módulos

Conforme Mestre *et al.* (2019), um simulador é geralmente composto por três componentes principais:

O componente físico, compreende os dispositivos da rede (sensores, atuadores, dispositivos móveis, *cloudlets, cloud, fog*, entre outros), sua organização hierárquica, os seus recursos computacionais e latência de comunicação entre eles.

O componente lógico, consiste na representação das aplicações que serão executadas. Tais aplicações são divididas em módulos e são necessárias alguma capacidade computacional para as executar. Podem ser afetadas aos diferentes dispositivos.

Segundo Gupta *et al.* (2016), as aplicações desenvolvidas para serem implementadas na *fog* funcionam com base no modelo *Distributed Data Flow* (DDF), e podem ser apresentadas através de um gráfico direcionado, onde os vértices representam os módulos da aplicação e as arestas direcionadas ilustram o fluxo de dados entre os módulos.

Para Mestre *et al.* (2019), as arestas permitem ligar o resultado de um módulo à entrada de outro. Afiançam ainda, que elas possuem duas orientações possíveis: *Up* e *Down*. Ela é do tipo *Up,* se enviar dados para serem executados no módulo seguinte e do tipo *Down,* se enviar dados já executados para o modulo destino que está a aguardar a resposta. Possuem entre outras, informações relacionadas com a quantidade de *bytes* que estão a ser transferidos entre os módulos. Essa transferência pode ser periódica ou baseada em eventos. Isto é, assim que os dados de uma aresta que incide sobre um módulo chegam, o módulo produz dados para serem encaminhados à próxima aresta. O simulador permite definir a forma como é feita a seleção dos módulos que recebem os dados da aresta e a forma como os dados são produzidos para serem encaminhados à aresta seguinte.

O cálculo do tempo de execução das aplicações é feito através do *delay* confirmados em ciclos de comunicação entre os diferentes módulos. Isto é, o tempo médio do início da execução do primeiro módulo do ciclo até a conclusão do último.

O componente de gestão, é o responsável pelo início da execução dos módulos, estatísticas da simulação e afetação dos módulos. Para esse fim, considera: os recursos solicitados e disponíveis, a prioridade da aplicação e a latência de comunicação.

A afetação dos módulos para serem executados no *iFogSim* é feita através das políticas *standards*. No entanto, podemos idealizar, melhorar e/ou propor outras abordagens.

Segundo Gupta *et al.* (2016), o *iFogSim* possui duas políticas encarregues para a afetação de módulos: *cloud-only placement* e *edge-ward placement*. Já Mestre *et al.* (2019), identificam três políticas para o *iFogSim: placement mapping, placement cloud only* e *placement edgewards*.

Atendendo que as políticas responsáveis para a afetação dos módulos identificados são complementares, descrevemos as identificadas em Mestre *et al.* (2019), que consideramos serem mais abrangentes.

Module Placement Mapping, esta política de afetação segue estritamente o mapeamento disponibilizado no *controller* do *iFogSim* quando este é criado, independente dos recursos computacionais disponíveis. Por conseguinte, se um módulo de uma aplicação for mapeado para uma *cloudlet* específica, mas esta já estiver a executar outras aplicações ou outras instâncias da mesma aplicação, e não houver recursos suficientes, este módulo permanece afetado a ela e, terá de aguardar o término da execução dos demais para então ser executado.

Module Placement Cloud Only, esta política visa resolver o principal problema da política de afetação *Placement Mapping*, que ocorre quando aumentam a quantidade de módulos afetados a uma mesma *cloudlet* e faz com que esta *cloudlet* ultrapasse a capacidade suportada.

Na política *Placement Cloud Only*, atendendo que a *cloud* é o dispositivo mais no topo da hierarquia e detém a capacidade de processamento muito superior que os demais dispositivos e considerada como "ilimitada", ela faz a afetação de todos os módulos da aplicação diretamente para a *cloud*. Este cenário simula o caso em que não há dispositivos *fog* na rede e os utilizadores enviam dados diretamente para a *cloud*. Esta abordagem apresenta inconveniências, cuja a principal prende-se com o aumento da latência, uma vez que a *cloud* é centralizada e se encontra muito mais distante dos dispositivos móveis do que dos dispositivos intermediários da hierarquia. Consequentemente, a latência de comunicação é consideravelmente maior.

Module Placement Edgewards, esta política de afetação combina as políticas *Module Placement Mapping* e *Module Placement Cloud Only* e visa resolver as suas inconveniências. Afeta os módulos nos *devices* com base no mapeamento disponibilizado

pelo *controller* do *iFogSim*, no entanto, antes de afetar o módulo, verifica se o dispositivo possui recursos suficientes para servir os seus requisitos. Caso o dispositivo não tenha capacidade suficiente, o módulo é encaminhado para o dispositivo logo acima na hierarquia (dispositivo pai).

Além disso, se o dispositivo filho possuir outras instâncias do módulo afetado, estas também são transferidas. E, ainda, se houver instâncias dos módulos, no dispositivo filho, que dependem do resultado de um dos módulos que estão a ser transferidos, essas instâncias também são transferidas. Faz-se a repetição desse processo até não houver mais módulos para serem transferidos.

> "Se o dispositivo pai não for capaz de atender a todos os módulos que foram transferidas para ele, o algoritmo se repete e envia os módulos ainda mais para o topo da hierarquia até que possam ser afetados. Assim sendo, no pior cenário, todos os módulos serão transferidos sucessivamente para o topo até chegarem à *cloud* e neste caso terá o mesmo resultado que a política *cloud only.*"
>
> (Mestre *et al.*, 2019:12)

APÊNDICES

APÊNDICE A – LISTA DE PUBLICAÇÕES ORIGINAIS

Durante os trabalhos relacionados com esta tese de doutoramento, foram publicados os seguintes artigos originais:

Barros, C., Rocio V., Sousa A. & Paredes H. (2019). *Architecture and platform for Fog Computing* 2019 7th International Workshop on ADVANCEs in ICT Infrastructures and Services, ADVANCEs 2019, Praia, Cabo Verde, 2019, https://easychair.org/publications/preprint_open/RvJ9;

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020). Scheduling in Cloud and Fog Architecture: Identification of Limitations and Suggestion of Improvement Perspectives. Journal of Information Systems Engineering and Management, 5(3), em0121. https://doi.org/10.29333/jisem/8429;

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020). Survey on Job Scheduling in Cloud-Fog Architecture. In 2020 15th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-7). IEEE. https://doi.org/10.23919/CISTI49556.2020.9141156;

Barros C., Rocio V., Sousa A., Paredes H. (2020) Context-Aware Mobile Applications in Fog Infrastructure: A Literature Review. In: Rocha Á., Adeli H., Reis L., Costanzo S., Orovic I., Moreira F. (eds) Trends and Innovations in Information Systems and Technologies. WorldCIST 2020. Advances in Intelligent Systems and Computing, vol 1160. Springer, Cham. https://doi.org/10.1007/978-3-030-45691-7_29.

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020). Uma Proposta de Algoritmos de Escalonamento de Aplicações Móveis Sensível ao Contexto para o Paradigma Fog Computing. Revista das Ciências da Computação da Universidade Aberta. https://doi.org/10.34627/rcc.v15i0.

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2020). Job Scheduling in Fog Paradigm - A Proposal of Context-aware Task Scheduling Algorithms. International Conference on Information Technology Systems and Innovation (ICITSI), Bandung - Padang, Indonesia, 2020, pp. 451-458, https://doi.org/10.1109/ICITSI50517.2020.9264945.

Barros, C., Rocio V., Sousa A. & Paredes H. (2021). Task Scheduling Algorithms for Fog paradigm. 2021 9th International Workshop on ADVANCEs in ICT Infrastructures and Services, ADVANCE' 2021, Zaragoza, Spain, 2-4 February 2021, ISSN: 2237-2901, https://doi.org/10.48545/advance2021-shortpapers-2.

APÊNDICE B - LISTA DE PUBLICAÇÕES EM ANÁLISE PARA PUBLICAÇÃO

O seguinte artigo encontra-se em análise para a publicação:

Barros, C., Rocio, V., Sousa, A., & Paredes, H. (2021). A proposal of context-aware task scheduling algorithms for fog architecture. Revista Ibérica de Sistemas e Tecnologias de Informação (RISTI) ISSN: 1646-9895;