UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Descarga adaptativa em ambiente com névoa heterogênea: estudo de caso para a área da saúde

José Rodrigues Torres Neto

Tese de Doutorado do Programa de Pós-Graduação em Ciências de
Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

José Rodrigues Torres Neto

Descarga adaptativa em ambiente com névoa heterogênea: estudo de caso para a área da saúde

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Jó Ueyama

USP – São Carlos
Fevereiro de 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

T693d   Torres Neto, José Rodrigues
              Descarga adaptativa em ambiente com névoa
              heterogênea: estudo de caso para a área da saúde /
              José Rodrigues Torres Neto; orientador JÓ Ueyama. --
              São Carlos, 2020.
              118 p.

              Tese (Doutorado - Programa de Pós-Graduação em
              Ciências de Computação e Matemática Computacional) --
              Instituto de Ciências Matemáticas e de Computação,
              Universidade de São Paulo, 2020.

              1. Computação em Névoa. 2. Internet das Coisas.
              3. Descarregamento de Dados. 4. Cuidados de Saúde.
              I. Ueyama, JÓ, orient. II. Título.

José Rodrigues Torres Neto

**Adaptive offloading in heterogeneous fog environment: a
case study in healthcare**

Thesis submitted to the Institute of Mathematics
and Computer Sciences – ICMC-USP – in
accordance with the requirements of the Computer
and Mathematical Sciences Graduate Program, for
the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Jó Ueyama

USP – São Carlos
February 2020

*Dedico este trabalho a Deus por me sustentar e me conduzir na minha jornada pessoal,
acadêmica e profissional.*

*Aos meus pais Rosário e Miguel Torres e aos meus irmãos Rennan e Aluana pelo apoio, amor e
admiração. Obrigado por me ensinarem o que é a gratidão e a perseverança.*

*À minha esposa Willyane Alvarenga pelo seu amor e pelas suas palavras de apoio e incentivo,
que foram fundamentais nessa jornada.*

AGRADECIMENTOS

Ao meu orientador, **Professor Dr. Jó Ueyma**, pela oportunidade de realizar este trabalho e pelo aprendizado diário. A sua orientação e encorajamento tornaram esta trajetória gratificante.

Ao **Instituto de Ciências Matemáticas e de Computação (ICMC)** da **Universidade de São Paulo (USP)**, especialmente, ao Laboratório de Sistemas Interativos Web e de Sistemas Multimídia (Intermídia) pelo acolhimento, infraestrutura para o desenvolvimento desta tese e qualidade da minha formação acadêmica. Um agradecimento especial aos meus colegas do *Wireless Networks Group (WinesGroup)* pelo acolhimento, disponibilidade, aprendizado diário e grande amizade.

À *University of Ottawa* e à *School of Electrical Engineering and Computer Science* pela oportunidade de realizar um estágio de pesquisa no exterior e pela infraestrutura para o desenvolvimento de parte deste projeto. Um agradecimento especial ao meu supervisor, **Dr. Azzedine Boukerche**, pela orientação e apoio, e aos colegas integrantes do *PARADISE Research Laboratory* pelo acolhimento e aprendizado.

Aos Professores Doutores **Gustavo Pessin, João Luis Garcia Rosa, Rossana Maria de Castro Andrade e Kalinka Regina Lucas Jaquie Castelo** pelas contribuições valiosas durante o meu Exame de Qualificação.

Aos Professores Doutores **Júlio Cezar Estrella, Cláudio Fabiano Motta Toledo, Edmundo Roberto Mauro Madeira e Raphael Yokoingawa de Camargo** por aceitarem integrar a comissão julgadora desta tese. Agradeço pelas valiosas contribuições.

À Coordenação de Aperfeiçoamento de Pessoal de nível Superior (**CAPES**) - **Código de Financiamento 001**, pelo apoio financeiro concedido na forma de Bolsa de Doutorado no início do curso de pós-graduação.

À **Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)** pelo apoio financeiro, a partir de uma Bolsa de Doutorado - **processo nº 2016/25865-2**, usufruída durante a maior parte do curso de pós-graduação. Agradeço-a, também, pela concessão de uma Bolsa Estágio de Pesquisa no Exterior (BEPE) - **processo nº 2017/23655-3**, destinada aos bolsistas de Doutorado da FAPESP, que me oportunizou a realização do estágio de pesquisa na *University of Ottawa*.

A todos os meus **amigos** que contribuíram com seu apoio e incentivo para eu concluir essa etapa muito importante na minha vida, especialmente, aos da República Meio Ácido que me acolheram nos momentos que precisei.

*“Ninguém caminha sem aprender a
caminhar, sem aprender a fazer o caminho
caminhando, refazendo e retocando o sonho
pelo qual se pôs a caminhar.”
(Paulo Freire)*

RESUMO

TORRES NETO, J. R. **Descarga adaptativa em ambiente com névoa heterogênea: estudo de caso para a área da saúde.** 2020. 115 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

O número de pessoas com necessidades de cuidados de saúde, tais como idosos, deficientes e pacientes com mobilidade reduzida, aumentou consideravelmente no mundo nos últimos anos. Esse aumento criou uma tendência na pesquisa envolvendo cuidados de saúde desses pacientes por meio de aplicações baseadas em IoT, principalmente quando recebem alta do hospital e retornam para casa. No entanto, aplicações de saúde são consideradas críticas e intolerantes aos atrasos, que exigem respostas em tempo real. Neste contexto, é comum que arquiteturas tradicionais baseadas em nuvem possuam ocorrência de desconexão da rede principal ou variações de largura de banda e latência, gerando prejuízos aos serviços remotos, principalmente serviços de saúde. Diante disso, a recém-criada computação em névoa permite o compartilhamento e gestão de dados nas bordas da rede, tornando o processamento inteligente descentralizado mais próximo da fonte de dados e dos usuários, o que garante uma latência mais baixa. Esta evolução contínua implica que os sistemas de software devem tornar-se cada vez mais versáteis e flexíveis, adaptando-se ao contexto de ambientes operacionais e intermitentes. O principal objetivo desta pesquisa de doutorado é mitigar o uso de recursos em sistemas de tempo-real em ambientes IoT. Neste sentido, é proposta uma arquitetura auto adaptativa e um *middleware* autônomo baseado em ciclos de *feedback* para a descarga dinâmica de dados em arquiteturas de névoa multicamadas *Fog-to-Cloud* e *Indie Fog*. Para isso, uma nova política para descarga dinâmica de dados foi definida. Já o paradigma *publish-subscribe* foi utilizado para a comunicação assíncrona e o MAPE-K como referência para o ciclo autônomo. Foi realizado uma avaliação quantitativa do *middleware* por meio da simulação de uma casa inteligente com duas aplicações de saúde: (i) monitoramento cardíaco e (ii) controle de temperatura corporal. Ainda, um estudo de caso real no sensoriamento do estado físico e emocional do usuário foi realizado com as aplicações: (i) reconhecimento facial do usuário, (ii) detecção de quedas, e (iii) reconhecimento de emoção por voz. Os resultados das avaliações quantitativa e qualitativa demonstraram uma melhora considerável no desempenho e a mitigação do uso de recursos. O *middleware* se apresenta promissor para o desenvolvimento de sistemas auto adaptativos no contexto de sistemas de tempo real e críticos.

Palavras-chave: Internet das Coisas, Computação em Névoa, Descarga de Dados, Cuidados de Saúde.

ABSTRACT

TORRES NETO, J. R. **Adaptive offloading in heterogeneous fog environment: a case study in healthcare.** 2020. 115 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

The number of people with healthcare needs, such as elderly, disabled, and patients with reduced mobility, has been increased considerably in the world in recent years. This increase has created a trend in research involving healthcare for these patients through IoT-based applications, especially when they are liberated from the hospital and returned home. However, healthcare applications are considered critical and intolerant of delays, which require real-time responses. In this context, it is common for traditional cloud-based architectures to experience disconnection from the leading network or variations in bandwidth and latency, which generate losses to remote services, especially healthcare services. Consequently, the newly created fog computing enables data sharing and management at network edges, bringing intelligent decentralized processing closer to the data source and users, which ensures low latency. This continuous evolution implies that software systems must become increasingly versatile and flexible, self-adapting to the context of operating and intermittent environments. The main goal of this doctoral research is to mitigate the use of resources in real-time systems in IoT environments. In this context, a self-adaptive architecture and an autonomous middleware based on feedback cycles for dynamic data offloading in Fog-to-Cloud and Indie Fog multi-layer fog architectures are proposed. For this, a new policy for dynamic data offloading has been defined. The publish-subscribe paradigm was used for reliable communication and MAPE-K as a reference for the autonomous cycle. A quantitative evaluation of the middleware was performed through the simulation of a smarthome with two health applications: (i) heart monitoring and (ii) body temperature control. Also, a real case study on the sensing of the user's physical and emotional state was performed with the applications: (i) facial recognition of the user, (ii) detection of falls, and (iii) recognition of emotion by voice. The results of the quantitative and qualitative evaluation have shown considerable improvement in performance and mitigation of resource use. The middleware is promising for the development of self-adaptive systems in the context of real-time and critical systems.

Keywords: Internet of Things, Fog Computing, Data Offloading, Healthcare.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquiteturas distintas aplicadas a IoT: A primeira utilizando somente a nuvem, enquanto que a segunda utiliza computação em névoa (Adaptado de (SOUZA <i>et al.</i> , 2018b))	26
Figura 2 – Arquiteturas de computação em névoa (Adaptado de (SOUZA <i>et al.</i> , 2018b))	36
Figura 3 – Autonomic Control Loop (BRUN <i>et al.</i> , 2009).	39
Figura 4 – MAPE-K (IBM, 2005).	41
Figura 5 – Ciclo de Controle de <i>Feedback</i> (BRUN <i>et al.</i> , 2009)	41
Figura 6 – Arquitetura em camadas hierárquicas de <i>Fog-to-Cloud</i> (Adaptado de Group <i>et al.</i> (2017))	57
Figura 7 – LOADING em uma arquitetura F2C para ambientes IoT	59
Figura 8 – MAPE-K (IBM, 2005).	60
Figura 9 – Fluxograma do algoritmo da decisão de descarga de dados	61
Figura 10 – Diagrama de classes do Framework AMOk	63
Figura 11 – Barramento de eventos	68
Figura 12 – Fluxograma de comunicação	69
Figura 13 – Bloco de dados utilizados.	70
Figura 14 – Impacto de desempenho da energia consumida	71
Figura 15 – Impacto de desempenho do tempo de resposta	71
Figura 16 – Impacto de desempenho do total de transmissão	72
Figura 17 – Monitoramento de saúde em ambientes IoT	76
Figura 18 – Monitoramento remoto individualizado	77
Figura 19 – Processo de mapeamento da face para o reconhecimento dos usuários (Adaptado de Mano <i>et al.</i> (2016))	78
Figura 20 – Aplicação para a detecção de queda (SYSTEM, 2015)	79
Figura 21 – Porcentagem de Acurácia das aplicações.	80
Figura 22 – Porcentagem de Acurácia das aplicações.	82
Figura 23 – Espectrogramas representando a frase “A cachoeira faz muito barulho.” para as seis emoções básicas.	83
Figura 24 – Aplicação de reconhecimento de emoções pela voz em um cenário de simulação clínica.	84
Figura 25 – Porcentagem de Acurácia da aplicação de reconhecimento de emoções pela VOZ.	86
Figura 26 – Tempo de processamento da aplicação de reconhecimento do usuário	89

Figura 27 – Tempo de processamento da aplicação de detecção de quedas	89
Figura 28 – Tempo de processamento da aplicação de reconhecimento de emoções pela voz	90
Figura 29 – Desempenho de consumo de memória da aplicação de reconhecimento de usuário	91
Figura 30 – Desempenho de consumo de memória da aplicação de detecção de queda . .	91
Figura 31 – Desempenho do consumo de memória da aplicação de reconhecimento de emoções pela voz	92
Figura 32 – Desempenho do consumo de processamento da aplicação de reconhecimento do usuário	92
Figura 33 – Desempenho do consumo de processamento da aplicação de detecção de queda	93
Figura 34 – Desempenho do consumo de processamento da aplicação de reconhecimento de emoções pela voz	93
Figura 35 – Impacto das aplicações no LOADING em relação ao descarregamento de dados	95
Figura 36 – Impacto das aplicações no LOADING em relação a quantidade de transmissões	96
Figura 37 – Impacto das aplicações no LOADING em relação ao consumo de energia . .	97

LISTA DE TABELAS

Tabela 1 – Síntese das principais características dos trabalhos relacionados	52
Tabela 2 – Cenário padrão de simulação	70
Tabela 3 – Média das acurácias obtidas pelos algoritmos para o reconhecimento do usuário	81
Tabela 4 – Média das acurácias obtidas pelos algoritmos para a detecção de quedas . .	81
Tabela 5 – Sentenças gravadas pelos atores e atrizes	85
Tabela 6 – Média das acurácias obtidas pelos algoritmos para o reconhecimento de emoção pela voz	86
Tabela 7 – Parâmetros de avaliação dos experimentos	87
Tabela 8 – Valores dos parâmetros para os cenários	88

LISTA DE ABREVIATURAS E SIGLAS

CPU	<i>Central Process Unit</i>
Customer Premises Equipments	CPEs
F2C	<i>Fog-to-Cloud</i>
HSH	<i>Health Smart Homes</i>
IaaS	<i>Infrastructure as a Service</i>
IoT	<i>Internet of Things</i>
ITS	<i>Intelligent Transportation Systems</i>
KNN	<i>k-Nearest Neighbor</i>
MCC	<i>Mobile Cloud Computing</i>
MEC	<i>Mobile Edge Computing</i>
MFCCs	<i>Mel-frequency cepstral coefficients</i>
MQTT	<i>Message Queue Telemetry Transport</i>
NFC	<i>Near Field Communication</i>
NIST	<i>National Institute of Standards and Technology</i>
PaaS	<i>Platform as a Service</i>
RFID	radiofrequência
RMI	<i>Remote Method Invocation</i>
RPC	<i>Remote Procedure Call</i>
RTOS	Sistemas Operacionais de Tempo Real
SaaS	<i>Software as a Service</i>
SVM	<i>Support Vector Machine</i>
VNF	<i>Virtual Network Functions</i>
WLAN	<i>Wireless Local Area Network</i>
WSN	<i>Wireless Sensor Network</i>

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Motivação	25
1.2	Problema de Pesquisa	28
1.3	Estrutura da Tese	29
2	FUNDAMENTAÇÃO TEÓRICA	31
2.1	Internet das coisas	31
2.2	Computação em nuvem	33
2.3	Computação em névoa	35
2.4	Computação Autônoma	37
2.4.1	<i>Propriedades de autogestão</i>	37
2.4.2	<i>Graus de automaticidade</i>	38
2.4.3	<i>Ciclos de feedback</i>	39
2.5	Descargamento de Dados	42
3	TRABALHOS RELACIONADOS	45
3.1	Ambientes inteligentes de IoT	45
3.2	Ambientes de névoa e descarga de dados	48
3.3	Considerações Finais	53
4	ARQUITETURA BASEADA EM CICLOS DE FEEDBACK E NÉVOA MULTICAMADA PARA A DESCARGA DINÂMICA DE DADOS	55
4.1	Arquitetura Proposta	55
4.2	LOADING - a middleware for Adaptive Data offloading	58
4.2.1	<i>AMOk - auto adaptive Mobile Offloading over Knowledge</i>	59
4.2.1.1	<i>Política de descarga de dados</i>	63
4.2.1.2	<i>Custo de computação local</i>	64
4.2.1.3	<i>Custo de computação remota</i>	64
4.2.2	<i>Conectividade e comunicação</i>	66
4.3	Avaliação quantitativa de desempenho	68
4.3.1	<i>Cenário modelado</i>	68
4.3.2	<i>Impacto dos resultados obtidos</i>	70
4.4	Considerações Finais	72

5	AVALIAÇÃO DE UM ESTUDO DE CASO NO SENSORIAMENTO DO ESTADO FÍSICO E EMOCIONAL DO USUÁRIO EM AMBIENTES INTELIGENTES	75
5.1	Modelo computacional para o sensoriamento do estado físico do usuário	77
5.1.1	<i>Aplicação de reconhecimento facial do usuário . . . . .</i>	77
5.1.2	<i>Aplicação de detecção de quedas . . . . .</i>	78
5.1.3	<i>Avaliação do modelo para o reconhecimento facial do usuário e detecção de queda . . . . .</i>	79
5.2	Modelo computacional para o sensoriamento do estado emocional do usuário	81
5.2.1	<i>Aplicação de reconhecimento de emoção pela voz . . . . .</i>	82
5.2.2	<i>Avaliação do modelo para a aplicação de reconhecimento de emoção pela voz . . . . .</i>	83
5.3	Avaliação do impacto das aplicações de monitoramento de saúde em ambientes IoT no middleware LOADING	86
5.3.1	<i>Avaliação da gerência de recursos . . . . .</i>	87
5.3.2	<i>Avaliação da infraestrutura de comunicação . . . . .</i>	94
5.4	Considerações Finais	97
6	CONSIDERAÇÕES FINAIS	99
6.1	Trabalhos Futuros	100
6.2	Principais publicações	100
6.2.1	<i>Publicações científicas geradas desta tese . . . . .</i>	101
6.2.2	<i>Publicações científicas complementares . . . . .</i>	101
6.2.3	<i>Produção técnica . . . . .</i>	102
	REFERÊNCIAS	103

INTRODUÇÃO

O número de pessoas com necessidades de cuidados de saúde, tais como idosos, deficientes e pacientes com mobilidade reduzida, aumentou consideravelmente no mundo nos últimos anos ([AL-FUQAHA et al., 2015](#); [MUTLAG et al., 2019](#)). Este aumento tem criado uma tendência ascendente na pesquisa envolvendo estas pessoas, especialmente quando elas têm alta dos hospitais e precisam ser cuidadas em casa. A transição de volta para casa após a alta hospitalar é um momento vulnerável para os pacientes. Neste contexto, as *Health Smart Homes* (HSH), também conhecidas como ambientes de Cuidados de Saúde em Casas Inteligentes, têm surgido como uma opção promissora para melhorar a qualidade de vida das pessoas tratadas em casa e fornecer conforto para pessoas com alguma deficiência, mobilidade reduzida e/ou idosos ([BENIGNO et al., 2014](#); [BANERJEE et al., 2015](#); [LIOUANE et al., 2016](#)). Tipicamente, o tratamento de pessoas com mobilidade reduzida dentro do contexto de HSH usa inteligência computacional para monitorá-las enquanto elas estão se recuperando em suas casas ([ROMERO et al., 2009](#)). Ambientes inteligentes compostos por componentes de computação capazes de auxiliar nos cuidados de saúde das pessoas são extremamente importantes, especialmente em países onde o número de pessoas com mobilidade reduzida é alto ([ROMERO et al., 2009](#); [TALEB](#); [BOTTAZZI](#); [NASSER, 2010](#)).

Neste contexto, a Internet das Coisas (do inglês, *Internet of Things* (IoT)) tem permitido um conjunto de aplicações no contexto de cidades inteligentes com o avanço na área de telecomunicações ([AL-FUQAHA et al., 2015](#); [SOUZA et al., 2018a](#)). A implantação de serviços IoT tem sido facilitada devido ao desenvolvimento de protocolos de comunicação sem fio emergentes, sensores de baixo custo, e comunicação de dados entre dispositivos mais inteligentes ([SOUZA et al., 2018b](#)). Até 2021, 1 milhão de dispositivos serão ativados por hora, o que também facilita o provimento de serviços não privativos ([PLUMMER, 2015](#); [AL-FUQAHA et al., 2015](#)). Com isso, arquiteturas e *frameworks* no contexto de IoT tem inspirado uma infinidade de serviços distintos adaptados a diversos cenários, tais como Sistemas de Transportes Inteligentes - *Intelligent Transportation Systems* (ITS) ([SOUZA et al., 2018a](#)), cidades inteligentes ([RODRIGUES](#)

et al., 2017), monitoramento ambiental (SALMAN *et al.*, 2019), casas inteligentes (FILHO *et al.*, 2020), assistentes de saúde pessoais (CROATTI *et al.*, 2019) e cuidados de saúde (do inglês, *healthcare*) (NETO *et al.*, 2017), entre outros.

Pelo que é conhecido na literatura, o desenvolvimento das arquiteturas e *frameworks* IoT para cuidados de saúde tem sido crescente em consequência da forte contribuição da evolução da computação em nuvem e da recém-criada computação em névoa (do inglês, *fog computing*), a qual está intimamente relacionada ao conceito de Computação Móvel de Borda - *Mobile Edge Computing* (MEC), onde os elementos de computação são colocados mais próximos do usuário final (PATEL *et al.*, 2014; SOUZA *et al.*, 2018b). Computação em névoa permite o compartilhamento e gestão de dados nas bordas da rede, tornando o processamento inteligente descentralizado mais próximo da fonte de dados e dos usuários (SAQIB; HAMID, 2016), o que garante uma baixa latência e respostas mais rápidas em sistemas de tempo real. Enquanto que a nuvem é composta de centro de dados com alto poder de processamento, facilitando a implantação de serviços IoT com a distribuição de dados sob demanda em ambientes inteligentes, a névoa vem para atender demandas de serviços específicos não atendidos pela nuvem, tais como latência (SOUZA *et al.*, 2018b; MUTLAG *et al.*, 2019).

Diante disso, é possível perceber que os dois modelos de computação são altamente complementares entre si, o que abre espaço para o surgimento de uma nova tendência de paradigma que é a combinação das características da nuvem com a névoa. Em outras palavras, uma camada de computação com poder de processamento maior que um nó de névoa comum e com latência menor que a nuvem, reduzindo as fraquezas individuais de cada uma (SOUZA *et al.*, 2018b). Neste sentido, Masip-Bruin *et al.* (2016) propuseram o modelo de névoa *Fog-to-Cloud* (F2C), uma arquitetura de gerenciamento de névoa que reúne múltiplas camadas hierarquicamente, a qual de forma colaborativa, distribui os recursos de borda da névoa para a nuvem (SOUZA *et al.*, 2018b). Por exemplo, os sistemas de monitoramento de saúde possuem aplicações críticas sensíveis a latência, consequentemente necessitam de um grau maior de acesso e robustez (AZIMI *et al.*, 2017). Assim, é desafiador projetar um conjunto distribuído de recursos e a forma como eles devem ser executados (MASIP-BRUIN *et al.*, 2016; SOUZA *et al.*, 2018b). Em vista disso, uma arquitetura hierárquica de névoa multicamadas que atenda tanto o alto poder de processamento da nuvem, quanto a baixa latência na comunicação em névoa, é uma solução promissora (AZIMI *et al.*, 2017). Além disso, a computação em névoa multicamadas abre espaço e oportunidades para a investigação de problemas como o descarregamento de dados (do inglês, *data offloading*) entre as camadas, provimento de recursos, consumo de energia, segurança, e privacidade (SOUZA *et al.*, 2018b; ZHOU *et al.*, 2018; AAZAM; ZEADALLY; HARRAS, 2018b).

Uma das limitações da arquitetura de névoa multicamadas é a respeito do uso de equipamentos e recursos que funcionam de forma exclusiva para serviços específicos em redes móveis ou de provedores de serviços. Para reduzir este problema, utilizamos o conceito *Indie*

Fog (CHANG; SRIRAMA; BUYYA, 2017), o qual indica utilizar, como parte da arquitetura de névoa, os próprios dispositivos dos usuários, denominados CPEs (Customer Premises Equipments). Este modelo permite maior interoperabilidade dos serviços e integração entre redes privadas e públicas, podendo ser implementado de duas formas: (1) integrada, onde os serviços são fornecidos por um *gateway* para os dispositivos; e (2) colaborativa, onde os serviços são fornecidos por uma estação de trabalho na mesma rede dos dispositivos. Para isto, foi desenvolvido um *middleware* distribuído baseado em *Indie Fog* colaborativa e no paradigma de comunicação *publish-subscribe* para gerenciar a conexão e comunicação desacoplada entre os dispositivos locais do usuário e o servidor de névoa.

Ainda, com a tecnologia em constante evolução e a crescente demanda por software em uma ampla gama de segmentos, os sistemas de computação têm operado em ambientes em constante mudança, resultando em uma sobrecarga administrativa cada vez maior para o gerenciamento desses sistemas (AZIMI *et al.*, 2017). Esta evolução contínua implica que os sistemas de software devem tornar-se cada vez mais versáteis e flexíveis, eficientes em termos energéticos, tolerantes a falhas, configuráveis e auto adaptáveis, adaptando-se ao contexto de ambientes operacionais (GIUNTINI; BEDER; UEYAMA, 2017). Sistemas adaptativos podem ajustar seu comportamento em resposta à sua percepção do ambiente e do próprio sistema (CHENG *et al.*, 2009). Para controlar o comportamento dinâmico e as razões individuais do sistema, seu estado e ambiente, as decisões de projeto em sistemas auto adaptáveis precisam ser feitas em tempo real (SOUZA *et al.*, 2018b). Os sistemas de tempo real ou sistemas de saúde claramente devem considerar tais requisitos, uma vez que são considerados como sistemas críticos e intolerantes a atrasos por causa de ocorrências de urgência para o paciente (AZIMI *et al.*, 2017). Em vista disso, ciclos de *feedback* fornecem a auto adaptação essencial em relação a um objeto gerenciado, pois são ciclos projetados para coletar observações do objeto gerenciado, planejar uma execução ou alteração no sistema e executar uma ação (GARLAN *et al.*, 2004).

1.1 Motivação

Computação em névoa permite que os serviços dos usuários se beneficiem pela proximidade dos recursos, diminuindo a demanda pelos recursos da nuvem. Com os recursos computacionais presentes nas bordas da rede (i.e., próximo ao usuário final), além de reduzir a latência, também reduz o tempo de resposta total de serviços, o congestionamento da rede e o consumo de energia, além de trazer aspectos benéficos a segurança do sistema e privacidade do usuário (ETEMAD; AAZAM; ST-HILAIRE, 2017; SOUZA *et al.*, 2018b). A Figura 1 mostra a vantagem que a computação em névoa tem em relação a nuvem, no que diz respeito ao atraso na comunicação em uma aplicação convencional de cidades inteligentes que utiliza dispositivos inteligentes (i.e., sensores, dispositivos móveis, carros, entre outros). Neste cenário, computação em névoa se beneficia ao fornecer um nó de névoa de baixo atraso mais próximo do usuário final, capaz de corresponder a demanda dos serviços de forma adequada e favorecer Sistemas

Operacionais de Tempo Real (RTOS) (da sigla anglo-saxónica *Real Time Operating System*).

Figura 1 – Arquiteturas distintas aplicadas a IoT: A primeira utilizando somente a nuvem, enquanto que a segunda utiliza computação em névoa (Adaptado de (SOUZA *et al.*, 2018b))

A combinação de névoa e nuvem, baseada em multicamadas, permite que toda estrutura de computação disponível seja utilizada através das múltiplas camadas, melhorando e distribuindo o processamento de sistemas, o que ultrapassa as abordagens tradicionais (MUTLAG *et al.*, 2019). As arquiteturas hierárquicas fornecem múltiplas camadas que aumentam o poder computacional gradativamente até a nuvem, permitindo que tarefas robustas de processamento sejam executadas em camadas de névoa e não seja necessário descarregá-las para a nuvem (SOUZA *et al.*, 2018b). Além disso, o compartilhamento de recursos em uma arquitetura multicamadas acelera o tempo de resposta e reduz o desperdício de recursos (HE *et al.*, 2017). Por exemplo, a aplicação de um serviço de detecção de infarto em tempo real em ambientes inteligentes utilizando computação em névoa, é considerada uma aplicação importante em sistemas de saúde, uma vez que possui uma baixa taxa de erro (ALI; GHAZAL, 2017). Outro exemplo, é uma aplicação de detecção de queda ao identificar a queda de idosos ou um AVC em usuários, que são cenários em que é necessário se ter uma resposta em tempo real do sistema de saúde. Neste contexto, sensores médicos são conectados por meio de *gateways* inteligentes presentes em uma camada de névoa para o gerenciamento do conjunto de dispositivos, compartilhamento de recursos nas bordas, e autenticação segura e mútua entre o *gateway* de saúde e o cliente, garantindo latência mínima e consciência do contexto do ambiente (RAJAGOPALAN *et al.*, 2017; CERINA *et al.*, 2017; NETO *et al.*, 2019). Além disso, a névoa multicamadas dá suporte ao pré-processamento de dados antes de serem enviados para a nuvem, melhorando o tráfego da rede, o tempo de resposta, o uso da largura de banda e o consumo de energia (ESCAMILLA-AMBROSIO *et al.*, 2018).

A computação em névoa possui duas características relacionadas aos dispositivos de borda que são consideradas como desafios que podem dificultar a sua aplicabilidade (SOUZA *et al.*, 2018b; MUTLAG *et al.*, 2019). Estas características são a capacidade de processamento limitada e a conectividade intermitente, as quais afetam diretamente na disponibilidade dos recursos de névoa (SOUZA *et al.*, 2018b; MUTLAG *et al.*, 2019). Em contrapartida, os serviços de computação requerem alto poder de processamento de dados em larga escala, como em computação em nuvem. No entanto, os serviços sensíveis a latência devem ser levados em consideração, o que torna a computação em nuvem e névoa complementares entre si (SOUZA *et al.*, 2018b). Além disso, a terceirização completa da análise de dados para esta camada de névoa pode, consequentemente, dar origem a um nível reduzido de precisão e adaptabilidade (MUTLAG *et al.*, 2019). Por outro lado, a comunicação dos dispositivos é mais custosa do que a computação local, uma vez que os dispositivos de uma HSH podem consumir uma grande quantidade de energia se a sua comunicação não for otimizada (MOHAN; KANGASHARJU, 2016; STOJKOSKA; TRIVODALIEV, 2017). Por exemplo, um cenário com múltiplas aplicações competindo por recursos limitados resultam em maior latência de processamento e mais transmissões para a segunda camada de névoa ou, principalmente, para a nuvem. Desse modo, a transmissão de todos os dados coletados nesta camada reduz o desempenho dos serviços de rede e aumenta o consumo de energia dos dispositivos, o que pode prejudicar as decisões em sistemas críticos de saúde (MOHAN; KANGASHARJU, 2016; MUTLAG *et al.*, 2019). Gerenciar esses recursos não é uma tarefa trivial.

Diante disso, sistemas auto adaptáveis têm sido desenvolvidos nas mais diversas áreas, teorias, métodos e ferramentas para atender a demanda constante por softwares capazes de reagir ao contexto, em tempo real (GIUNTINI; BEDER; UYAMA, 2017). Além disso, sistemas de tempo real possuem a necessidade de responder às mudanças e alterações constantes, além de dar suporte a intermitência e disponibilidade de recursos. Embora a geração de um grande volume de dados seja inevitável, o pré-processamento inteligente do fluxo de dados nas bordas da rede pode mitigar a quantidade de dados sendo trafegados no ambiente (MASIP-BRUIN *et al.*, 2016; AZIMI *et al.*, 2017).

Esta pesquisa de doutorado teve como objetivo responder à seguinte questão de pesquisa:

“Como mitigar o uso de recursos em sistemas de tempo real em ambientes IoT, considerando que eles tendem a ser voláteis?”

O principal desafio é que os dispositivos de borda possuem limitações na capacidade de processamento e conectividade intermitente. Assim, para garantir a gestão de recursos em sistemas IoT críticos é preciso considerar três fatores importantes, que são descarga de dados, equilíbrio de carga e respostas em tempo real. Uma solução baseada em arquiteturas de névoa multicamadas capaz de prover respostas em tempo real das aplicações com adaptabilidade e eficiência na comunicação pode mitigar o uso dos recursos de ambientes IoT.

1.2 Problema de Pesquisa

Em sistemas de ambientes IoT, o monitoramento remoto de saúde é reconhecido como um dos setores mais críticos, uma vez que o sistema deve responder em tempo real as requisições. Além disso, este ambiente consiste de dispositivos móveis suscetíveis a conexão e desconexão frequentes, os quais são utilizados para coletar, processar, armazenar e/ou transmitir dados. Este cenário reflete a volatilidade e intermitência de ambientes IoT. Como discutido na seção anterior, é comum que as arquiteturas tradicionais baseadas em nuvem possuam ocorrência de desconexão da rede principal ou variações de largura de banda e latência, o que gera prejuízos aos serviços remotos, principalmente serviços de saúde. Neste contexto, os sistemas necessitam prover um grau maior de confiabilidade e adaptabilidade durante o sensoriamento pelos dispositivos de borda.

O principal objetivo desta pesquisa de doutorado é mitigar o uso de recursos em sistemas de tempo real em ambientes IoT. Diante disso, é proposto:

1. uma arquitetura auto adaptativa de névoa multicamadas baseada em F2C e *Indie Fog*; e
2. um *middleware* autônomo baseado em ciclos de *feedback* para a descarga dinâmica de dados.

Devido à complexidade e diversidade de problemas relacionados a sistemas de tempo real em ambientes IoT, esta tese possui a seguinte hipótese:

Hipótese: Uma arquitetura multicamadas de névoa baseada em ciclos de *feedback* para a descarga dinâmica de dados e comunicação assíncrona entre os dispositivos provê a mitigação de uso dos recursos computacionais em ambientes IoT.

Os sistemas de ambientes IoT são altamente sensíveis ao volume de dados e ao atraso para o tempo de resposta (AL-FUQAHA *et al.*, 2015). Diante disso, esta pesquisa adota o uso de ciclos de *feedback* para garantir a adaptabilidade na execução dos dados, tomando decisões de descarregamento de dados mais eficientes. A arquitetura de referência MAPE-K (IBM, 2005) e o paradigma *publish-subscribe* (EUGSTER *et al.*, 2003) são utilizados como base para a construção de um *middleware* autônomo para *offloading* dinâmico de dados em uma arquitetura baseada na névoa multicamadas. Além disso, foi proposto um modelo de comunicação capaz de desacoplar os sensores e dispositivos do ambiente IoT e evitar danos às aplicações causados pelas possíveis desconexões dos dispositivos.

Uma infraestrutura adequada e o provisionamento de recursos de forma eficiente e autônomo, por meio de uma arquitetura multicamadas de névoa, pode mitigar os efeitos indesejáveis no sensoriamento remoto de saúde e, conseqüentemente, melhorar o uso dos recursos e a confiabilidade de sistemas IoT através do ciclo de *feedback*. A hierarquia da arquitetura de névoa multicamadas é fundamental para lidar com o congestionamento e redundância de dados

ocasionados pelo grande número de produtores de informações em sistemas IoT. Além disso, a arquitetura de névoa pode facilitar a tomada de decisões e gestão dos dados, uma vez que pode ser realizada logo que os dados são coletados ou na camada de névoa seguinte, sem que haja a necessidade de enviar os dados de entrada diretamente para a nuvem.

Por sua vez, a adaptabilidade ajuda a mitigar os efeitos de sobrecarga no ambiente de execução e a melhorar as limitações arquiteturais dos dispositivos, aproveitando a interface de comunicação e mecanismos de descarga para transmissão dos dados para a próxima camada de névoa ou nuvem. No entanto, alguns dos seus inconvenientes são acentuados na comunicação entre a camada de névoa e a nuvem, por exemplo, o tempo do processamento de aplicativos IoT críticos ser limitado para o descarregamento dos dados para a nuvem. O desempenho de sistemas críticos pode diminuir severamente se esse cenário não for cuidadosamente considerado. Cada abordagem possui suas desvantagens, que quando combinadas, podem mitigar o uso de recursos em sistemas de tempo real em ambientes IoT.

Estudo de caso. Monitoramento da saúde física e emocional do usuário:

A arquitetura proposta nesta tese é avaliada por meio de um estudo de caso em ambientes inteligentes de sensoriamento de saúde, que está relacionado ao conceito de HSH. Com isso, uma HSH pode ser definida como um ambiente ou casa munidos com dispositivos inteligentes e/ou “coisas” especializadas para cuidados de saúde de maneira remota (i.e., acompanhar e monitorar o paciente a distância pelos profissionais de saúde e/ou familiares em qualquer lugar e a qualquer momento (TALEB; BOTTAZZI; NASSER, 2010)). Estas “coisas” podem ser sensores ou atuadores que irão agir/atuar quando uma situação crítica for detectada. Neste cenário, as informações podem ser agregadas de múltiplas fontes e distribuídas para consumidores que não têm conexões diretas com os produtores de informações (e.g., dados de monitoramento de um paciente em tratamento domiciliar compartilhados com um sistema de hospital (EUGSTER *et al.*, 2003)). Além disso, deve-se garantir a adaptabilidade na disponibilidade dos dispositivos do ambiente e melhorar a confiabilidade na comunicação, uma vez que ambientes IoT são ambientes em que os dispositivos possuem comunicação intermitente. Para isso, foi utilizado o paradigma *publish-subscribe* para fornecer um módulo de comunicação assíncrona entre os dispositivos da arquitetura de névoa proposta.

1.3 Estrutura da Tese

Este capítulo resumiu a motivação, os problemas abordados, os objetivos e as principais contribuições desta pesquisa de doutorado. Os capítulos restantes desta tese estão organizados da seguinte forma: O Capítulo 2 fornece uma visão geral dos conceitos necessários para o entendimento desta tese. O Capítulo 3 apresenta os trabalhos presentes na literatura relacionados com o tema de pesquisa desta tese. O Capítulo 4 apresenta uma arquitetura hierárquica auto adaptativa de névoa multicamadas para a realização da descarga dinâmica em ambientes IoT. No Capítulo

5, um estudo de caso no sensoriamento do estado físico e emocional do usuário é realizado em ambientes inteligentes de monitoramento. Finalmente, no Capítulo 6 são apresentados as considerações finais e contribuições desta tese, bem como as direções para trabalhos futuros.

FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos que serão utilizados neste projeto. Destacamos os principais conceitos relacionados a Internet das Coisas, computação em nuvem, computação em névoa, computação autônoma e descarregamento de dados.

2.1 Internet das coisas

Diariamente, bilhões de pessoas utilizam a Internet para acesso a redes sociais, compras, notícias, entretenimento, entre muitas outras. Cada vez mais, essas pessoas têm acesso a dispositivos conectados a uma infraestrutura global de rede que permite uma interação em larga escala entre pessoas e máquinas. A necessidade de conectividade é crescente e hoje se estende além de interações que envolvem seres humanos. Para isso, é preciso que dispositivos sejam capazes de se conectarem a essa plataforma mundial, alcancem e reconheçam parceiros e coordenem-se para que, juntos, consigam alcançar um objetivo. Neste contexto, a Internet das Coisas é palco de uma evolução para uma rede de objetos interconectados que não só colhem informações do ambiente (percepção) e interage com o mundo físico (atuação/comando/controle), mas também usa padrões da Internet para prover serviços para transferência de informação, análises, aplicações e comunicações (GUBBI *et al.*, 2013). Neste contexto, continuará sendo usada como núcleo, através do qual acontecerá toda a difusão de informações e dados entre dispositivos com capacidades computacionais que utilizam um conjunto de tecnologias diversas (MIORANDI *et al.*, 2012).

Apesar de ser hoje um conceito amplamente utilizado, a definição de IoT ainda é um pouco complicada por ser resumida a apenas duas palavras, “Internet” e “coisas”, as quais, separadas, representam vieses diferentes, mas que, juntas, têm poder maior que sua mera combinação. Atzori, Iera e Morabito (2010) discutem que a Internet das Coisas, para ser melhor entendida, pode ser dividida em três visões: uma centrada na Internet, outra, em *coisas* e ambas culminam na necessidade de uma terceira, centrada em *semântica*. Gubbi *et al.* (2013) adicionam

que a verdadeira utilidade de IoT é encontrada onde as três visões se intersectam.

A primeira visão é a centrada em coisas e ela foi o ponto de partida da IoT. As *coisas* eram, inicialmente, dispositivos de identificação por radiofrequência (RFID). Esses objetos oferecem uma opção simples e cada vez mais barata para a incorporação de rastreabilidade, conhecimento de estado e de localização. A segunda visão se refere à importância que a Internet possui enquanto infraestrutura de conexão de objetos virtuais a objetos físicos e o seu desenvolvimento para permitir a criação e crescimento de serviços e aplicações com alto grau de autonomia, conectividade de rede, interoperabilidade e volume de eventos. Por fim, a visão semântica sugere que, com um número extremamente elevado de dispositivos conectados na Internet do futuro, desafios grandes relacionados a representação, visão, interconexão, busca e organização de informação naturalmente aparecerão, mas poderão ser amparados por tecnologias semânticas. Elas permitirão a modelagem para a descrição de coisas, decisão com base em dados, a criação de ambientes semânticos e arquiteturas que permitam atingir requisitos IoT, melhorias de armazenamento e criação de infraestrutura de comunicação escaláveis com base no significado dos dados. A interoperabilidade, integração e adaptação desses sistemas sugerem o uso de uma arquitetura e infraestrutura capazes de prover todos esses recursos e o caminho para a realização dessa visão é uso de soluções em nuvem (ATZORI; IERA; MORABITO, 2010; MIORANDI *et al.*, 2012).

As características da infraestrutura IoT são a existência de dispositivos heterogêneos, com restrição de recursos, que realizam interações espontâneas com outros objetos. Essas interações ocorrem dentro de uma rede de escala enorme, gerando uma grande quantidade de eventos. Por conta de fatores de falha, tanto de *software*, de *hardware* ou de ambiente, a todo momento dispositivos conectam-se e desconectam-se da rede, fazendo-a dinâmica e, muitas vezes, não estruturada (*ad-hoc*). A noção de contexto e de localização, além da inteligência, são também características das coisas que formam a infraestrutura naturalmente distribuída IoT (GUBBI *et al.*, 2013; STOJKOSKA; TRIVODALIEV, 2017; MUTLAG *et al.*, 2019).

As áreas que se beneficiam da Internet das Coisas podem ser divididas em quatro domínios principais (GUBBI *et al.*, 2013):

1. **Pessoal e doméstico:** a informação coletada é trafegada dentro de redes pessoais e usadas apenas por indivíduos que são donos da rede. A proximidade de dispositivos de pessoas permite, por exemplo, o monitoramento de funções vitais a partir de sensores *Bluetooth*, relógios inteligentes ou *smartphones*, que enviam essas informações para servidores, possibilitando cuidados de saúde ubíquos, uma vez que médicos podem acessar essa informação de qualquer lugar. Além disso, os eletroeletrônicos de casa, como televisões, ar-condicionados, refrigeradores etc. equipados com interfaces de tecnologia sem fio podem ser controlados remotamente, permitindo um gerenciamento melhor dos equipamentos e da energia que consomem.

2. **Empresarial:** nesse domínio, os dados costumam ser gerados a partir de sensores localizados em um ambiente corporativo e são utilizados apenas pelas empresas. O monitoramento do local de trabalho é um dos casos de uso mais comum, para acompanhar o número de ocupantes de um lugar e realizar o gerenciamento de recursos de uma instalação. Um dos desafios históricos de empresas é a instalação de sensores em suas plantas. Antigamente, isso significava dezenas de metros de cabos de rede atravessando salas e pavimentos e dificultando a instalação e manutenção desses dispositivos. Com sensores IoT sem fio, esse ambiente ganha flexibilidade, sendo um dos maiores desafios apenas a configuração de uma sub-rede dedicada aos dispositivos.
3. **Serviços de utilidade pública:** os dados coletados nesse domínio são usados para a melhoria de serviços ao invés de consumo por usuários. Empresas de serviços públicos como energia elétrica e água já usam esses sistemas para gerenciamento próprio de recursos, otimizando o balanço entre custo e lucro. O alcance das redes para esse tipo de domínio varia em escala regional e nacional e as tecnologias utilizadas podem ir desde celular até satélites. Monitoramento das redes elétricas, captura, processamento e análise de vídeo para acompanhamento e vigilância também são usos recentes de IoT por empresas. Por fim, IoT tem sido usada por empresas para garantir a qualidade da água que chega aos consumidores por meio de sensores, monitoramento de irrigação e de solo na agricultura, provendo os *stakeholders* com dados importantes para a tomada de decisões.
4. **Serviços móveis:** IoT está sendo usada nos ramos de transporte e de logística, melhorando, assim, operações de cadeia de produção. A relação disso com serviços móveis é que o tipo de rede utilizada pelos dispositivos na cidade é do tipo *Wireless Sensor Network* (WSN), que pode ter alcance metropolitano, o que facilita o monitoramento em larga escala de tempos de viagem, condições de trânsito, o planejamento de rotas e até mesmo controle de barulho. A introdução da tecnologia *Bluetooth* em vários produtos digitais, como sistemas de som automotivos, fones de ouvido, telefones, GPS, também colabora com expansão do caráter móvel da Internet das Coisas.

2.2 Computação em nuvem

O conceito mais largamente aceito para computação em nuvem foi concebido pelo *National Institute of Standards and Technology* (NIST), que define:

Computação em nuvem é um modelo para permitir o acesso ubíquo, conveniente e sob demanda a um conjunto de recursos computacionais configuráveis (ex.: redes, servidores, armazenamento, aplicações e serviços) que podem ser provisionados e lançados rapidamente com o mínimo esforço de gerenciamento ou interação com provedor de serviço (MELL; GRANCE *et al.*, 2011, p. 6).

Além das promessas de acesso facilitado aos recursos, o modelo de computação em nuvem promete uma redução do investimento inicial para a criação de aplicações, alta disponibilidade, tolerância a falhas e escalabilidade virtualmente infinita. Esses fatores têm chamado a atenção tanto da academia quanto do mercado (CAVALCANTE *et al.*, 2016).

Segundo Mell, Grance *et al.* (2011), dentre as características essenciais da computação em nuvem estão o *provisionamento de recursos sob demanda* sem a necessidade de interação humana com provedores; *largo acesso à rede*, permitindo que clientes de plataformas heterogêneas consigam se conectar aos recursos; o *agrupamento de recursos computacionais* (como armazenamento, processamento, memória e rede) para o serviço de múltiplos clientes com recursos físicos e virtuais requisitados sob demanda e que possibilite ao cliente endereçar todos os recursos; a *elasticidade rápida*, que dita que os recursos podem ser provisionados e lançados automaticamente sob demanda em qualquer quantidade e a qualquer momento; e a capacidade de *medição de serviços*, para monitoramento, informação e transparência com o provedor e com o usuário.

O oferecimento da infraestrutura de computação em nuvem pode ser dividido em três modelos de serviço: *Software as a Service* (SaaS), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS) (MELL; GRANCE *et al.*, 2011).

- SaaS (*Software as a Service*): é o mais simples, no qual o serviço oferecido ao cliente é de uma aplicação que executa em uma infraestrutura em nuvem e que é disponibilizada por meio de interfaces leves como um navegador *web* ou interfaces de programação. Não é responsabilidade do cliente o gerenciamento de infraestrutura em nenhum nível, com a exceção do gerenciamento de configuração da aplicação.
- PaaS (*Platform as a Service*): fornece ao cliente o poder de implantar em uma infraestrutura em nuvem aplicações criadas ou adquiridas por ele com o auxílio de ferramentas, serviços, bibliotecas ou linguagens de programações suportadas pelo provedor. Assim como no modelo SaaS, o cliente não tem controle sobre a infraestrutura, porém nesse ele controla totalmente a sua aplicação. É permitido ao cliente alterar configurações do ambiente em que suas aplicações são executadas, se for necessário.
- IaaS (*Infrastructure as a Service*): é o modelo mais abrangente dos três e permite ao cliente provisionar recursos de processamento, armazenamento, redes e outros necessários para a execução de qualquer tipo de software, além de permitir a escolha do sistema operacional. Apesar disso, o cliente não gerencia a infraestrutura na nuvem, tendo no máximo o poder de controlar elementos como o Sistema Operacional, *firewalls*, discos e aplicações que ele próprio lançou.

Apesar da evolução independente das áreas de computação em nuvem e de Internet das Coisas, ambas são complementares. Os dispositivos IoT, por terem restrições de memória,

processamento, energia e de conectividade, são candidatos ruins para o processamento em larga escala necessário para muitas aplicações de alcance global oferecidas na Internet. Já o modelo de computação em nuvem oferece recursos virtualmente ilimitados que podem ser provisionados a qualquer momento, permitindo uma computação ubíqua, poderosa e escalável, acessível de qualquer lugar do planeta. Unir os dois conceitos significa aproximar a nuvem dos dispositivos fontes de dados, tornando-a mais pervasiva, enquanto que os dispositivos, quando aproximados da nuvem, têm a capacidade de serem ubíquos, reutilizáveis e mesmo oferecidos como serviço, além de ganharem um potencial de processamento muito grande que mesmo juntos não conseguiriam alcançar (BOTTA *et al.*, 2014; CAVALCANTE *et al.*, 2016).

2.3 Computação em névoa

O conceito de computação em névoa foi introduzido pela CISCO em 2012, que corresponde a uma extensão do paradigma da computação em nuvem do núcleo para a borda da rede. O conceito de computação em névoa é trazer os recursos de rede mais próximos dos nós que estão gerando os dados (i.e., dispositivos do usuário final) (AAZAM; ZEADALLY; HARRAS, 2018a; MOURADIAN *et al.*, 2018). Névoa é uma plataforma virtualizada que é responsável pelo provimento de recursos de computação e armazenamento fisicamente mais próximos do usuário, além de servir como uma arquitetura de comunicação capaz de conectar os dispositivos IoT e a nuvem (AAZAM; ZEADALLY; HARRAS, 2018a; MOURADIAN *et al.*, 2018). Névoa deve estar ligada a uma nuvem, no entanto pode possuir uma arquitetura de n-camadas para fornecer flexibilidade ao sistema. Esta característica tem tido uma atenção especial, que visa as interações entre as camadas de névoa ou entre a névoa e a nuvem tradicional (BONOMI *et al.*, 2012; HABAK *et al.*, 2017; AAZAM; ZEADALLY; HARRAS, 2018a; MOURADIAN *et al.*, 2018).

A computação em névoa funciona fora da nuvem como uma infraestrutura capaz de garantir os serviços de nuvem (por exemplo, processamento, armazenamento, fluxo de trabalho, aplicações e análise de dados) nas bordas da rede de forma satisfatória e verdadeiramente distribuída (HABAK *et al.*, 2017; AAZAM; ZEADALLY; HARRAS, 2018a). No entanto, a névoa não substitui a nuvem, pois trata-se de apenas uma extensão que desenvolve os conceitos de serviços em nuvem. Os serviços são alojados nas bordas da rede ou mesmo em dispositivos finais, *set-top-boxes* ou *access points*, etc (MOHAN; KANGASHARJU, 2016; AAZAM; ZEADALLY; HARRAS, 2018a). A computação em névoa conceitual baseia-se em tecnologias existentes e comuns, como redes de distribuição de conteúdo, mas com base em tecnologias de nuvem, capazes de garantir a entrega de serviços mais complexos. No entanto, o desenvolvimento de aplicações usando recursos de computação em névoa é mais complexo (MOHAN; KANGASHARJU, 2016; MOURADIAN *et al.*, 2018).

Como pode ser visto na Figura 2, a computação em névoa tradicional é dividida em

três camadas: (1) Dispositivos IoT, (2) Nó de névoa, e (3) Nuvem. Os dispositivos IoT são responsáveis por coletar os dados dos sensores e enviar para o nó de névoa. Tais dispositivos possuem limitação no poder de processamento. O nó de névoa pode ser formado por um dispositivo ou um conjunto de dispositivos conectados (i.e., roteadores, *switches*, *gateways*, *access point*, PCs, smartphones, *set-top boxes*, etc.) capazes de controlar o fluxo de dados providos dos dispositivos de borda e fornecer os serviços de forma adequada. Vale ressaltar, que a computação de névoa pode fornecer consciência de contexto e um melhor apoio à mobilidade dos usuários. Finalmente, a nuvem é formada por centros de dados com alto poder de processamento, podendo fornecer serviços ilimitados.

Figura 2 – Arquiteturas de computação em névoa (Adaptado de (SOUZA *et al.*, 2018b))

A camada de névoa entre os dispositivos IoT e a nuvem torna os serviços mais conscientes e contextualizados com o ambiente de aplicação, devido à sua proximidade física da camada física onde ficam os dispositivos. Devido à sua baixa latência, serviços de *streaming* podem ser fornecidos com mais qualidade (CHIANG *et al.*, 2017), aplicações no transporte inteligentes (HONG *et al.*, 2013), serviços relacionados aos cuidados de saúde, entre outros. Em serviços de saúde, a névoa pode processar os dados de acordo com os requisitos de cada serviço. Assim, é possível fazer a distribuição de processamento entre a névoa e a nuvem. Por exemplo, dados brutos pré-processados na névoa e transmitidos para a nuvem, enquanto que a nuvem transforma as informações pré-processadas em serviços melhorados (AAZAM; ZEADALLY; HARRAS, 2018a). Por exemplo, com base no histórico dos dados do usuário, um serviço na nuvem pode identificar que o nível de açúcar no sangue de um paciente aumentou, ou que os batimentos cardíacos estão oscilando mais do que o normal (AAZAM; ZEADALLY; HARRAS, 2018a).

2.4 Computação Autônoma

A computação autônoma é um conceito que reúne diversas áreas da computação com o objetivo de criar sistemas computacionais de autogestão, reduzindo assim, o envolvimento humano com esses sistemas (HUEBSCHER; MCCANN, 2008).

Segundo Huebscher e McCann (2008) o termo “autônomo” vem da biologia, visto que no corpo humano, é o sistema nervoso autônomo que cuida dos reflexos inconscientes, i.e, das funções corporais que não requerem a nossa atenção como os ajustamentos corporais, tais como, o tamanho da pupila, as funções digestivas do estômago e dos intestinos, a taxa e a profundidade da respiração, e a dilatação ou constrição dos vasos sanguíneos. Sem o sistema nervoso autônomo, estaríamos constantemente ocupados, conscientemente, adaptando nosso corpo para as suas necessidades e ao meio ambiente.

2.4.1 Propriedades de autogestão

A IBM sugere em seu manifesto (HORN, 2001), que sistemas computacionais complexos também devem ter propriedades autonômicas, ou seja, devem ser capazes de tratar de modo independente as tarefas regulares de manutenção e otimização, reduzindo assim a carga sobre os administradores de sistemas. Assim, são definidas quatro propriedades de autogestão, a saber:

Autoconfiguração: Um sistema computacional autônomo configura-se conforme suas metas de alto nível, ou seja, especificando o que é desejado, mas não necessariamente como deve ser realizado. Isto pode significar ser capaz de instalar e configurar-se com base nas necessidades da plataforma e do usuário (KEPHART; CHESS, 2003).

Auto otimização: Um sistema de computação autônomo aprimora o uso de recursos. Com objetivo de melhorar o desempenho ou a qualidade do serviço, proativamente, ele pode decidir iniciar uma mudança (KEPHART; CHESS, 2003).

Autocura: Um sistema de computação autônomo detecta e realiza o diagnóstico dos problemas, que podem ser de baixo nível, como uma falha de hardware, ou alto nível, como uma falha de software. Embora o sistema deva tentar resolver o problema, como por exemplo, instalando automaticamente atualizações no sistema, é importante que o sistema não seja mais prejudicado com novos erros nesse processo de cura. Por esta razão, um aspecto importante neste processo, é a tolerância a falhas, ou seja, um sistema autônomo reativo a falhas ou aos primeiros sinais de uma eventual falha. (KEPHART; CHESS, 2003).

Autoproteção: O sistema automaticamente precisa se ajustar com objetivo de alcançar segurança, privacidade e proteção de dados, visto que a segurança é um aspecto crucial da autoproteção, tanto em software como no hardware. Ele também pode ser capaz de prever e impedir possíveis violações de segurança, uma vez que, desse modo, o sistema nervoso autônomo apresenta propriedades proativas (KEPHART; CHESS, 2003).

As propriedades citadas acima são inspiradas nas características de software agentes, que são (WOOLDRIDGE; JENNINGS, 1995; HUEBSCHER; MCCANN, 2008):

Autonomia: Agentes atuam sem a mediação de humanos ou outros, e possuem controle sobre suas atividades e estado interno.

Habilidade social: Agentes exercem interação uns com os outros, até mesmo com humanos, por meio de alguma linguagem de comunicação.

Reatividade: Agentes constataam seu ambiente e rapidamente respondem às transformações que ocorrem nele.

Proatividade: Agentes tomam iniciativa, visto que são capazes de apresentar um comportamento dirigido a objetivos.

Sistemas auto adaptativos contém alguns elementos dessas propriedades, especialmente para proporcionar a otimização. Um exemplo disso pode ser visto em sistemas de *streaming* de mídia, em que o objetivo é manter a reprodução de música ou vídeo em uma qualidade tão alta quanto possível, mas adaptando à largura da banda.

2.4.2 Graus de automaticidade

Em IBM (2002) é proposto um conjunto de níveis de computação autônoma, que se estende do nível 1: básico, para o nível 5: Autônomo.

1. Este primeiro nível define o estado em que elementos do sistema são geridos por pessoas qualificadas, que fazem uso de ferramentas de monitoramento, e posteriormente implementam manualmente as alterações. Acredita-se a maioria dos sistemas de TI estão neste nível atualmente.
2. Conhecido como *Managed* ou Gerenciado, neste nível a coleta de informações é feita de forma inteligente, por meio de ferramentas de monitoramento, diminuindo a carga da administração de sistemas.
3. Já no nível 3 é realizado o reconhecimento de padrões de comportamento do sistema e partir disso, é sugerido ações a serem aprovadas e realizadas pela equipe de TI.
4. Neste nível, conhecido como nível de adaptação, o sistema faz uso das ferramentas disponíveis no nível 3, mas é mais capaz de agir. É reduzida a interação humana, e espera-se que o sistema realize o serviço com bom desempenho.
5. Já no nível autônomo completo, os componentes são geridos de forma dinâmica por regras e políticas de negócios, permitindo que a equipe fique focada nas necessidades de negócios.

2.4.3 Ciclos de feedback

No caso de sistemas de software intensivos, que são aqueles que interagem com outros softwares, dispositivos, sensores e com as pessoas (WIRSING, 2006), a auto adaptação surge de diversas formas. As decisões de projeto em sistemas auto adaptativos precisam ser tomadas em tempo real para controlar o comportamento dinâmico, o estado e as razões individuais do sistema (BRUN *et al.*, 2009).

Segundo Brun *et al.* (2009) ciclos de *feedback* (i.e., loops de feedback) fornecem um mecanismo genérico para a auto adaptação. Normalmente, um ciclo de *feedback* é formado por quatro passos principais: coletar, analisar, decidir e agir. Ao longo do ciclo, os sensores coletam dados durante a execução do sistema e seu contexto. Esses dados acumulados são filtrados e armazenados para referência futura com o objetivo de retratar um modelo preciso de estados passados e atuais. Em seguida, os dados são analisados para inferir tendências e identificar sintomas. Posteriormente, tenta-se prever o futuro, para decidir como agir sobre a execução do sistema e seu contexto, por meio de atuadores e efetores.

Figura 3 – Autonomic Control Loop (BRUN *et al.*, 2009).

O ciclo de controle autônomo, como representado na Figura 3, é um modelo genérico de retroalimentação fechado e centrado nas atividades que realizam *feedback*. Este modelo é um refinamento da abordagem *sense-plan-act* para controlar robôs autônomos, desenvolvido na década de 80 (NILSSON, 1980; BRUN *et al.*, 2009). Enquanto a figura mostra um ciclo de controle único em um sistema prático, são envolvidos múltiplos ciclos. Neste modelo, o fluxo de controle é unidirecional, e apesar do fluxo de dados não ser detalhado, o modelo é um bom ponto de partida para a discussão de loops de controle.

O ciclo de *feedback* inicia-se com a coleta de dados relevantes de sensores ou outras fontes, que refletem o estado atual do sistema. A partir daí, deve-se assegurar que os dados dos sensores sejam confiáveis e se eles fornecem informações suficientes para identificação do sistema (BRUN *et al.*, 2009). Assim, os dados obtidos podem ser analisados pelo sistema. Para isso, existem várias abordagens para estruturação e raciocínio dos dados brutos, como, por

exemplo, por meio de modelos, teorias e regras. Contudo, segundo Brun *et al.* (2009), dentre os pontos que precisam ser refletidos após a coleta dos dados são: Quão atual é o estado que o sistema infere? Quanto que o estado passado pode ser necessário posteriormente? Quais dados precisam ser arquivados para validação e verificação? Quão fiel vai ser o modelo para o mundo real a partir dos dados dos sensores disponíveis? Quão estável será o modelo ao longo do tempo?

Segundo Brun *et al.* (2009), uma decisão sobre como adaptar o sistema precisa ser tomada, para que chegue a um estado desejável. Para isso, algumas abordagens, como a análise de risco, podem ajudar na escolha entre várias alternativas. Neste ponto, as questões determinantes são: Como será o estado do sistema no futuro? Como é tomada uma decisão? Quais são as prioridades para a auto adaptação, vários *loops de feedback* dentro de um único ciclo de *feedback*?

Por último, o sistema deve agir por meio de efetores ou atuadores para implementar a decisão. Com isso, alguns pontos determinantes precisam de atenção, como por exemplo, se a adaptação pode ser feita com segurança, se ajustes de diferentes ciclos interferem uns nos outros, e qual dos *feedbacks* (centralizado ou descentralizado) melhor contribui para o alcance da meta global, e adicionalmente se os efetores são suficientes para guiar o sistema nas direções desejadas (BRUN *et al.*, 2009).

No entanto, o comportamento do *feedback* em um sistema auto adaptativo, que se realiza com ciclos de controle, é uma característica essencial e deve ser elevado a uma entidade de primeira classe em sua modelagem, projeto, implementação, validação e operação. Em vista disso, além de deixar os ciclos de controle explícitos, as suas propriedades também devem ser explicitadas. Garlan *et al.* (2004) defendem que a auto adaptação seja feita externamente para separar as preocupações da adaptação das funcionalidades do sistema.

Com ênfase em sistemas de autogestão, uma iniciativa da IBM *Autonomic Computing* trouxe um grande avanço em ciclos de *feedback* explícitos (KEPHART; CHESS, 2003). Uma das principais constatações desta pesquisa é o uso do *MAPE-K loops de feedback*, que compreende as fases monitorar, analisar, planejar e executar, considerando uma base de conhecimento, para a construção de sistemas autônomos (IBM, 2005), como mostrado na Figura 4. As fases do MAPE-K atendem facilmente o ciclo de controle genérico, apresentado na Figura 3. Como visto, ambos os diagramas destacam as principais atividades do ciclo de *feedback* enquanto abstraem características do fluxo de controle e de dados ao redor do ciclo. No entanto, o *MAPE-K* fornece instruções abrangentes sobre como arquitetar e implementar as quatro fases, as bases de conhecimento, sensores e atuadores. Além disso, ele também descreve como compor elementos autônômicos para orquestrar a autogestão (BRUN *et al.*, 2009).

O elemento autônômico *MAPE-K*, apresentado por Kephart e Chess (2003) e popularizado por causa da iniciativa da IBM (2005), é a primeira arquitetura a expor explicitamente o ciclo de controle de *feedback* (Ver Figura 5), assim como os passos apresentados na Figura 3, os quais identificam componentes funcionais e interfaces para decompor e gerenciar o ciclo de *feedback*. Neste caso, a IBM usa o elemento autônômico como base fundamental para a

Figura 4 – MAPE-K (IBM, 2005).

realização da autoconfiguração, autocura, auto autoproteção e sistemas auto otimizáveis.

Figura 5 – Ciclo de Controle de *Feedback* (BRUN *et al.*, 2009)

A Figura 5 mostra o ciclo de controle de *feedback* clássico. O objetivo do sistema é manter as propriedades específicas do processo Y_p ou suficientemente próximas das entradas de referência (*set points*) acima. A saída do processo Y_p pode variar naturalmente; além disso, perturbações externas D podem ocorrer no processo. A saída do processo Y_p é realimentada por meio de sensores e Y_b calcula a diferença considerando as entradas de referência acima. O controlador implementa um algoritmo de controle especial ou estratégia, que leva em conta a diferença entre U_p e Y_b , para decidir sobre uma correção adequada U que leva a Y_p mais próximo de U_p , utilizando atuadores específicos do processador (BRUN *et al.*, 2009).

Conforme apresentado na Figura 4, um sistema autônomo é constituído por um elemento gerenciado e um gestor autônomo, arquitetado com um ciclo de controle de *feedback*. Assim sendo, no ciclo de *feedback* genérico, o gestor autônomo corresponde ao controlador, e o elemento gerenciado, assemelha-se ao processo (BRUN *et al.*, 2009).

Além do motor *MAPE-K*, o gestor possui duas interfaces de gerenciamento, o sensor e o efector. O monitor detecta o processo de gestão e de seu contexto, filtra os dados do sensor acumulados, e armazena eventos relevantes na base de conhecimento para uma futura referência. Em seguida, para diagnosticar sintomas, o analisador passa a comparar os dados de eventos

com os padrões da base de conhecimento. Com isso, o planejador entra em ação interpretando esses sintomas e elaborando um plano de execução, que é realizado em seguida. As interfaces de gerenciamento, que consistem de sensores e efetores, são padronizadas em todos elementos gerenciados e blocos de construção autônomos, para facilitar a colaboração, a integração de dados e o controle entre os elementos autônômicos. A partir do elemento de gestão e das informações de estados atuais e passados o gestor autônomo reúne medições e ajusta o elemento gerenciado, por meio de sua interface de gerenciamento e de acordo com o seu objetivo (BRUN *et al.*, 2009).

2.5 Descargamento de Dados

Com a rápida evolução da internet e tecnologias como nuvem (do inglês, *Cloud computing*), foram surgindo novas formas de prover serviços, como IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*) e SaaS (*Software as a Service*). Aliado a tais serviços, computação em névoa (do inglês, *Fog Computing* (BONOMI *et al.*, 2012)) e descarga computacional (do inglês, *Computation Offloading* (KUMAR *et al.*, 2013)) se destacam na distribuição de recursos computacionais (KUMAR *et al.*, 2013). A descarga computacional difere de outras técnicas que envolvem o uso de recursos computacionais, pois a técnica busca manter em sua essência, a ideia de computação distribuída e computação paralela (KUMAR *et al.*, 2013). Em outras palavras, a técnica de descarga computacional define que tanto o dispositivo local quanto a máquina remota tem, instalados em si, os *softwares* ou dados para que aplicações possam responder efetivamente em ambos os ambientes computacionais (KUMAR *et al.*, 2013; Mach; Becvar, 2017).

A decisão de realizar a descarga em outro ambiente computacional almeja estender os recursos limitados das máquinas locais (por exemplo, *smartphones*) através dos recursos disponibilizados pelas máquinas remotas (por exemplo, *notebooks* configurados como servidores). Essa decisão de descarga permite o processamento dos dados pelas máquinas remotas e respostas em tempo real de tais máquinas para os dispositivos escassos (KUMAR *et al.*, 2013; Mach; Becvar, 2017)

As principais variações para descrever as categorias de descarga computacional, segundo Kumar *et al.* (2013), podem ser representadas pelas questões: (1) o que descarregar? e (2) quando descarregar?

1. O que descarregar:

- Descarga de tarefas: essa metodologia propõe que o dispositivo local deve enviar para o dispositivo remoto um par de informações, o que processar e como processar.
- Descarga de métodos: similar ao item anterior, este cenário descreve que o dispositivo local envia ao dispositivo remoto um par de informações, o que processar e como processar, no entanto, neste caso é esperado que a definição de como processar seja feita de forma mais específica, com uma função.

- Descarga de dados: é presumido que o dispositivo remoto esteja ciente de como processar a requisição e então apenas uma informação singular é compartilhada, o que processar.

2. Quando descarregar:

- Descarga estática: nesta modalidade, a decisão de quando realizar a descarga é definida em tempo de compilação, ou seja, definida pelo programador durante a criação da aplicação. Por exemplo, o programador define um limiar crítico de 50MB para memória do dispositivo e sempre que a memória livre do dispositivo estiver em menos que o limiar, a descarga deve ser feita.
- Descarga dinâmica: nesta abordagem, o programador não define previamente quando ocorrerá o descarregamento. A aplicação deve conter um decisor de descarga capaz de avaliar o seu ambiente, em relação aos recursos disponíveis e decidir em tempo real, enquanto a aplicação está em execução. Se naquele momento a descarga é ou não necessária. Por exemplo, após monitorar a conexão com o servidor durante algum tempo, nota-se que a conexão fica lenta por longos períodos. Neste caso, o decisor de descarga pode optar pelo processamento local.

A decisão de descarga estática possui todas as etapas de descarga previstas no código durante a arquitetura da aplicação (Mach; Becvar, 2017). A aplicação é dependente de conexão à Internet para realizar estas tarefas e normalmente estas são tarefas exclusivas de um servidor remoto, como acesso a um banco de dados ou *backup* remoto de informações. Estas técnicas são amplamente usadas e existem diversos protocolos de troca de informação para este fim, como por exemplo *REST* (FIELDING; TAYLOR, 2000) ou *GraphQL* (GRAPHQL, 2019).

A decisão de descarga dinâmica possui um mecanismo de decisão em tempo de execução, que permite a execução de uma tarefa localmente ou remotamente (Mach; Becvar, 2017; XU *et al.*, 2018). Tais mecanismos são baseados na arquitetura de comunicação cliente-servidor (Mach; Becvar, 2017; XU *et al.*, 2018), porém podem se distinguir pela sincronia do ambiente móvel com uma máquina virtual (SHI *et al.*, 2014; QIAN; ANDRESEN, 2015; XIA *et al.*, 2014) ou pela descarga de código fonte, invocando-o através de uma Chamada de Procedimento Remota - *Remote Procedure Call* (RPC) por meio de anotações, compilação especial ou modificação de binário (CUERVO *et al.*, 2010; NETO *et al.*, 2018a). Importante destacar que os mesmos servidores remotos que provém serviços para aplicações desenvolvidas utilizando técnicas de descarga estática podem servir aplicações desenvolvidas para suportar descarga dinâmica (Mach; Becvar, 2017). Nesta pesquisa, será considerada a descarga dinâmica com o descarregamento de dados.

A grande maioria dos estudos presentes na literatura sobre descarga dinâmica usam, principalmente, duas métricas para avaliar a necessidade de executar uma tarefa remotamente:

tempo de execução e consumo de energia (SHI *et al.*, 2014; XIA *et al.*, 2014; QIAN; ANDRESEN, 2015; ZHU *et al.*, 2017; LI *et al.*, 2018; HUANG *et al.*, 2019).

Geralmente, a decisão de descarga dinâmica é dividida em três etapas (Mach; Becvar, 2017; XU *et al.*, 2018): (1) particionamento da aplicação; (2) pré-processamento; e (3) decisão de descarga. Na primeira etapa, a aplicação é dividida em componentes que podem e componentes que não podem ser executados remotamente. O especialista da aplicação pode decidir durante a arquitetura do sistema ou por análise de código fonte. Na etapa de pré-processamento, a aplicação realiza todas as ações necessárias para poder executar os componentes selecionados remotamente. Isto inclui a seleção de um servidor remoto, a transferência e instalação de código fonte. Além da transferência de código fonte, pode-se transferir metadados para preparar o ambiente para a execução remota. Finalmente, a decisão de descarga é executada. A decisão pode ser tomada em tempo de compilação ou em tempo de execução. Na decisão em tempo de execução, informações mais precisas estão disponíveis. Por exemplo, o dispositivo móvel pode não possuir conexão à Internet ou energia suficiente para a transferência durante o tempo de execução. Portanto, na decisão em tempo de execução, o algoritmo toma a decisão automaticamente assim que uma situação muda, porém esta modelagem gera uma complexidade adicional.

TRABALHOS RELACIONADOS

Neste capítulo, é apresentada uma investigação do estado da arte em ambientes IoT, seguido de ambientes de névoa e descarga de dados. Serão discutidas as limitações dos principais trabalhos relacionados a esta Tese, bem como as questões em aberto. As buscas dos trabalhos relacionados seguiu uma metodologia de pesquisa, a qual utilizou a seguinte *String* de busca: “fog AND adapt* AND offloading AND (architecture OR middleware)”. Nesta busca, foram utilizados as seguintes bases de dados de indexação proeminentes: *ACM Digital Library* ¹, *IEEE Digital Library* ², *Web of Science* ³, *Science Direct* ⁴ e *Scopus* ⁵. Os trabalhos encontrados foram divididos nos dois grupos apresentados nas seções seguintes.

3.1 Ambientes inteligentes de IoT

Há uma necessidade crítica em relação às arquiteturas IoT, uma vez que esse tipo de arquitetura deve ser capaz de interconectar bilhões ou trilhões de objetos heterogêneos por meio da Internet. A literatura mostra que a padronização de arquiteturas IoT é essencial para criar um ambiente ideal de competitividade entre as empresas para o provimento de serviços de qualidade (AL-FUQAHA *et al.*, 2015).

Fortino *et al.* (2014) propõem uma arquitetura para sistemas IoT baseados em agentes e nuvem para permitir a cooperação de objetos inteligentes. A arquitetura proposta visa a integração de componentes baseados em agentes com a nuvem, uma vez que tais agentes podem suportar o desenvolvimento distribuído. Uma arquitetura web é proposta por Khan *et al.* (2016) que permite que variados dispositivos heterogêneos possam compartilhar informações. A arquitetura foi planejada para o ambiente de IoT com o intuito de garantir um processamento

---

¹ <http://portal.acm.org>

² <http://ieeexplore.ieee.org>

³ <http://www.isiknowledge.com>

⁴ <http://www.sciencedirect.com>

⁵ <http://www.scopus.com>

maior de informações com grandes números de dispositivos comunicando-se uns com os outros. Neste contexto, [Vinh et al. \(2015\)](#) sugerem a integração de IoT com *Mobile Cloud Computing* (MCC), técnica que discute a implantação de uma rede de dispositivos móveis capazes de se comunicar entre si e com servidores. Um paralelo com cenários de *smart homes* é estabelecido para exemplificar a relevância do método em um ambiente real. O trabalho discute alguns pontos considerados cruciais para estabelecer uma comunicação estável e segura, como o registro de dispositivos que entram na rede, a execução parcial de tarefas em aparelhos de recursos escassos e protocolos de segurança para a rede. A arquitetura final do modelo desenvolvido coloca um *smartphone* com Android como um *gateway* entre os sensores presentes em uma placa Arduino e os servidores em nuvem.

[Prado, Ortiz e Puig \(2017\)](#) apresenta uma arquitetura que permite melhorar a tomada de decisão inteligente nos escopos de IoT. A arquitetura proposta é baseada em computação de névoa, o que evita o consumo adicional dos recursos dos dispositivos nas bordas da rede e economiza custos da infraestrutura. A arquitetura utiliza *brokers* como serviços e diversos contextos para auxiliar na coleta e disseminação de dados e tomada de decisão inteligente. Os autores não comentam sobre o uso de técnicas para enfrentar falhas de *brokers* ou sensores, uma vez que podem gerar constrangimentos com a perda de conexão.

[Augusto et al. \(2018\)](#) apresentam uma nova metodologia centrada no usuário para desenvolver ambientes inteligentes. Os autores se concentram na integração de tecnologias e criam uma infraestrutura de software para fornecer serviços. O desafio enfrentado pelos autores está relacionado à confiança nos sistemas no mundo real, como qualidade de desempenho, correção comportamental e sua validação. Além disso, os autores se concentram nas partes interessadas envolvidas na construção de sistemas em ambientes inteligentes. [Patel, Ali e Sheth \(2017\)](#) descrevem uma abordagem inteligente para a análise de aplicativos de IoT focada na automação das transições entre as bordas da rede e a nuvem em situações dinâmicas. Os autores descrevem uma arquitetura dividida em três camadas, composta por diferentes componentes e atores que assumem papéis em soluções adaptáveis de análise de dados de IoT. No entanto, nenhuma avaliação é feita pelos autores. Além disso, nenhum dos autores acima apresentam uma avaliação para atestar como suas soluções afetam a infraestrutura de ambientes inteligentes.

Aplicações interativas em dispositivos móveis podem ser utilizadas para o sensoriamento inteligente e monitoramento individualizado de pessoas utilizando câmeras, sensores de dispositivos móveis ou *gadgets* específicos ([BENIGNO et al., 2014](#)). Neste contexto, HSH tem sido extensivamente estudado na literatura com os avanços do uso da IoT na assistência de saúde domiciliar. Por exemplo, aplicações podem emitir alertas para equipes de profissionais de saúde ou membros da família sempre que detectarem alguma anormalidade ([GONÇALVES et al., 2013](#); [MANO et al., 2016](#)). Além disso, esses sistemas podem monitorar idosos em suas casas ([LE et al., 2008](#); [GONÇALVES et al., 2013](#)), reconhecer a atividade humana ([BANERJEE et al., 2015](#); [LIOUANE et al., 2016](#)), ou interagir com pessoas ([MANO et al., 2016](#)).

Da mesma forma, [Maia et al. \(2015\)](#) desenvolveram uma plataforma de *middleware* para IoT baseada na Web para conectar médicos e pacientes utilizando sensores corporais acoplados e atuadores, denominada *EcoHealth* – Ecossistema Web de Dispositivos de *Health Care*. A solução é dividida em módulos de conexão de dispositivos, manipulação de dados, atuação, visualização, gerenciamento, armazenamento e serviços comuns. O foco principal dos autores é no monitoramento de pessoas em observação, uma vez que sugere sensores de respiração acoplados em uma máscara de oxigênio, um sensor de temperatura no braço do paciente e um sensor de eletrocardiograma. Dessa forma, o médico pode avaliar o paciente em tempo real remotamente. No entanto, considerando o cenário de uma HSH, o monitoramento apresentado pelos autores é considerado intrusivo e desconsidera a qualidade de vida e independência do usuário em seu cotidiano. Por outro lado, [Mano et al. \(2016\)](#) propõem uma arquitetura que explora aplicações de saúde utilizando tecnologias IoT. A arquitetura proposta, denominada SAHHc (*Smart Architecture for In-Home Healthcare*), é baseada no uso de IoT para o monitoramento individualizado e inteligente de pacientes idosos, propondo um sistema que emprega técnicas de reconhecimento de face e de expressões faciais para determinar o estado emocional dos pacientes e proporcionar-lhes o tratamento adequado em casas inteligentes. A solução proposta é composta por sensores não intrusivos, como câmeras e atuadores.

[Taleb, Bottazzi e Nasser \(2010\)](#) propõem um *framework*, denominado PEACH (*Pervasive Environment for AffeCtive Healthcare*), projetado para detectar alterações nos estados fisiológicos, condições psicofísicas dos pacientes e compartilhar essa informação com os cuidadores interessados, (e.g. profissionais de saúde, familiares e amigos). O PEACH é capaz de integrar um conjunto de sensores agregando informações sensoriais e detectando situações potencialmente perigosas para os pacientes com dependência química. Os autores utilizam diferentes biossensores vestíveis (e.g. eletrocardiograma, sensor para medir a condutividade da pele, o volume de sangue e a respiração) integrados com sapatos, brincos ou relógios e camisetas, detectando 80% de acurácia dos problemas de saúde mental (e.g. depressão comum). No entanto, os autores consideram a análise apenas de um único domínio, o que pode causar a imprecisão e dificuldade nas análises dos resultados. Diante disso, a solução proposta nesta tese visa o monitoramento a partir de múltiplas fontes de dados para integrar os dispositivos e garantir a comunicação assíncrona de qualquer tipo de dados do ambiente.

[Thangaraj et al. \(2016\)](#) propõem uma abordagem de gerenciamento inteligente e autônomo de pacientes em IoT com as seguintes vias tecnológicas: seleção de dispositivos, interoperabilidade, mapeamento e transformação de dados, validação e limpeza de dados, mapeamento de ontologia e agrupamento de nível de conceito por meio de agente analítico e semântico. Os autores utilizam agregação de dados para melhorar a eficiência energética, uma vez que há um grande volume de dados sendo transmitidos entre os diversos dispositivos no ambiente. Além disso, objetos inteligentes são integrados com enxames de agentes móveis para facilitar a cooperação e inteligência global. [Mane e Surve \(2016\)](#) propõem um sistema de monitoramento da atividade humana baseado em um *middleware* para provisionamento de contexto. A sensibilidade

ao contexto é fornecida baseada nas atividades humanas, por meio da tomada de decisão de classificadores em cada aplicação.

Gomes *et al.* (2015) apresentam uma infraestrutura de software com o suporte para o monitoramento de saúde de idosos. A plataforma é dividida em componentes capazes de se descobrirem, conectarem e comunicarem-se. Destaca-se a capacidade de estabelecer conexões rápidas e oportunistas com objetos inteligentes com base em diferentes tecnologias de rede sem fio de curto alcance. A infraestrutura se mostra escalável nos experimentos com vários dados coletados. Sakr e Elgammal (2016) apresentam um sistema baseado em Big Data para o gerenciamento no cuidado de saúde de pacientes por meio de serviços. O *framework* proposto apresenta uma arquitetura de camadas e serviços distribuídos no ambiente e na nuvem. Uma arquitetura de *middlewares* baseada em serviços foi apresentada por Ahlsén *et al.* (2012), que propuseram um sistema de informações de saúde. O sistema é dividido em camadas para realizar um gerenciamento aprimorado, além de facilitar o desenvolvimento de aplicações de saúde, garante a interoperabilidade dos dispositivos, serviços e aplicativos.

Aiello *et al.* (2012) propõem um sistema orientado a agentes móveis para programação de RSSF, denominado TinyMAPS. A arquitetura proposta é composta por agentes móveis interagindo entre si divididos por categoria de componentes. Este estudo de caso para o monitoramento de atividades humanas foi realizado com o objetivo de identificar a postura e os movimentos das pessoas. Sensores corporais foram utilizados para capturar os movimentos e o algoritmo de classificação *k-Nearest Neighbor* (KNN) alcançou uma acurácia de 78% caminhando, 97% sentado, 97% parado e 99% deitado. No entanto, esta solução é suscetível à perda de mensagens, especialmente em ambientes com recursos limitados e com mobilidade frequente, o que reflete o ambiente da IoT.

Vale destacar que, uma infraestrutura de névoa em ambientes inteligentes de IoT oferece possibilidades encorajadoras na provisão de recursos de ponta para uma HSH, uma vez que permite o provimento de recursos e serviços inteligentes em ambientes distintos para um ou mais usuários (ELKHATIB *et al.*, 2017). Por exemplo, dispositivos móveis conectados podem monitorar a saúde de um usuário em uma casa inteligente, atuando como múltiplos nós de névoa. Um nó de névoa implantado na borda da rede é capaz de lidar com a consciência do contexto do ambiente (SAQIB; HAMID, 2016).

3.2 Ambientes de névoa e descarga de dados

À medida que a IoT evolui, a computação em nuvem continua evoluindo a fim de provê melhores implementações de IoT no mundo real. No entanto, a nuvem fornece um alto poder de processamento de dados e ainda possui um alto atraso na comunicação com os dispositivos, desconexão da rede central ou variações de largura de banda (SOUZA *et al.*, 2018b). Por outro lado, a névoa atende a serviços específicos de baixa latência, uma vez que é implantada mais

próximo do usuário final e pode fornecer respostas em tempo real mais facilmente (SOUZA *et al.*, 2018b; MUTLAG *et al.*, 2019). Apesar dos benefícios da computação em névoa, deve-se levar em consideração que este tipo de infraestrutura possui recursos limitados.

Németh *et al.* (2018) propuseram um método centralizado para inclusão de serviços que combina um programa linear inteiro com um algoritmo de mapeamento guloso. O mecanismo de orquestração de recursos proposto usa recursos computacionais e de rede em ambientes de computação de névoa emergentes. Os autores avaliam o impacto da descarga por meio de migrações de *Virtual Network Functions* (VNF). Os autores focam no método de orquestração de recursos na infraestrutura, considerando dispositivos de borda e a nuvem.

O processamento distribuído em uma infraestrutura baseada nas limitações arquiteturais dos dispositivos do ambiente tem sido um dos principais desafios em IoT. Daneels *et al.* (2017) descreveram uma plataforma IoT para garantir a disseminação e análise de dados confiáveis durante a comunicação em áreas rurais e remotas, onde a infraestrutura sem fio é escassa. Assim, os autores lidam com o problema de conectividade usando múltiplos saltos de longo alcance entre sensores e atuadores. A solução proposta pelos autores transmite dados tanto na borda quanto na nuvem. Vale ressaltar que o equilíbrio da computação é essencial, uma vez que um alto volume de dados na borda pode sobrecarregar os dispositivos, o que torna o sistema menos escalável. Kang, Kim e Choo (2017) propõem um *gateway* auto configurável capaz de executar a detecção e configuração de dispositivos IoT em tempo real. Os autores exploram uma arquitetura de três camadas para aplicativos de IoT, com base em diferentes cenários de recursos computacionais. Seu foco está na descoberta dinâmica de dispositivos IoT e no gerenciamento de suas conexões. No entanto, suas configurações são limitadas ao hardware, o que pode tornar sua solução dependente de dispositivos ou aplicativos específicos. Além disso, a flexibilidade do volume de dados não foi abordada.

Xia *et al.* (2014) apresentam uma plataforma para descarregamento de dados chamada *Phone2Cloud*. O objetivo é melhorar a eficiência energética dos *smartphones* e melhorar o desempenho da aplicação. Ao contrário dos *frameworks* anteriores, os autores se concentram na realização de uma análise totalmente quantitativa sobre economia de energia do sistema através da realização de experimentos e experiências de cenários. *Phone2Cloud* é uma estrutura de descarga semiautomática. Para executar aplicativos na nuvem e receber os resultados, os aplicativos precisam ser modificados manualmente durante a etapa de preparação para permitir a execução em servidores em nuvem. A decisão de *offloading* é baseada em uma análise estática, considerando o limite de tolerância de atraso do usuário. Para aplicativos tolerantes a atrasos, o *framework* usa um modelo simples para esperar conectividade *WiFi*. O limite é definido com base em previsões para atrasar as transferências com a finalidade de transmitir mais dados via *WiFi*, respeitando o limite de tolerância da aplicação.

Compartilhando a mesma preocupação, mas de uma perspectiva diferente, Qian e Andresen (2015) apresentam um sistema, denominado Jade, que monitora o estado do aplicativo

e do dispositivo e que decide automaticamente onde o código deve ser executado. O objetivo do *Jade* é maximizar os benefícios do *offloading* de computação com consciência energética para aplicativos móveis, ao mesmo tempo minimizar o fardo para os desenvolvedores criarem essa aplicação. Durante o particionamento, as aplicações são particionadas ao nível de classe em *Jade* com base nas informações coletadas. Como preparação, o sistema verifica o estado do aplicativo e do dispositivo monitorando os custos de comunicação, variação de carga de trabalho e estado de energia. A estrutura fornece um modelo de programação sofisticado com um conjunto completo de *API*, para que os desenvolvedores tenham controle total: como a aplicação é particionada e como o código remoto interage com o código local. A decisão de *offloading* é tomada em tempo de execução para decidir onde o código deve ser executado. O *Jade* suporta dois tipos de servidores: (1) servidores Android e (2) servidores não-Android que executam sistemas operacionais como *Windows* e *Linux*. Os servidores não-Android devem ter Java instalado para suportar o *Jade*. O *Jade* pode alterar dinamicamente a sua decisão de *offloading* de acordo com o estado do dispositivo e assim reduzir eficientemente o consumo de energia de dispositivos móveis.

[Ra et al. \(2011b\)](#) apresentam um mecanismo Odessa que consegue avaliar o estado atual dos recursos do dispositivo e a complexidade da tarefa que será executada por meio de um mecanismo denominado "*Lightweight Application Profiler*". Essa ferramenta tem como objetivo monitorar o sistema móvel como um todo e fornecer os dados coletados para um decisor de *offloading*, sem impactar a performance da aplicação, executando em *background*. O Odessa surge como uma solução para cenários onde não existem informações prévias sobre o ambiente de execução, o que significa que ele pode se auto adaptar como necessário à aplicação em execução, ao estado da conexão de rede e aos tempos de processamento medidos conforme o aplicativo é executado.

[Lingaraj, Biradar e Patil \(2018\)](#) desenvolveram uma pesquisa que busca estender o *middleware* Agilla ([FOK; ROMAN; LU, 2009](#)) para redes de sensores sem fio a um nível mais centrado na reutilização de recursos para minimizar o seu desperdício. O Agilla sugere o estabelecimento de tuplas de processamento, entre nós e aplicações, dentro da rede de sensores, formando pares mutuamente exclusivos. Já o Eagilla, proposto por [Lingaraj, Biradar e Patil \(2018\)](#), sugere que essas tuplas podem ser integradas. Por esta perspectiva, ao invés de aplicar a descarga computacional, os autores buscam aproveitar os recursos contidos na rede ao máximo. A técnica sugere um *middleware* que tem a capacidade de agrupar nós de acordo com as aplicações que eles executam. Devido à aplicação do agrupamento ocorrer em nível de aplicação, como se os nós fossem recursos, muitas vezes os mesmos nós podem pertencer ao conjunto de recursos de aplicações diferentes. Dessa forma, a alocação de recursos garante que um mesmo recurso possa ser utilizado por mais de uma aplicação, permitindo assim o aumento na performance geral do sistema, por evitar desperdícios.

[Neto et al. \(2018a\)](#) propõem um *framework* com um mecanismo de decisão que determina

se uma tarefa é descarregada para um servidor externo, denominado ULOOF (*User Level Online Offloading Framework*). Os autores estimam o consumo de energia e o tempo de execução antes da decisão de execução local ou externamente, em seguida atualizam a estimativa de custo. Assim, o *framework* pode se adaptar ao cenário atual dinamicamente. As funções de custo de tempo de execução e consumo de energia utilizam dados históricos para a tomada de decisão de descarga. ULOOF é um *framework* baseado na descarga de métodos, desta forma, o especialista da aplicação deve informar na aplicação, por meio de anotações, quais métodos podem ser descarregados. Ainda, ULOOF pode decidir pela descarga em um servidor remoto ou em um *smartphone* Android próximo. No entanto, a decisão de descarga é baseada na estimativa de complexidade do algoritmo, o que pode ser difícil para os desenvolvedores identificar. Além disso, o ULOOF não define, explicitamente, uma arquitetura de comunicação. Ademais, apesar da abstração de implementação de código no servidor, o servidor é dependente do código executado no dispositivo móvel e de suas bibliotecas, i.e., não é possível que o código seja otimizado no servidor ou implementado de maneira diferente.

Com a finalidade de obter respostas, em tempo real, das mudanças nos comportamentos dos usuários, algumas propostas estão relacionadas a computação de borda emergente. Assim, é possível que algumas das tarefas de processamento de dados e tomada de decisão possam ser feitas nas fontes de dados ou próximas a elas (ou seja, nós sensores) ao invés da nuvem remota. Apesar de conceitualmente serem similar, a computação de névoa apresenta diferenças cruciais em relação a nuvem e deve ser investigada particularmente, como por exemplo a alta dinâmica, mobilidade, problemas de energia, confiabilidade e heterogeneidade (SOUZA *et al.*, 2018b).

Azimi *et al.* (2017) propõem uma arquitetura de computação hierárquica para sistemas de monitoramento de saúde de pacientes baseados em IoT. Os autores propõem uma nova metodologia para execução de sistemas auto adaptáveis por meio de uma técnica de gerenciamento de ciclos de *feedback* MAPE-K, considerando uma arquitetura de três camadas. A arquitetura proposta se beneficia dos recursos da computação em névoa e da computação em nuvem. Além disso, os autores terceirizam os procedimentos pesados de treinamento na nuvem, enquanto exploram o conhecimento da camada de névoa. Os autores não consideram decisões de descarga de dados, pois os procedimentos nas camadas já estão pré-definidos estaticamente na metodologia.

Gia *et al.* (2015) exploram o conceito de computação de névoa em *gateways* inteligentes, aplicada em um cenário de saúde. Os nós de névoa consistem em *gateways* fixos implantados no hospital que recebem os dados coletados dos *wearables* dos pacientes e fornecem técnicas e serviços avançados, como mineração de dados incorporados, armazenamento distribuído e serviço de notificação na borda da rede. Embora os autores evidenciam a alocação de recursos de TI, não é definida um mecanismo de comunicação para lidar com a volatilidade desses recursos no cenário de saúde, onde existem muitas aplicações consideradas críticas, em que a perda de conexão pode levar sérios danos aos sistemas e aos dados do paciente.

La *et al.* (2019) propõem uma arquitetura com um classificador de aprendizagem de

máquina para detectar o comportamento do usuário. Seguindo a arquitetura de névoa, o classificador fica no dispositivo de *gateway* para extrair conhecimento da camada de aplicação. Assim, a decisão de agendamento do MAC pode ser tomada sem o envio de dados para a nuvem, obtendo respostas em tempo real. O processamento de dados no *gateway* (i.e., ambiente de névoa) evita longos atrasos congestionamento de banda e desperdício de energia. O foco principal dos autores é na redução da latência da rede e na economia de largura de banda e energia, além de monitorar os usuários efetivamente por meio de respostas em tempo real.

Zhu *et al.* (2019) exploram um trade-off entre o custo computacional e o custo de comunicação em uma rede de névoa por meio de uma arquitetura de computação híbrida através da cooperação entre nós de névoa e redes definidas por softwares. Para determinar o volume e o destino dos dados de descarregamento, os autores formulam um problema de maximização do bem-estar social no qual foram considerados o custo computacional local, o custo de comunicação do descarregamento e o custo computacional nos nós de névoa. Os autores utilizam um método JacobiProximal de direção alternada dos multiplicadores (ADMM) para resolver o problema de otimização de maneira distribuída e paralela, considerando a carga que pode ser processada em cada dispositivo. Um controlador baseado em redes definida por software (SDN) é utilizado para fornecer e controlar as últimas atualizações no nós de névoa.

A Tabela 1 apresenta as principais características dos trabalhos apresentados nesta Seção.

Tabela 1 – Síntese das principais características dos trabalhos relacionados

Trabalhos	Características						
	Névoa	Middleware	Adaptabilidade	Decisão de descarga	Eficiência energética	Latência	Intermitência
Fortino <i>et al.</i> (2014)	Não	Sim	Não	Não	Não	Não	Não
Xia <i>et al.</i> (2014)	Sim	Não	Sim	Sim	Sim	Sim	Não
Gia <i>et al.</i> (2015)	Sim	Não	Não	Não	Sim	Sim	Não
Qian e Andresen (2015)	Não	Não	Sim	Sim	Sim	Sim	Não
Vinh <i>et al.</i> (2015)	Não	Sim	Não	Não	Não	Não	Não
Khan <i>et al.</i> (2016)	Não	Sim	Não	Não	Sim	Não	Não
Maia <i>et al.</i> (2015)	Não	Sim	Não	Não	Não	Não	Não
Gomes <i>et al.</i> (2015)	Não	Sim	Não	Não	Não	Não	Sim
Thangaraj <i>et al.</i> (2016)	Não	Sim	Sim	Sim	Sim	Não	Não
Mane e Surve (2016)	Não	Sim	Não	Não	Não	Não	Não
Mano <i>et al.</i> (2016)	Sim	Não	Não	Não	Não	Sim	Não
Prado, Ortiz e Puig (2017)	Sim	Sim	Não	Não	Sim	Não	Não
Patel, Ali e Sheth (2017)	Sim	Não	Não	Não	Não	Não	Não
Daneels <i>et al.</i> (2017)	Sim	Não	Não	Não	Não	Sim	Não
Azimi <i>et al.</i> (2017)	Sim	Não	Sim	Não	Não	Sim	Não
Kang, Kim e Choo (2017)	Sim	Sim	Não	Não	Sim	Sim	Não
Neto <i>et al.</i> (2018a)	Não	Não	Sim	Sim	Sim	Sim	Não
Németh <i>et al.</i> (2018)	Sim	Não	Sim	Sim	Não	Sim	Não
La <i>et al.</i> (2019)	Sim	Não	Não	Não	Sim	Sim	Não
Zhu <i>et al.</i> (2019)	Sim	Não	Sim	Sim	Não	Sim	Não
Nossa solução	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Importante destacar que a maioria dos trabalhos não consideram *middleware* em uma arquitetura de névoa. Apenas Prado, Ortiz e Puig (2017), que utiliza uma arquitetura para integrar dispositivos de borda a nuvem por meio de *brokers*, e Kang, Kim e Choo (2017), que utiliza um *gateway* inteligente como intermediário no contexto de névoa. Porém, nenhum dos

dois especifica como lidar com as perdas de conexões em ambientes voláteis como IoT e não consideram adaptabilidade do sistema. Por outro lado, a maioria dos trabalhos que consideram adaptabilidade, levam em consideração a decisão de descarregamento em diferentes ambientes. Apenas [Azimi et al. \(2017\)](#) consideram uma metodologia de sistemas auto adaptativos, como MAPE-K. [Azimi et al. \(2017\)](#) é o trabalho que mais se assemelha ao estudo desta tese, uma vez que [Azimi et al. \(2017\)](#) propuseram uma arquitetura de névoa multicamadas e auto adaptativa. No entanto, estes autores não consideram o ambiente de IoT com comunicação intermitente e dinâmico para a decisão de transmissões para a camadas de névoa ou nuvem. Além disso, os dispositivos de borda são utilizados somente para a coleta dos dados, desperdiçando os recursos próximos dos usuários.

Embora a inclusão de elementos de névoa possa reduzir a latência da rede, o tráfego na rede e o consumo de energia, existem desafios que ainda trazem problemas para a total efetividade desse tipo de arquitetura. A volatilidade e dinamicidade são uns desses desafios, pois a desconexão de dispositivos ou sensores podem prejudicar o funcionamento dos sistemas como um todo. Alguns trabalhos na literatura consideram ambientes IoT com recursos de TI estáticos sendo utilizados como fonte de dados, desconsiderando o total aproveitamento desses recursos com processamento e armazenamento ([SOUZA et al., 2018b](#)). O trabalho proposto nesta tese tem foco no maior aproveitamento de recursos do ambiente IoT por meio de uma arquitetura multicamadas de névoa capaz de fornecer respostas em tempo real. Além disso, foi desenvolvido um *middleware* para aumentar a efetividade do ambiente de névoa, por meio de um *framework* baseado em ciclos de *feedback* para lidar com a adaptabilidade em relação ao fluxo de dados usando decisão de descarga, e um módulo de comunicação assíncrona e desacoplada para o gerenciamento de mensagens durante o funcionamento de sistemas IoT, uma vez que os ambientes IoT são por natureza voláteis e dinâmicos.

3.3 Considerações Finais

Neste capítulo, foram apresentados os trabalhos relacionados ao estudo de pesquisa desta tese. Os trabalhos encontrados foram divididos em dois grupos. O primeiro abordou trabalhos que abrangem ambientes inteligentes de IoT. Neste grupo, muitos trabalhos consideram suas soluções ao ambiente de monitoramento de saúde. O segundo grupo consiste em trabalhos que consideram arquiteturas de névoa de descarregamento de dados para a nuvem. As principais lacunas encontradas nos trabalhos relacionados foram a adaptabilidade das soluções em relação ao estado do sistema como um todo; o compartilhamento de recursos no ambiente por meio da decisão de descarga de dados, considerando os dispositivos de borda como provedores de recursos; o uso de um *middleware* que forneça o gerenciamento de fluxo de dados e interoperabilidade entre os dispositivos e serviços; e mecanismos de comunicação que considere o ambiente volátil e dinâmico de IoT para as comunicações entre os dispositivos, principalmente no contexto de saúde com aplicações de necessidades de respostas em tempo real.

ARQUITETURA BASEADA EM CICLOS DE FEEDBACK E NÉVOA MULTICAMADA PARA A DESCARGA DINÂMICA DE DADOS

Neste capítulo, é apresentada uma arquitetura baseada em controle de *feedback* e névoa de camadas hierárquicas (i.e., de múltiplas camadas) de *Fog-to-Cloud* (F2C) para a descarga adaptativa de dados (do inglês, *self-adaptive data offloading*) em ambientes IoT. Para a realização da descarga adaptativa, um *middleware* foi proposto e dividido em duas partes principais: (1) um *framework* baseado no modelo MAPE-K para realizar a descarga adaptativa nos ambientes de execução (i.e., ambientes de névoa), e (2) um módulo para comunicação desacoplada entre os dispositivos e o servidor remoto (por exemplo, servidor de névoa ou nuvem), baseado no paradigma de comunicação *publish-subscribe* para ambientes intermitentes e hostis (i.e., ambientes onde sensores e dispositivos podem se desconectar facilmente).

4.1 Arquitetura Proposta

Este estudo apresenta uma arquitetura baseada em ciclos de *feedback* e névoa multicamadas para a realização adaptativa de descarregamento dinâmico de dados, com o objetivo de mitigar o uso dos recursos computacionais de ambientes IoT. Uma arquitetura de névoa que considera ciclos de *feedback* para fornecer um mecanismo genérico para auto adaptação de descarga é uma abordagem inovadora e promissora (SOUZA *et al.*, 2018b). Para a modelagem da arquitetura, foi desenvolvido um *middleware* distribuído para dispositivos móveis. O *middleware* proposto baseia em *Indie Fog* e foi projetado para realizar a comunicação confiável e assíncrona entre os dispositivos e o servidor local por meio do paradigma *publish-subscribe*. Além disso, foi desenvolvido um *framework* baseado em MAPE-K que considera ciclos de *feedback* para a realização de descarga de dados em uma arquitetura de névoa, denominado AMOk (*Auto-adaptive Mobile Offloading over Knowledge*). O *framework* desenvolvido é

acoplado ao *middleware*, uma vez que o *middleware* é genérico e pode suportar novas políticas de gerenciamento de recursos, podendo apenas serem adicionadas como um *plugin* em tempo de execução ou instanciadas ao iniciar o *middleware*. A arquitetura MAPE-K fornece instruções para o gerenciamento automatizado de unidades computacionais e especifica o comportamento do sistema por meio de componentes (AZIMI *et al.*, 2017). Nesta pesquisa, tais instruções são utilizadas para auxiliar na decisão de descarga de dados.

A arquitetura proposta foi baseada em névoa multicamadas e computação autônoma para ambientes IoT (por exemplo, ambientes compostos por aplicações críticas de saúde e intolerantes ao atraso). As tecnologias IoT fornecem monitoramento eficiente, bem como podem realizar a coleta de dados diretamente de dispositivos em tempo real (AZIMI *et al.*, 2017; SOUZA *et al.*, 2018a). Além disso, essas tecnologias podem se comunicar e colaborar umas com as outras usando rede com ou sem fio para fornecer serviços de alta qualidade para os usuários finais (AZIMI *et al.*, 2017; SOUZA *et al.*, 2018a).

O modelo F2C (Ver Figura 6) prioriza as camadas mais próximas dos usuários para o processamento de dados, uma vez que o processamento pode ser distribuído de forma colaborativa da névoa para a nuvem (SOUZA *et al.*, 2018b). Assim, a arquitetura proposta consiste na alocação inteligente e eficiente dos recursos de processamento, utilizando camadas de névoa capazes de processar dados brutos o mais próximo possível das fontes geradoras de dados, com o objetivo de reduzir o tráfego de dados para nuvem e a latência de rede (GROUP *et al.*, 2017). Importante salientar que o *middleware* proposto conecta os dispositivos do usuário de forma colaborativa na primeira camada de névoa, estabelecendo o conceito de *Indie Fog*. Além disso, cada camada possui uma instância do *middleware* proposto (*MQTT Broker*), as quais interagem entre si por meio da comunicação baseada na arquitetura cliente-servidor. Essa comunicação é permitida pelo barramento de eventos que é responsável por gerenciar as transmissões de forma assíncrona usando o paradigma *publish-subscribe*. É possível perceber na Figura 6, que a arquitetura baseada em multicamadas permite que a descarga dos dados seja feita gradativamente da névoa para a nuvem. Com base em Azimi *et al.* (2017), foi projetado uma arquitetura bem definida e distribuída em duas camadas de névoa (*Fog Layers*) e uma de nuvem (*Cloud*), como prova de conceito. No entanto, esta arquitetura é genérica e pode ser projetada com mais níveis de camadas de névoa. As camadas desta arquitetura podem ser descritas da seguinte forma:

- *Fog Layer 1*: Caracterizada por sua proximidade de recursos (processamento, memória, dados) aos usuários finais, geralmente garantindo acesso de baixa latência, porém é composta com dispositivos IoT móveis os quais possuem alta volatilidade, como *smartphones*, *laptops* ou *wearables*. Também pode ser composta por uma rede distribuída de dispositivos móveis para ter maior poder de processamento. Importante destacar que tais dispositivos são alimentados por baterias, as quais têm tempo de vida útil limitado.
- *Fog Layer 2*: Caracterizada por menor volatilidade e com maior poder de processamento

Figura 6 – Arquitetura em camadas hierárquicas de *Fog-to-Cloud* (Adaptado de Group *et al.* (2017))

e armazenamento que a *Fog Layer 1*. Também pode ser vista como ponto de agregação dos recursos localizados em uma vizinhança ampla, o que garante uma latência média ($2 - 4\ ms$ (JIN; LEE, 2019)). Por exemplo, uma *Fog Layer 2* pode ser configurada com um servidor local em um hospital ou casa de repouso.

- *Cloud*: Caracterizada por recursos de nuvem interconectados às camadas de névoa através de redes *multihops* (i.e. múltiplos saltos). Possuem um alto poder de processamento e armazenamento capaz de fornecer um amplo espectro de serviços. Por exemplo, realizar análises complexas, como análise de séries temporais, e fornecer notificações para serem analisadas posteriormente por especialistas. No entanto, possui uma alta latência na resposta do servidor ($106 - 108\ ms$ (JIN; LEE, 2019)).

O foco desta arquitetura é principalmente nas camadas mais próximas do usuário, em que se concentra um *middleware* composto por dois módulos: (1) um *framework* que aborda os conceitos da MAPE-K para o desenvolvimento de um modelo de gerenciamento de recursos baseado em ciclos de *feedback* para o controle do fluxo dos dados de forma adaptativa, e (2) um módulo responsável por gerenciar a comunicação e conexão dos dispositivos IoT por meio do paradigma *publish-subscribe*. Na *Fog Layer 1*, temos os dispositivos que além de coletar os dados do ambiente ou do paciente, também podem realizar o pré-processamento desses dados. Na *Fog Layer 2*, temos um servidor remoto com maior poder de processamento e armazenamento capaz de responder e fazer análises de dados em tempo real. O monitoramento e adaptação da arquitetura concentra-se na decisão de descarregamento dos dados em ambientes IoT de névoa.

Tanto os dispositivos locais (*Fog Layer 1*) quanto o servidor remoto (*Fog Layer 2*) possuem uma instância do *middleware* proposto para que essas instâncias trabalhem em colaboração para gerenciar o processamento ou descarregamento dos dados nas camadas superiores da arquitetura, seja ela de névoa ou nuvem. Vale destacar que, a *Fog Layer 2* possui latência média enquanto a nuvem possui uma alta latência. A arquitetura dividida em multicamadas permite lidar melhor com a latência em sistemas de tempo real, uma vez que aumenta a disponibilidade de recursos gradativamente, reduz o tempo de resposta e garante maior qualidade nos serviços (SOUZA *et al.*, 2018b).

O sensoriamento é feito através da coleta de dados dos sensores dos dispositivos do cotidiano do usuário (por exemplo, *smartwatch*, *smartphone* ou câmeras), que são os produtores de informação. A primeira camada de névoa é composta por dispositivos conectados capazes de corresponder as demandas de processamento de tarefas exigidas no ambiente distribuído de sensoriamento criado pelo *middleware*, como mostra a Figura 7. Vale ressaltar que os sensores e atuadores estão presentes na *Fog Layer 1* (Ver Figura 6), considerando os dispositivos do usuário para coletar e processar os dados (Ver Figura 7). Enquanto que, a segunda camada de névoa possui um poder de processamento mais elevado para garantir um equilíbrio no desempenho dos sistemas de sensoriamento em ambientes inteligentes, uma vez que esta camada provê uma latência abaixo do que a nuvem provê e um poder de processamento maior do que os dispositivos de borda (SOUZA *et al.*, 2018b; MUTLAG *et al.*, 2019). Nesta camada, uma análise aprofundada (por exemplo, análise de séries temporais) baseada no histórico de dados do usuário pode ser realizada sem a necessidade de enviar os dados para a nuvem, uma vez que este ambiente de execução possui maior poder de processamento e capacidade de armazenamento que os dispositivos da *Fog Layer 1*.

4.2 LOADING - a middLeware fOr Adaptative Data offloadING

A tecnologia de *middleware* em IoT fornece uma plataforma de custo-eficiência, em que o hardware e o software podem ser reutilizados, além de auxiliar no desenvolvimento de aplicações, resolvendo vários problemas relacionados aos sistemas distribuídos (FERSI, 2015; RAZZAQUE *et al.*, 2016). Apesar das pesquisas relevantes em IoT crescentes, ainda não existe padronização em *middlewares* para ambientes pervasivos (RAZZAQUE *et al.*, 2016; CRUZ *et al.*, 2018). Entretanto, existem maiores exigências e questões de *design* para apoiar o desenvolvimento de *middlewares* em ambientes inteligentes, e.g., IoT (FERSI, 2015; RAZZAQUE *et al.*, 2016; CRUZ *et al.*, 2018). Diante disso, nesta seção é apresentado o *middleware* desenvolvido, denominado LOADING (middLeware fOr Adaptative Data offloadING).

LOADING consiste na descarga adaptativa de dados em ambientes de névoa para mitigar o uso de recursos computacionais em ambientes IoT usando ciclos de *feedback* e *publish*-

subscribe. Nesta pesquisa, foi considerado que os sensores estão acoplados nos mesmos dispositivos de borda presentes na *Fog Layer 1*, desta forma, ao coletar os dados, o mesmo dispositivo pode tomar a decisão sobre onde realizar o processamento. A Figura 7 mostra como o *middleware* proposto atua durante a coleta de dados na camada *Fog Layer 1*. As setas verdes representam o fluxo de comunicação entre os módulos do *middleware*. Os módulos de cor laranja representam os componentes do *framework* desenvolvido, enquanto que o módulo de cor azul representa o *Broker* onde encontram-se o barramento de eventos do protocolo *Message Queue Telemetry Transport* (MQTT) (STANFORD-CLARK; NIPPER, 2014), o modelo de decisão de descarga de dados e os serviços ativos que podem ser executadas na camada *Fog Layer 1*. Na camada *Fog Layer 2*, os principais componentes que podem ser instanciados no *middleware* são apresentados. Por ser mais robusta que a camada *Fog Layer 1*, a camada *Fog Layer 2* suporta serviços mais complexos, como Análise de Dados. Vale ressaltar que tanto o *framework* quanto o *Broker* de comunicação estão presentes nas camadas *Fog Layer 2* e *cloud*.

Figura 7 – *LOADING* em uma arquitetura F2C para ambientes IoT

O *middleware* proposto possui duas frentes principais como contribuição descritas a seguir: (1) um *framework* para descarga adaptativa baseado em MAPE-K, e (2) um módulo de comunicação em ambientes intermitentes IoT usando o paradigma *publish-subscribe*.

4.2.1 *AMOk - auto adaptive Mobile Offloading over Knowledge*

Como mencionado anteriormente, a arquitetura de referência para o controle cíclico MAPE-K fornece instruções suficientes para sistemas de auto adaptação. Diante disso, foi desenvolvido um *framework*, denominado AMOk, para dispositivos móveis que permite um controlador ter módulos individuais capazes de realizar as seguintes funções:

- monitorar uma aplicação com o uso de sensores;
- analisar os dados monitorados;
- planejar uma estratégia de execução;
- executar a estratégia por meio de atuadores.

A arquitetura foi escolhida por ter práticas de reutilização por meio dos seus módulos bem definidos e execução sequencial destes, o que facilita o entendimento do funcionamento do *framework*, além de sua fácil manutenção e adaptação. Além disso, a modularização do *framework* na arquitetura fornece um controle mais eficiente de acesso em relação ao domínio da aplicação, visto que a interação entre o *framework* e a aplicação é realizada sempre através de um componente Sensor e Atuador (Ver Figura 8). Assim, o *framework* proposto busca simplificar o desenvolvimento de aplicações inteligentes em relação à utilização dos recursos computacionais disponíveis. Este *framework* foi projetado para dispositivos móveis com o objetivo de abstrair a adaptação do sistema independente de aplicações, fornecendo um arcabouço para o desenvolvimento de aplicações adaptativas. Além disso, o *framework* foi formulado como um módulo independente que pode se conectar a qualquer aplicação evitando qualquer adaptação de código relacionada à instalação do *framework*. Em outras palavras, o *middleware* pode instanciar os componentes de adaptação do *framework* quando for requerido.

Figura 8 – MAPE-K (IBM, 2005).

Com base na referência da arquitetura MAPE-K (IBM, 2005), o *framework* AMOk especifica quatro componentes principais: Monitor, Analyser, Planner e Executer. Importante ressaltar que todos os componentes estão ligados a uma base de conhecimento a qual fornece dados de estados anteriores do sistema para que todos os componentes operem com base nos dados. É possível visualizar na Figura 8 que o componente Sensor coleta dados de um

elemento gerenciado. Nesta pesquisa, o elemento gerenciado compreende as variáveis do sistema de *hardware* (i.e., ciclos e capacidade da *Central Process Unit* (CPU)) e o tamanho dos dados de entrada para conservar o custo computacional de um determinado dispositivo ao realizar o processamento dos dados de entrada. Assim, foi desenvolvido um algoritmo para a descarga dinâmica de dados baseada no estado atual do dispositivo local (Ver Figura 9). O algoritmo foi projetado de forma recursiva para que, quando o custo local for maior que o custo remoto, os dados sejam descarregados e processados na camada superior.

Figura 9 – Fluxograma do algoritmo da decisão de descarga de dados

Fonte: Elaborada pelo autor.

A Figura 9 apresenta o fluxograma de como o *framework* foi modelado, de acordo com a arquitetura de referência MAPE-K, para realizar a descarga de dados. Importante destacar que, a implementação das funções cabe ao especialista da aplicação definir, uma vez que é proposto um *framework* genérico para atender diferentes aplicações adaptativas. Cada componente do *framework* AMOk é descrito a seguir:

- Monitor é responsável por coletar os detalhes do estado do dispositivo, i.e., dos recursos gerenciados pelo dispositivo (por exemplo, capacidade oferecida, taxa de transferência, memória, consumo de processamento). Este componente está ligado aos sensores dos dispositivos por meio do componente Sensor para sensoriar e determinar os dados a serem analisados, fornecendo uma funcionalidade mínima necessária para coletar dados

do dispositivo, como memória, CPU, conexão da Internet, memória, entre outros. O componente Monitor é responsável por instanciar e gerenciar o componente Sensor, o qual interage com o domínio da aplicação, coletando dados desta para utilização no *framework*. A partir dos dados de entrada, é realizado o pré-processamento dos dados coletados pelo sensor antes de serem adicionados à base de conhecimento. Desta forma, é possível agregar e correlacionar informações vindas dos sensores.

- Analyser faz a análise de dados sobre as informações fornecidas pelo componente Monitor, explorando os metadados e as técnicas de previsão. Após a adição dos valores relacionados ao estado do dispositivo na base de dados, o componente Analyser consulta estes valores na base de conhecimento para realizar o cálculo de métricas (por exemplo, custo energético do dispositivo) para a realização ou não da descarga dos dados a serem processados. Assim, é possível estimar o custo computacional de processamento no ambiente local C^l e o custo de descarga C^r na camada de computação superior, a partir do estado atual do sistema (Ver seção 4.2.1.2 e 4.2.1.3).
- Planner cria ou seleciona um procedimento para executar uma alteração desejada no recurso gerenciado, a partir dos dados de entrada fornecidos pelo componente Analyser. O componente Planner pode assumir diferentes formas, desde um único comando até um fluxo de trabalho complexo. A partir do custo computacional estimado pelo componente Analyser é possível definir o comportamento da aplicação na qual o *framework* está atuando. O planejamento deste componente visa tomar a decisão sobre o processamento local dos dados de entrada ou a realização da descarga dos dados. Nesta pesquisa, este componente decide processar os dados localmente se o custo de processamento local for menor que o custo de descarga. Caso contrário, ele decide que a descarga dos dados seja feita para a camada superior na arquitetura. Assim, o componente Executer aponta para o componente Analyser e este passo se repete até que o custo atual local seja menor que o custo da camada superior.
- Executer fornece mudanças necessárias no sistema, i.e., altera o comportamento do recurso gerenciado usando atuadores, com base nas ações recomendadas pelo componente Planner. O componente Executer instancia e gerencia o componente Actuator, que interage com o domínio da aplicação, invocando a ação a ser realizada de acordo com os parâmetros gerados pelo componente Planner. Caso necessário, o componente Actuator é também responsável por realizar algum pós-processamento dos dados antes de serem adicionados à base de conhecimento. O componente Actuator por sua vez, atua no sistema processando os dados de entrada localmente ou realizando a descarga dos dados. Nesta pesquisa, a mudança do sistema é a troca entre processar ou descarregar os dados e os atuadores são os componentes de chamada do método de processamento ou de comunicação.

A Figura 10 apresenta como foi modelado o diagrama de classes do *framework* proposto.

Figura 10 – Diagrama de classes do Framework AMOk

Fonte: Elaborada pelo autor.

4.2.1.1 Política de descarga de dados

A decisão de descarga de dados considerou o consumo de energia e o tempo de processamento para a realização do processamento dos dados de entrada (LI *et al.*, 2018; HUANG *et al.*, 2019). No entanto, a decisão é baseada na arquitetura proposta, desta forma, foi considerado o custo computacional para o dispositivo processar os dados localmente e o custo do dispositivo para enviar os dados para a camada seguinte de computação. Por exemplo, os dispositivos da *Fog Layer 1* calculam o custo do processamento local e o custo de enviar os dados para *Fog Layer 2*.

Nesta pesquisa, foi modelada a decisão com base na latência L_{app} e no consumo de energia da aplicação E_{app} . Na *Fog Layer 1*, cada dispositivo d é responsável por estimar o seu custo computacional local C^l e o custo computacional da *Fog Layer 2* C^r . Em outras palavras, na arquitetura hierárquica proposta, cada camada tem conhecimento do poder computacional da camada superior. Isso ocorre porque a cada registro de serviços na camada superior, a instância do *middleware* da camada superior atualiza os serviços ativos da camada de baixo, bem como as informações de poder computacional onde os serviços são executados. O coordenador do *middleware* é responsável por compartilhar essa informação com os dispositivos na rede. Assim, o dispositivo local pode estimar o (1) custo de computação local C^l , e o (2) custo de computação remoto C^r .

4.2.1.2 Custo de computação local

Para a execução do processamento local de uma determinada tarefa ta , considera-se o atraso do processamento local baseado na CPU do dispositivo (SHUKLA; MUNIR, 2017; LI *et al.*, 2018). Assim, a seguinte equação define a latência da aplicação para a execução local da tarefa:

$$L_{app}^l = \frac{T_c}{C_d} \quad (4.1)$$

onde T_c foi definido como o total de ciclos de CPU requeridos pela tarefa ta e C_d é a capacidade computacional (i.e., quantidade de ciclos por segundo da CPU) do dispositivo d . Importante destacar que a capacidade computacional pode variar entre os dispositivos.

Para estimar o consumo de energia E_{app}^l , foi considerado o quanto de energia é gasta por ciclo da CPU do dispositivo d para completar a tarefa ta localmente. Então, definimos z_n como a medida que corresponde a energia consumida por ciclo de CPU para completar a tarefa ta , em que $z_n = 10^{-27}(C_d)^2$ (WEN; ZHANG; LUO, 2012; LI *et al.*, 2018). Desta forma, a seguinte equação representa o consumo de energia da tarefa ta :

$$E_{app}^l = z_n T_c \quad (4.2)$$

Diante disso, o custo computacional local C^l para cada dispositivo d é definido pela soma da latência L_{app} com o consumo de energia E_{app}^l da aplicação para executar a tarefa ta . Ainda, foi definido α e β para representar os pesos para o custo de tempo de resposta e energia, respectivamente, para a execução da tarefa ta . Vale destacar que os valores de L_{app} e E_{app}^l foram normalizados. A equação 4.3 representa o custo computacional local:

$$C^l = \alpha L_{app} + \beta E_{app}^l \quad (4.3)$$

onde, $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ e $w_t + w_e = 1$. Importante destacar que os pesos podem ser diferentes para cada tipo de tarefa.

4.2.1.3 Custo de computação remota

Para calcular o custo da computação remota para a descarga dos dados em relação a latência da aplicação em tempo real e o consumo de energia, foi necessário considerar o custo de envio dos dados para a camada superior na arquitetura (por exemplo, névoa ou nuvem). Vale destacar que a latência da névoa é menor que a latência da nuvem, uma vez que está fisicamente localizada mais próxima do usuário, desta forma, a descarga na névoa é mais adequada para sistemas em tempo real, os quais são intolerantes a atraso (SOUZA *et al.*, 2018b; MUTLAG *et al.*, 2019).

Similarmente ao custo local, o custo de latência da aplicação para a descarga dos dados considerou o atraso do processamento baseado na CPU do servidor (SHUKLA; MUNIR, 2017). Além disso, quando um dispositivo realiza a descarga dos dados de entrada, é necessário considerar os recursos gastos na transmissão dos dados e na recepção do resultado (ZHU *et al.*, 2017; LI *et al.*, 2018; HUANG *et al.*, 2019). O custo de transmissão entre as camadas varia de acordo com a camada atual, local onde os dados de entrada estão sendo analisados, uma vez que os módulos do *middleware* são sincronizados em todas as camadas. Assim, o *MQTT Broker* é responsável por identificar em qual camada da arquitetura os dados de entrada se encontram, antes de calcular o custo computacional de latência e energia. Em seguida, para calcular o custo de latência da aplicação durante a descarga de dados, utiliza-se a equação 4.4 para definir a latência L_{app}^f da descarga no servidor de névoa ou a equação 4.5 para definir a latência L_{app}^n da descarga na nuvem.

$$L_{app}^f = \frac{T_c^f}{C^f} + \zeta \quad (4.4)$$

$$L_{app}^n = \frac{T_c^n}{C^n} + \zeta \quad (4.5)$$

O total de ciclos de CPU requeridos para completar a tarefa ta no servidor de névoa ou na nuvem, é representado por T_c^f e T_c^n , respectivamente. C^f e C^n representam a capacidade computacional (i.e., quantidade de ciclos por segundo da CPU) do servidor de névoa e nuvem, respectivamente. Segundo Jin e Lee (2019), o tempo de transmissão e recepção para descarga dos dados compreende um intervalo de $2 \leq \zeta \leq 4 \text{ ms}$ quando a descarga dos dados é feita na névoa e um intervalo de $106 \leq \zeta \leq 108 \text{ ms}$ quando a descarga dos dados é feita na nuvem. No entanto, o tempo de transmissão e recepção para descarga dos dados na nuvem pode variar dependendo da distância do servidor de nuvem. Diante disso, utilizamos os valores apresentados por Jin e Lee (2019) para avaliação de nossa proposta.

Para estimar o consumo de energia da aplicação na descarga dos dados E_{app}^r , foi considerado o custo de energia gasto pelo dispositivo para descarregar os dados de entrada, ou no servidor de névoa ou na nuvem. Similar a Shukla e Munir (2017), foi considerado a energia gasta pelo dispositivo para transmitir γ e o tamanho dos dados a serem descarregados T_f (Ver equação 4.6).

$$E_{app}^r = \gamma T_f \quad (4.6)$$

Portanto, combinando o custo da latência com o custo de energia, normalizados, o custo da descarga computacional pode ser representada por C^f (Ver equação 4.7) quando a descarga é na névoa ou por C^n (Ver equação 4.8) quando a descarga é na nuvem. Assim como na equação 4.3,

α e β representam os pesos para o custo de tempo de resposta e energia, respectivamente, para a execução da tarefa ta no servidor de névoa ou nuvem.

$$C^f = \alpha L_{app}^f + \beta E_{app}^r \quad (4.7)$$

$$C^n = \alpha L_{app}^n + \beta E_{app}^r \quad (4.8)$$

4.2.2 Conectividade e comunicação

Apesar da eficiência do *framework* na adaptabilidade de decisão de descarga de dados, um ambiente inteligente composto por dispositivos móveis (por exemplo, *smartphones*, *smartwatches*, *laptops*, entre outros) é volátil e a conexão e desconexão de tais dispositivos prejudicam a estabilidade de sistemas em ambientes IoT (FERSI, 2015; RAZZAQUE *et al.*, 2016). A combinação de névoa e descarregamento de dados permite a flexibilidade do sistema frente a uma grande quantidade de dados, de maneira que a arquitetura seja completamente explorada e os dispositivos de bordas fiquem menos ociosos (RA *et al.*, 2011a; BONOMI *et al.*, 2014; MUTLAG *et al.*, 2019). Por esta razão, foi desenvolvido um módulo de comunicação no *middleware* responsável por um modelo unificado capaz de interagir com dispositivos e de gerenciar informações. Este módulo permite a abstração da comunicação e ocultação de problemas, como heterogeneidade e distribuição que enfrenta-se ao interagir com dispositivos (por exemplo, diferentes aplicações executando em diferentes plataformas). Assim, o *middleware* proposto atinge duas formas de implementação do modelo *Indie Fog*:

1. **modelo clusterizado:** modelo distribuído entre os nós da mesma rede permitindo um maior desempenho de processamento dos dados de forma colaborativa, em forma de *cluster*;
2. **modelo *smartphone*:** modelo que utiliza a infraestrutura dos *smartphones* dos usuários presentes no ambiente.

Diante disso, o *middleware* desenvolvido utiliza a biblioteca Blaubot proposta em Barth e Groß (2017), a qual permite a criação de uma rede *ad-hoc* entre múltiplos dispositivos com capacidades heterogêneas e flexíveis (i.e., qualquer dispositivo pode assumir a rede como coordenador). Desta forma, os dispositivos podem colaborar e compartilhar recursos entre si por meio das interfaces de comunicação suportadas (por exemplo, *Near Field Communication* (NFC), Bluetooth, Sockets e *Wireless Local Area Network* (WLAN)) e um canal de comunicação criado entre os dispositivos e o servidor de névoa (i.e., próxima camada de processamento da névoa), formando um único nó de névoa, que na nossa arquitetura, foi denominado *Fog Layer* 1. Cada dispositivo na rede possui uma instância do *middleware*, bem como as aplicações dos

serviços fornecidos (por exemplo, aplicações de sensoriamento de saúde). Assim, quando o coordenador perder a conexão, é possível que outro dispositivo seja o novo coordenador. O coordenador é responsável por gerenciar todos os serviços ativos do servidor local e atualizar os dispositivos sempre que um serviço estiver desativado. Isto é feito por transmissões periódicas entre o coordenador e o servidor remoto. O *middleware* também permite o cadenciamento dos dados nas bordas da rede a fim de reduzir o tráfego na rede. Diante disso, considera-se o uso do paradigma *publish/subscribe*, o qual permite a difusão assíncrona de informações pela arquitetura do sistema proposto, por meio de uma abordagem flexível e assíncrona, entre os produtores (*publishers*) e os consumidores (*subscribers*) de informação (EUGSTER *et al.*, 2003; FERSI, 2015).

O uso de *Publish/Subscribe* preenche uma parte importante dos requisitos para aplicações baseadas em IoT, uma vez que eles são capazes de conceder, de forma assíncrona, múltiplos eventos aos seus respectivos *subscribers* de interesse, a partir de fontes de eventos (FERSI, 2015). A Figura 11 ilustra a comunicação baseada *publish/subscribe*. Neste modelo, os *subscribers* enviam uma mensagem de assinatura para um barramento de eventos contendo os seus interesses em um determinado evento. Um *publisher* que recebeu a mensagem de assinatura através do barramento, envia mensagens de acordo com a notificação de interesse do *subscribe*. Há também um serviço de notificação responsável por fornecer armazenamento e gerenciamento de assinaturas. Desta forma, é possível fornecer uma comunicação com desacoplamento total entre as entidades (TEKIN; SAHINGOZ, 2016; OLIVIERI; RIZZO; MORARD, 2015).

O *middleware* criado elege um dispositivo coordenador responsável por atualizar os serviços ativos que estão presentes no servidor remoto (*Fog Layer 2*). Cada dispositivo na rede possui uma instância do *middleware*, desta forma, a comunicação entre o coordenador e os dispositivos é feita pelo barramento de eventos presente no *middleware*. A Figura 11 apresenta como o protocolo de comunicação MQTT (STANFORD-CLARK; NIPPER, 2014) funciona. As setas vermelhas indicam as notificações de assinaturas dos serviços, enquanto que as setas pretas indicam as publicações no barramento de eventos. Pode-se destacar o seguinte exemplo de funcionamento: (1) o serviço de monitoramento cardíaco é registrado, (2) a aplicação assina os eventos relacionados aos batimentos, (3) o barramento de eventos atualiza os serviços ativos no dispositivo coordenador, (4) um dispositivo publica dados que representam batimentos cardíacos no barramento de eventos, (5) o barramento de eventos notifica o serviço de monitoramento cardíaco com dados de batimentos cardíacos publicados.

A Figura 12 ilustra o processo de publicação no barramento de eventos. As setas indicam a direção em que os procedimentos fluem. O dispositivo coordenador possui um mapa virtual dos serviços ativos no servidor local. Desta forma é possível informar aos dispositivos do ambiente quais são os serviços que estão ativos para o sensoriamento. Após um sensor de um dispositivo S_n gerar um evento com o dado D_n , o protocolo de correlação temporal proposto por Filho *et al.* (2015) foi utilizado para correlacionar os eventos gerados com o tempo. Assim, o conteúdo do

Figura 11 – Barramento de eventos

evento gerado é comparado com o conteúdo de eventos anteriores com o objetivo de eliminar redundâncias na comunicação e reduzir a quantidade das transmissões na rede. Em seguida, a mensagem m_n é criada e publicada no barramento de eventos, que é responsável por enviar a mensagem m_n para o servidor de névoa. Após receber a mensagem m_n na instância do barramento no servidor de névoa, o serviço S_n que está inscrito no tópico da mensagem processa os dados da mensagem. Para que o serviço S_n seja notificado com o evento da mensagem m_n , é preciso que o serviço esteja ativo e inscrito no barramento de eventos com o seu ID, o tópico do conteúdo a ser recebido e a aplicação que faz parte. Após o processamento dos dados, o barramento de eventos envia a resposta para o dispositivo da D_n . Entretanto, caso o serviço tenha detectado urgência no ambiente (e.g., queda do usuário), um componente de alerta atua de acordo com a solução de urgência implementada (por exemplo, aplicação para chamar uma ambulância, ou informar a equipe de saúde).

4.3 Avaliação quantitativa de desempenho

Nesta seção, são apresentados os resultados da avaliação de desempenho e a metodologia utilizada na constituição de resultados. Nessa avaliação, utilizamos a arquitetura HiCH proposta por Azimi *et al.* (2017) como baseline. Os resultados foram gerados por meio de simulações, com o auxílio do Network Simulator 3¹.

4.3.1 Cenário modelado

Nesta pesquisa, utilizou-se uma infraestrutura de healthcare em uma casa inteligente (FILHO *et al.*, 2015) como estudo de caso para o *offloading*. A infraestrutura supracitada possui

¹ Network Simulator 3 - NS3, <http://www.nsnam.org/>

Figura 12 – Fluxograma de comunicação

duas aplicações healthcare: (a) controle de batimentos cardíacos; (b) controle de temperatura corporal. Ambas aplicações foram modeladas para que apresentassem diferentes valores durante o monitoramento.

Para uma comparação justa, não foi usado o componente Analyser da arquitetura de Azimi *et al.* (2017). Isto porque os autores utilizam o componente Analyser na nuvem para treinar o algoritmo de aprendizagem de máquina, o que não faz parte do escopo deste trabalho. A Tabela 2 exibe os parâmetros necessários para gerar os resultados. Os parâmetros de *hardware* (total de ciclos para completar a tarefa) são armazenados durante o registro das aplicações. O nível de bateria foi selecionado arbitrariamente entre 15% e 100%, uma vez que é realístico admitir que as aplicações não podem ser executadas quando a bateria for inferior a 15%. A energia gasta na transmissão pelo dispositivo móvel foi definida em 500 mW, com base em Li *et al.* (2018). O tempo de resposta do dispositivo até o nó de névoa é definido entre um intervalo de 2 – 4 ms (JIN; LEE, 2019). Os blocos de dados (Figura 13) utilizados para realizar o processamento foram obtidos das aplicações *healthcare*. Os blocos de dados são os dados de sensoriamento do paciente que as aplicações devem processar para atuar no ambiente de saúde.

Com o propósito de analisar a capacidade do *middleware* proposto, utilizou-se como variável de resposta as métricas:

- **Tempo de resposta**, representa o tempo para finalizar a execução de uma determinada tarefa da aplicação, incluindo o tempo para determinar sobre o *offloading*;

Tabela 2 – Cenário padrão de simulação.

Parâmetro	Valor padrão
Total de ciclos para completar a tarefa T_c , T_c^f e T_c^n	$900 \leq \text{aleatório} \leq 1100$
Pesos de custo de tempo α e energia β	0.5
Energia gasta na transmissão pelo dispositivo γ	Dispositivo móvel: 500 mW
Nível de bateria	$15\% \leq \text{aleatório} \leq 100\%$
Interface sem fio	802.11, 2.4 GHz
Capacidade Computacional - C_d , C^f , e C^n	(@1 GHz, 1GB), (@5 GHz, 16GB), e (@10 GHz, 32GB)
Bloco de dados	8, 16, 32, 64, 128
Tempo de transmissão para nó de névoa	2 - 4 ms
Número de replicações	33
Intervalo de confiança	95%

Figura 13 – Bloco de dados utilizados.

- **Transmissão de dados**, representa o total de mensagens transmitidas entre o dispositivo local e o servidor de névoa; e
- **Consumo de energia**, representa o total de energia consumida pelo dispositivo local.

Os experimentos foram executados 33 vezes, utilizados para determinar a média com um intervalo de confiança de 95% de acordo com a distribuição *t-Student*. Os resultados são apresentados na próxima subseção.

4.3.2 Impacto dos resultados obtidos

A Figura 14 exibe o resultado da energia consumida pelo dispositivo em função da quantidade de dados processados. O intuito é analisar a eficiência energética do descarregamento dinâmico proposto em dispositivos com recursos escassos. Nota-se que com 128 blocos de dados processados (pior caso, Figura 14), o consumo médio de energia nos dispositivos usando o *middleware* proposto foi 1.6 vezes menor quando comparado com a arquitetura HiCH. O mesmo comportamento acontece quando o dispositivo processa 16, 32, e 64 blocos de dados. Quando processados oito blocos de dados (melhor caso, Figura 14), com a nossa solução foi 2.5 vezes melhor que a arquitetura HiCH. Isso ocorre porque o nosso *middleware* não teve a necessidade de sempre realizar a transmissão de dados para o servidor, uma vez que na primeira camada já é

possível realizar o processamento dos dados. Em outras palavras, o processamento é realizado no próprio dispositivo. Portanto, independentemente da quantidade de blocos de dados processados, usando o *middleware* proposto é possível obter um desempenho superior na eficiência energética quando comparado com HiCH.

Figura 14 – Impacto de desempenho da energia consumida

Após analisar a quantidade de energia consumida pelo dispositivo local, investigou-se o tempo de resposta gasto para finalizar a execução dos blocos de dados, como apresentado na Figura 15. Observa-se que, à medida que a quantidade de dados processados aumenta, o *middleware* proposto em nossa arquitetura possui respostas mais rápidas que a arquitetura HiCH, em todos os cenários. Em média, o processamento dos dispositivos ao usar nossa solução foi 25% mais rápido quando comparado com a arquitetura HiCH. O pior caso ocorre quando processados 128 blocos de dados e o melhor caso ocorre quando processados oito blocos de dados, respectivamente, 18% e 48%. Esse ganho no tempo de resposta faz sentido, posto que uma grande parte dos dados foram processados localmente. Assim, o tempo gasto para realizar uma transmissão foi utilizado para processar os dados. Isso auxilia a reduzir o consumo de energia do dispositivo.

Figura 15 – Impacto de desempenho do tempo de resposta

A Figura 16 mostra a quantidade de mensagens transmitidas durante a avaliação do *middleware* proposto. Destaca-se que a quantidade de mensagem transmitida dos dispositivos ao

usar o *middleware* proposto é menor que a arquitetura HiCH, independente da quantidade de dados processados. Com ressalva do primeiro bloco de dados processados, é importante observar que o *middleware* proposto possui um crescimento proporcional das mensagens transmitidas conforme a quantidade de dados processados aumenta. Já o crescimento mais acentuado da arquitetura HiCH está relacionado com o repasse direto das informações para o servidor remoto e por não utilizar nenhum mecanismo para identificar a necessidade de processar os dados localmente ou remotamente.

Figura 16 – Impacto de desempenho do total de transmissão

4.4 Considerações Finais

Este capítulo apresentou uma arquitetura hierárquica e um *middleware* autônomo para o descarregamento dinâmico de dados em ambientes de névoa, denominado LOADING (a middLeware fOr Adaptative Data offloadiNG). A arquitetura proposta baseia-se em *Fog-to-Cloud* (F2C), conceito que define uma arquitetura de névoa por meio de camadas hierárquicas, aumentando o poder de processamento próximo do usuário final. Assim, é possível obter respostas em tempo real e mais efetivas, diminuindo a latência de rede. Já o *middleware* baseia-se na arquitetura de referência para sistemas autônomos, MAPE-K e no paradigma de comunicação *publish-subscribe* para ambientes intermitentes.

A modularização do *middleware* permite uma fácil manutenção, tendo em vista que os módulos são independentes. Por se tratar de um *middleware* genérico, desenvolvedores de software podem construir aplicações sem preocupar em tratar aspectos de adaptação. Isso traz contribuições para o contexto de HSH e de Internet das Coisas, tendo em vista que normalmente são ambientes mutáveis, em que as aplicações necessitam se ajustar aos novos contextos com frequência, alcançando a dependabilidade, isto é, a capacidade de manter a confiança no sistema, mesmo com ocorrência de falhas ou indisponibilidade de recursos. Adicionalmente, uma política para o descarregamento dinâmico de dados foi definida com base no consumo de energia, processamento e tempo de resposta em cada camada da arquitetura F2C. Além da adaptabilidade, foi considerada também a volatilidade de ambientes IoT, onde os dispositivos tendem a perder

---

a conexão com facilidade. Por fim, uma avaliação de desempenho foi realizada utilizando a arquitetura proposta por [Azimi *et al.* \(2017\)](#) como *baseline*.

AVALIAÇÃO DE UM ESTUDO DE CASO NO SENSORIAMENTO DO ESTADO FÍSICO E EMOCIONAL DO USUÁRIO EM AMBIENTES INTELIGENTES

Estima-se um impacto econômico de US\$ 2.7 a US\$ 6.2 trilhões em aplicações de saúde e serviços baseados em IoT, como *m-health*, bem-estar, prevenção, diagnóstico, tratamento e monitoramento (AL-FUQAHA *et al.*, 2015). Dentre as áreas de IoT, cuidados de saúde tem previsão de abranger 41% das aplicações de IoT, causando o maior impacto econômico entre as áreas (AL-FUQAHA *et al.*, 2015). Em razão desse crescimento das tecnologias de IoT, aplicações para a captura e análise de dados dos indivíduos em diferentes contextos, especialmente no contexto residencial e personalizado, são viáveis (HE *et al.*, 2017; MUTLAG *et al.*, 2019). Neste contexto, sistemas de monitoramento de saúde auxiliam principalmente pessoas com deficiência, com mobilidade reduzida (e.g., idosos) (BENIGNO *et al.*, 2014; BANERJEE *et al.*, 2015; LIOUANE *et al.*, 2016; HE *et al.*, 2017). Um exemplo desse tipo de aplicação é a detecção de quedas e a identificação do indivíduo, as quais ajudam na assistência aos cuidados de saúde de um determinado usuário.

Este Capítulo é dividido em duas partes principais:

1. **Um estudo de caso no monitoramento da saúde física e emocional do usuário.** Neste estudo, foram consideradas aplicações interativas para o monitoramento do estado físico e emocional do usuário apresentadas nas seções 5.1 e 5.2, respectivamente. As aplicações utilizadas no monitoramento do estado físico do usuário foram uma aplicação de reconhecimento facial do usuário e uma aplicação de detecção de quedas. Enquanto que para o monitoramento do estado emocional do usuário, foi utilizado uma aplicação de reconhecimento de emoção pela voz.

Figura 17 – Monitoramento de saúde em ambientes IoT

2. **Avaliação qualitativa e quantitativa do impacto das aplicações de saúde com o *middleware* proposto.** Neste estudo, as aplicações de monitoramento de saúde foram utilizadas para validar o *middleware* proposto em um estudo de caso real (Seção 5.3). Além disso, foi realizada uma análise exploratória do *middleware* implementado na arquitetura multicamadas de névoa.

As aplicações de saúde permitem o monitoramento individualizado e não intrusivo do usuário. Estas aplicações foram escolhidas porque apresentam diferentes tipos de dados de entrada, os quais podem demonstrar o comportamento do *middleware* proposto em diferentes cenários de monitoramento, avaliando também a efetividade das respostas em tempo real permitidas pela implementação da arquitetura multicamadas de névoa. A Figura 17 mostra o monitoramento da saúde em ambientes IoT, considerando desde a coleta dos dados pelos dispositivos sensores até a visualização dos resultados. Estes resultados podem ser fornecidos tanto pelo ambiente de névoa local formado de forma distribuída pelo *middleware* proposto, como pelo servidor remoto.

A Figura 18 apresenta como realizamos o monitoramento individualizado e, em seguida, como é executada a tarefa de forma não intrusiva. Para isso, seguimos dois passos: (1) monitoramento da aplicação que consiste na coleta de dados de forma individualizada; e (2) um módulo de aprendizagem que consiste em adquirir conhecimento para detectar as tarefas de aplicação. Este monitoramento abre espaço para investigar métodos para se auto ajustar de acordo com o perfil dos usuários; por exemplo, uma aplicação para chamar uma ambulância quando um idoso cai. Vale destacar que o uso do *middleware* permite várias frentes de monitoramento, uma vez que é projetado para lidar com a interoperabilidade dos dispositivos. Por exemplo, é possível

Figura 18 – Monitoramento remoto individualizado

ter uma aplicação multimodal que receba os dados de voz, imagem e movimento ao mesmo tempo para identificar uma situação de risco. Neste cenário, os dispositivos IoT podem ser intermitentes, porém ainda assim será possível tratar os dados com o *middleware* proposto, uma vez que o módulo de comunicação é baseado no *publish-subscribe* e implementa melhor este tipo de ambiente, comparado com o mecanismo baseado no *Remote Procedure Call* (RPC) e no *Remote Method Invocation* (RMI). Estes possuem uma conexão fortemente acoplada, o que pode ocorrer falhas na aplicação oriundas da desconexão de um dispositivo ou erro na comunicação.

Utilizamos algoritmos de Inteligência Artificial nas aplicações para o reconhecimento do estado físico e emocional do usuário. Neste caso, avaliamos os algoritmos de aprendizagem de máquina Árvore de decisão (HARTMANN *et al.*, 1982), *k-nearest neighbors* (KNN) (SIMOUDIS; AHA, 1997), Naive Bayes (LANGLEY; IBA AND; THOMPSON, 1992) e *Support Vector Machine* (SVM) (VAPNIK, 1995) para a escolha do modelo de aprendizagem.

5.1 Modelo computacional para o sensoriamento do estado físico do usuário

Nesta seção, será apresentada a implementação do estudo de caso para avaliação do *middleware* proposto. Primeiramente, foram criados os modelos das aplicações de reconhecimento facial do usuário e de detecção de quedas para o sensoriamento do estado físico do usuário.

5.1.1 Aplicação de reconhecimento facial do usuário

Nos sistemas de HSH, as imagens são transmitidas por dispositivos e utilizadas por aplicações de reconhecimento para identificar um determinado indivíduo. A face é analisada através da imagem, que determina a identidade de cada usuário. A aplicação de reconhecimento do usuário utilizada nesta pesquisa foi desenvolvida por um aluno de doutorado do mesmo grupo de pesquisa deste discente e publicado em Mano *et al.* (2016). O mapeamento facial proposto

por [Mano et al. \(2016\)](#) é dividido em três etapas: (1) etapas de aquisição de face, (2) extração de traços faciais, e (3) identificação baseada em aprendizagem automática.

De acordo com [Mano et al. \(2016\)](#), o processo de mapeamento facial é realizado considerando 33 pontos obtidos a partir da extração de traços faciais. Os pontos são partes específicas da face que, quando unidas, identificam de forma única um indivíduo. Os pontos são distribuídos da seguinte forma: (a) oito para mapear a boca; (b) seis para cada olho; (c) três para cada sobrancelha; (d) três para o queixo; (e) dois para as narinas; e (f) dois para marcar as extremidades laterais da face próxima aos olhos. Além disso, as distâncias e ângulos são obtidos para todas as combinações possíveis de pontos, para mostrar a linha que liga dois pontos distintos com o eixo horizontal. Consequentemente, cria-se uma representação de dimensionalidade de $D_1 = 2 * 33 + 8 + 2 * 528 = 1130$ atributos. Após obter a dimensionalidade da face, utilizamos esses pontos característicos de cada usuário para formar um método de referência, baseado no classificador de aprendizagem de máquina, para a aprendizagem e identificação do usuário. A Figura 19 apresenta o processo de mapeamento da face para o reconhecimento dos usuários.

Figura 19 – Processo de mapeamento da face para o reconhecimento dos usuários (Adaptado de [Mano et al. \(2016\)](#))

5.1.2 Aplicação de detecção de quedas

Uma aplicação de detecção de queda monitora os movimentos dos usuários em ambientes inteligentes como um meio de monitorar sua rotina à distância e detectar atividade incomum, como quedas. Esta é uma das aplicações críticas de saúde das quais são intolerantes a atraso, uma vez que uma queda pode ser oriunda de um problema grave de saúde (por exemplo, um desmaio de um usuário após sofrer um AVC) e deve ser identificada o mais rápido possível. Isto reflete os ambientes com aplicações de respostas em tempo real. Neste contexto, a implementação do *middleware* proposto na arquitetura multicamadas de névoa é uma solução promissora, pois busca fornecer respostas em tempo real através das camadas de névoa mais próximas do usuário. Assim, nosso sistema coleta e correlaciona dados dos seguintes sensores: (1) o acelerômetro para determinar o deslocamento de um corpo em movimento, e (2) o giroscópio para determinar a rotação e mudança de direção de um corpo em movimento. Nossa proposta de detecção de

quedas é baseada em [Fortino e Gravina \(2015\)](#), no entanto, utilizamos um *smartwatch* para coletar dados de acelerômetros e giroscópios.

O uso de um *smartwatch* apresenta um desafio adicional na coleta de dados, que são os falsos positivos devido a sensibilidade dos movimentos do braço. Para remediar a tarefa não trivial mencionada, utilizamos o *smartwatch* sincronizado com um *smartphone* para coletar os dados. Em outras palavras, esse aplicativo combina simultaneamente dois dispositivos diferentes (um *smartphone* e um *smartwatch*) para monitorar os movimentos do usuário. Assim, suas coordenadas são cruzadas para obter a melhor precisão na decisão de queda ou não. Ao utilizar a combinação dos dados de acelerômetros e giroscópios de ambos os dispositivos, conseguimos reduzir os falsos positivos na identificação de quedas. Em resumo, o aplicativo verifica se há movimento em ambos os dispositivos, verifica se um dispositivo está em contato com o usuário e, em seguida, se ele usa dados para identificar quedas. Assim, os cuidados/assistência de saúde podem ser notificados com maior precisão, de modo que seja possível agir o mais rapidamente possível. Isto é possível por causa da implementação do módulo de comunicação baseado no paradigma *publish-subscribe*, que mesmo com problemas de conexão e falhas na rede é possível garantir a comunicação assíncrona entre os dispositivos sem que ocorra erro na aplicação devido a essas falhas. Diante disso, mesmo que um dos dispositivos falhe, não irá impedir de ter uma inferência sobre a queda. O módulo de comunicação proposto pode receber dados dos dois dispositivos (melhor cenário) ou de apenas um (*smartphone* ou *smartwatch*), sem que o sistema como um todo deixe de operar. A Figura 20 apresenta como pode ser feita a interação do *smartphone* após detectar a queda de um usuário.

Figura 20 – Aplicação para a detecção de queda ([SYSTEM, 2015](#))

5.1.3 Avaliação do modelo para o reconhecimento facial do usuário e detecção de queda

Para ambos os experimentos (i.e., reconhecimento facial do usuário e detecção de queda), o desempenho dos algoritmos foi analisado separadamente usando a técnica de validação cruzada *k-fold cross validation* ([STONE, 1974](#)). Esta é provavelmente a técnica mais popular para avaliar a capacidade de generalização de um modelo a partir de um conjunto de dados. Ela consiste em dividir o conjunto total de dados em *k* subconjuntos independentes do mesmo tamanho, com um

subconjunto usado para testar e o restante $k - 1$ para treinar. Este processo é realizado k vezes, alternando o subconjunto de teste para que todas as combinações sejam satisfeitas, o que nos dá uma estimativa mais precisa. Em nossos experimentos, usamos $k = 10$ na técnica *k-fold cross validation*.

Na avaliação da aplicação de reconhecimento do usuário, utilizou-se imagens de expressões faciais do banco de dados Cohn-Kanade (CK +) (LUCEY; COHN *et al.*, 2010) de acesso livre, que consiste em 593 expressões faciais de 123 atores adultos (69% femininos e 31% masculinos). No entanto, apenas um subconjunto CK + banco de dados foi utilizado para os experimentos, uma vez que existem muitas imagens com poucas diferenças entre elas, pois o tempo de intervalo entre os *frames* são curtos. Isto ocorre devido ao conjunto de dados CK + consistir de uma sequência de imagens obtidas a partir de vídeos. Diante disso, o subconjunto CK + foi utilizado a partir da remoção de imagens com muitas semelhanças para facilitar a criação de um subconjunto equilibrado de imagens relevantes, as quais podem ser capazes de melhorar o desempenho de generalização do classificador de aprendizagem de máquina investigado. Este subconjunto compreende apenas imagens frontais de 30 indivíduos diferentes com 100 imagens cada, totalizando 3000 imagens para os experimentos, tanto para treinar o algoritmo quanto para os testes realizados para identificar o usuário.

A Figura 21 e a Tabela 3 apresentam os resultados das acurácias obtidas pelos algoritmos de aprendizagem de máquina. Podemos visualizar que o algoritmo SVM possui maior acurácia que os demais algoritmos com acurácia média de 79.324%. Diante disso, utilizamos o modelo baseado no algoritmo SVM para avaliar o impacto da aplicação de reconhecimento do usuário no *middleware* proposto.

Figura 21 – Porcentagem de Acurácia das aplicações.

Para avaliação da aplicação de detecção de queda, foi criada uma base de dados baseada em Hsieh *et al.* (2014) e Casilari, Luque e Morón (2015). Assim, os dados do acelerômetro e

Tabela 3 – Média das acurácias obtidas pelos algoritmos para o reconhecimento do usuário

Algoritmo	Acurácia média (%)
Árvore de decisão	70.184
KNN	74.953
Naive Bayes	76.225
SVM	79.324

do giroscópio foram coletados de três indivíduos diferentes utilizando um *smartwatch* (Moto 360) e um *smartphone* (Moto X2). A estratégia de utilização de dados reais foi adotada devido à necessidade de garantir que os dados do acelerômetro e do giroscópio fossem adquiridos ao mesmo tempo. Os dados coletados correspondem a 15 minutos de cada usuário realizando atividades do cotidiano (andando, correndo, em pé, sentado, e deitado) e de quedas. Os dados dos sensores foram coletados em intervalos de tempo de vinte milissegundos. Assim, ao final do experimento, havia um conjunto de 135.000 dados para cada atividade, num total de 810.000 dados. Utilizamos uma sequência de dados com base no desvio padrão dos dados com o objetivo de representar um intervalo de tempo de um segundo.

A metodologia adotada para os experimentos foi a de realizar a média simples de cada vinte e cinco instâncias de dados, equivalente à média dos dados do acelerômetro e do giroscópio em um intervalo de tempo de um segundo.

Os resultados dos algoritmos de aprendizagem de máquina utilizados na classificação de queda podem ser visualizados na Figura 22 e a Tabela 4. Após análise estatística, foi identificado que as acurácias médias dos algoritmos analisados não apresentam diferença estatística. No entanto, é possível destacar que o algoritmo SVM possui menor variância, consequentemente tem mais probabilidade de precisão na classificação da queda. Diante disso, o SVM foi escolhido para avaliar o impacto da aplicação de detecção de queda no *middleware* proposto.

Tabela 4 – Média das acurácias obtidas pelos algoritmos para a detecção de quedas

Algoritmo	Acurácia média (%)
Árvore de decisão	91.225
KNN	91.204
Naive Bayes	90.529
SVM	92.172

5.2 Modelo computacional para o sensoriamento do estado emocional do usuário

Nesta seção, apresentamos o modelo criado para a aplicação de reconhecimento da voz para o sensoriamento do estado emocional do usuário.

Figura 22 – Porcentagem de Acurácia das aplicações.

5.2.1 Aplicação de reconhecimento de emoção pela voz

Esta aplicação foi projetada para coletar dados de um ambiente de monitoramento, processar os sinais de fala e reconhecer as emoções do usuário. Uma casa equipada com dispositivos inteligentes, incluindo *smartphones* e tecnologias vestíveis, pode contribuir para este tipo de sistemas de saúde remotos, uma vez que tais dispositivos podem coletar áudios de fala e tomar decisões em situações críticas (GONÇALVES *et al.*, 2017; MANO *et al.*, 2016). O reconhecimento da emoção dos indivíduos através do discurso é dado como o problema de classificação de aprendizagem supervisionada com múltiplos rótulos. Neste trabalho, os rótulos (classes) são as seis emoções universais descrita em (EKMAN, 1973; EKMAN, 1993), as quais são raiva, desgosto, medo, alegria, tristeza e surpresa, e o estado neutro.

Emoções expressas através da fala podem ser diferenciadas através do sinal de fala e de como o espectro de energia do sinal se comporta. A Figura 23 mostra que emoções cuja expressão vocal é caracterizada por exaltação e gritos tendem a carregar mais energia do que emoções cuja expressão é mais suave. Isto ocorre porque, durante a extração da energia do sinal, o logaritmo $\text{LOGE}(x)$ da soma da potência instantânea do sinal de áudio é usado para cada amostra.

Neste estudo, os sinais extraídos da voz foram *wavelets filter-bank*, *Mel-frequency cepstral coefficients* (MFCCs) (COSTANTINI *et al.*, 2014) incluindo parâmetros de espectro, frequência e energia, e recursos de prosódia. O conjunto de características da prosódia foi composto por *loudness*, *jitter*, *shimmer*, e *pitch*. Além disso, aplicamos a normalização da média e do desvio padrão em todas as características de cada áudio da base de dados. Durante o processo de extração, um parâmetro de memória foi escolhido para o filtro *high-pass* com valor 0,97, baseado em Mano, Vasconcelos e Ueyama (2016). Este valor foi multiplicado pelo fator de atraso unitário ($z - 1$ no domínio Z) no numerador da função de transferência do filtro.

Figura 23 – Espectrogramas representando a frase “A cachoeira faz muito barulho.” para as seis emoções básicas.

Os valores da taxa de duração e sobreposição das janelas aplicadas ao conjunto de características foram respectivamente 40 ms e 1,5 s (EYBEN *et al.*, 2016). Aplicamos a janela de *Hamming* com ganho unitário, que é responsável pela obtenção de *frames* com bordas atenuadas em relação ao centro. A equação 5.1 mostra que o coeficiente de energia logarítmica é obtido pelo cálculo do logaritmo da energia (total) do fragmento de sinal digital $x[n]$.

$$E(x) = \sum_{n=0}^{N-1} x[n]^2 \quad (5.1)$$

A Figura 24 apresenta a aplicação de reconhecimento de emoções pela voz em um experimento realizado pelo nosso grupo de pesquisa. O experimento ocorreu durante um workshop em um cenário de simulação clínica com alunos de enfermagem da Escola de Enfermagem da USP de Ribeirão Preto.

5.2.2 Avaliação do modelo para a aplicação de reconhecimento de emoção pela voz

Um dos principais problemas na criação de modelos com algoritmos de aprendizagem de máquina é o grau de generalização do modelo, uma vez que o modelo pode gerar *overfitting*, dependendo da quantidade de épocas ou da condição de parada. Isto é, o modelo fixa as saídas

Figura 24 – Aplicação de reconhecimento de emoções pela voz em um cenário de simulação clínica.

Fonte: Elaborada pelo autor.

que não estão nos dados de treinamento (REED, 1993; PESSIN, 2013; FILHO, 2018). Diante disso, a avaliação do modelo para a aplicação de reconhecimento de emoção pela voz também foi realizada utilizando a técnica *k-fold cross validation* apresentada na subseção 5.1.3. Esta técnica garante maior precisão na estimativa do classificador e na sua capacidade de generalização (REED, 1993).

Visto que até o momento não se conhecia nenhuma base de dados de voz emocional em língua portuguesa, criamos a base de dados chamada VERBO (Voice Emotion Recognition dataBase in the pOrtuguese language), a qual foi publicada em Neto *et al.* (2018b). A base de dados VERBO foi criada no Instituto de Matemática e Ciências da Computação da Universidade de São Paulo, e está disponível online no site (<https://sites.google.com/view/verbodatabase/>). VERBO é uma base de dados atuada, a qual consiste em gravações de áudio da fala humana feitas por atores, que proferem algumas palavras, frases ou textos pré-definidos, que expressam todas as emoções de cada frase. A grande vantagem do corpus de voz atuada é a sua simplicidade durante a coleta das gravações. Pode-se observar que a atuação segue um roteiro em que o ator está ciente do texto a ser pronunciado e da emoção a ser expressa. Assim, a base de dados pode ser dividida em diretórios separados por ator e/ou emoção.

A base de dados VERBO é composta por áudios foneticamente equilibrados de sete emoções que foram registradas por doze atores brasileiros, entre homens (6) e mulheres (6). Os atores escolhidos para a criação da base de dados VERBO são membros de diferentes grupos de teatro no Brasil e todos têm entre 2 e 23 anos de experiência de atuação profissional. Além disso, todos atuam regularmente, entre 2 vezes por semana (23,1%) e 5 ou mais vezes por semana (46,2%). Embora a formação acadêmica não tenha sido um critério para a escolha dos atores, 83% dos atores têm um diploma de ensino técnico ou superior em Teatro ou Artes. Muitos atores foram contratados para realizar os experimentos, pois nem sempre foi possível encontrar voluntários.

O material linguístico para as gravações foi baseado na base de dados EMOVO (COSTANTINI *et al.*, 2014) e adaptado para o português brasileiro. O material linguístico registrado

Tabela 5 – Sentenças gravadas pelos atores e atrizes

Sentenças gravadas	
1	Os operários levantam cedo.
2	Os bombeiros estão equipados com uma arma.
3	A cachoeira faz muito barulho.
4	No próximo outono, Antônio vai a Minas em quinze de outubro.
5	Agora vou pôr a camiseta e sair para uma caminhada.
6	Um momento depois, ele caminhou ... e tropeçou.
7	Eu queria o número de telefone de seu João.
8	A casa forte quer com pão.
9	A Força está para cima e alho vermelho.
10	O gato está rolando na pêra.
11	Salada de massa pata de carneiro amendoim.
12	Um quarenta e três vinte e sete noventa e cinco mil.
13	Sábado à noite, o que vai fazer?
14	Você vai trazer aquela coisa com você?

consiste em 14 frases, que foram validadas por um profissional linguístico, abrangendo todos os fonemas da língua portuguesa. Todas as sete emoções apresentadas na subseção 5.2.1 foram expressas para cada frase, totalizando 1167 gravações. As sentenças foram criadas fora do contexto de qualquer emoção, ou seja, o valor semântico do conteúdo foi neutro para que os atores e atrizes fossem capaz de colocar suas emoções nas frases. Assim, o significado do texto não influenciou o resultado (por exemplo, a palavra “dor” está relacionada a emoção tristeza ou medo). Para isso, o material linguístico foi composto por frases curtas, longas, sem sentido, e perguntas. Além disso, é possível identificar todos os fonemas da língua portuguesa, contendo vogais e consoantes. O material linguístico das gravações dos áudios são apresentadas na Tabela 5.

A Figura 25 e a Tabela 6 apresentam os resultados das acurácias médias obtidos pelos algoritmos de classificação para o reconhecimento de emoção pela voz. Importante destacar que nesta avaliação, os algoritmos KNN e SVM obtiveram o maior número de acertos na classificação, alcançando uma acurácia média de 76.489 e 75.375%, respectivamente. Porém os dois algoritmos não apresentam diferença estatística entre eles. Neste caso, o classificador KNN, usando a distância euclidiana, foi utilizado na avaliação do impacto da aplicação de reconhecimento de emoção pela voz no *middleware* proposto.

A base VERBO tem um largo segmento de aplicações no mundo real. Deve-se notar que a base de dados da VERBO foi concebida para ser utilizada em sistemas de reconhecimento emocional através da voz. Por exemplo, qualquer sistema de *helpdesk* e *call center* para falantes de língua portuguesa pode se beneficiar da base de dados da VERBO. Isto porque tais sistemas podem coletar e analisar dados de voz das emoções dos clientes e funcionários e como eles reagem a certas situações (HOLMAN; CHISSICK; TOTTERDELL, 2002; DEERY; IVERSON;

Figura 25 – Porcentagem de Acurácia da aplicação de reconhecimento de emoções pela voz.

Tabela 6 – Média das acurácias obtidas pelos algoritmos para o reconhecimento de emoção pela voz

Algoritmo	Acurácia média (%)
Árvore de decisão	66.84
KNN	76.489
Naive Bayes	64.318
SVM	75.375

WALSH, 2002). O reconhecimento de emoções por voz também pode ocorrer usando reuniões presenciais ou via Skype e Hangout, para monitorar os funcionários (JONES; GRAHAM, 2015). Além disso, os usuários podem ser persuadidos por sistemas interativos de acordo com suas emoções (por exemplo, o sistema pode recomendar ouvir uma música para relaxar). O banco de dados da VERBO pode ser adotado em sistemas para detectar distúrbios depressivos unipolares e bipolares e auxiliar psicólogos ou médicos na clínica. Além disso, o tratamento de outras doenças, tais como distúrbios mentais (KOSTOULAS *et al.*, 2012), o stress (C.K. *et al.*, 2017) e Parkinson (ZHAO *et al.*, 2014) também podem ser beneficiados com a base de dados da VERBO. Por fim, é importante salientar que a língua portuguesa é a 6^a língua mais falada a nível mundial (LANE, 2016).

5.3 Avaliação do impacto das aplicações de monitoramento de saúde em ambientes IoT no middleware LOADING

As aplicações de reconhecimento de usuário, detecção de quedas e reconhecimento de emoção pela voz foram usadas para validar a infraestrutura do *middleware* LOADING diante

do sensoramento do estado físico e emocional do usuário. Nesta avaliação, consideramos a arquitetura multicamadas de névoa com o *middleware* *LOADING* presente em duas camadas, definidas como *Fog Layers* na arquitetura proposta: (1) *Fog Layer 1* (névoa - Raspberry Pi) e (2) *Fog Layer 2* (Servidor Remoto). Assim, pode-se inferir em qual camada da arquitetura o processamento da aplicação é válido a partir da quantidade de dados de entrada e do seu custo de processamento. Para esta avaliação, criamos um ambiente distribuído de nós de névoa, denominado *Fog cluster*, onde variamos a quantidade de dispositivos na *Fog Layer 1*, com 1, 3, e 5 dispositivos. O objetivo foi avaliar o módulo de comunicação fornecido pelo *middleware* proposto. Os dados de imagem e voz são coletados usando um *smartphone*, enquanto que os dados de acelerômetro e giroscópio são coletados usando um *smartphone* e um *smartwatch* ao mesmo tempo. As interfaces coletaram os dados e os enviaram para o *Fog cluster* presente na *Fog Layer 1*, no qual foram comparados aos dados anteriores recebidos e criado os novos blocos de dados. Posteriormente, os blocos foram publicados no *MQTT Broker* para servir como entrada para aplicações de sensoramento do estado físico e emocional do usuário (i.e., reconhecimento de usuário, detecção de quedas, e reconhecimento de emoção pela voz). A Tabela 7 apresenta os parâmetros utilizados nos experimentos, os quais foram considerados um arquivo de imagem, um arquivo de coordenadas e um arquivo de áudio como entrada. Vale ressaltar que o arquivo de coordenadas forneceu uma medição de 1 segundo do sensor do acelerômetro e do giroscópio, produzindo um arquivo de 1KB, sendo que cada imagem e áudio tinham em média 32KB e 30KB de tamanho, respectivamente. Vale ressaltar que a implementação do módulo baseado no *publish-subscribe* facilita a comunicação confiável dos dados no ambiente intermitente que é a IoT, e mesmo com a perda de conexão de um dispositivo, o sistema como um todo continua funcionando.

Tabela 7 – Parâmetros de avaliação dos experimentos

	Parâmetros de avaliação									
	Fog Layer 1	Fog Layer 2								
	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10
Número de bloco de dados	1	1	2	4	8	16	32	64	128	256

As avaliações foram realizadas em dispositivos Raspberries Pis na camada *Fog Layer 1* (i.e., dispositivos que suportam a execução das aplicações em termos de consumo de memória e de processamento). Assim, os experimentos foram focados na métrica de tempo de processamento das aplicações em quatro cenários distintos, apresentados na Tabela 8. Os experimentos foram realizados com 33 repetições e 95% de confiança utilizando o *t-Student*.

5.3.1 Avaliação da gerência de recursos

Para medir o impacto das aplicações na infraestrutura, avaliamos duas camadas de névoa da arquitetura de multicamadas hierárquicas com base no desempenho de cada aplicação, as

Tabela 8 – Valores dos parâmetros para os cenários

Parâmetros para os cenários usando <i>offloading</i> de dados	
Cenário 1	1 Servidor Remoto (<i>Fog Layer 2</i>)
Cenário 2	1 Servidor Remoto (<i>Fog Layer 2</i>) e 1 Dispositivo de borda (<i>Fog Layer 1</i>)
Cenário 3	1 Servidor Remoto (<i>Fog Layer 2</i>) e 3 Dispositivos de borda (<i>Fog Layer 1</i>)
Cenário 4	1 Servidor Remoto (<i>Fog Layer 2</i>) e 5 Dispositivos de borda (<i>Fog Layer 1</i>)

quais foram tanto as aplicações no sensoriamento do estado físico quanto do estado emocional do usuário. A avaliação em duas camadas (i.e., *Fog Layer 1* e *Fog Layer 2*) foi suficiente para a avaliação da arquitetura multicamadas com o estudo de caso de saúde, em que as aplicações são intolerantes ao atraso.

No sensoriamento do estado físico do usuário, as Figuras 26 e 27 apresentam as avaliações de desempenho das *Fog Layers 1* e 2 em relação ao tempo de processamento das aplicações interativas de reconhecimento do usuário e de detecção de quedas. O objetivo desta avaliação foi explorar o quão viável é a interação entre as camadas de névoa no processamento de dados. A Figura 26 apresenta o tempo que a aplicação de reconhecimento de usuário levou para ser executada em cada camada de névoa, de acordo com o número de blocos de dados tomados como entrada. Com 1 bloco de dados, a extração de características e classificação na *Fog Layer 1* leva em torno de cinco segundos, enquanto que na *Fog Layer 2*, o tempo é em torno de 0.5 segundos. Entretanto, à medida que o número de blocos de dados aumenta, o tempo de processamento da *Fog Layer 2* se aproxima do tempo de processamento de 1 bloco de dados na *Fog Layer 1*. Por exemplo, com 16 blocos de dados na *Fog Layer 2*, o tempo médio de processamento é de 6.39 segundos. Pode-se observar que em uma grande rajada de dados, o que pode representar um cenário no contexto de IoT em ambientes inteligentes, a *Fog Layer 1* é usada para o processamento, tornando-a menos ociosa enquanto a *Fog Layer 2* processa uma grande quantidade de dados. Diante disso, é possível estabelecer um *trade-off* do processamento entre as duas camadas.

A Figura 27 apresenta o tempo de processamento da aplicação de detecção de queda. Neste cenário, o dispositivo de borda (*Fog Layer 1*) leva 2.2 segundos para detectar uma queda com um bloco de dados. Enquanto que o tempo de resposta desta aplicação executada na *Fog Layer 2* é sete vezes menor. Da mesma forma que a aplicação de reconhecimento do usuário, a aplicação de detecção de queda também atinge o tempo de resposta do dispositivo de borda em um determinado instante, quando a quantidade de dados no ambiente aumenta. A *Fog Layer 2* fornece maior poder computacional ao ambiente de IoT por estar fisicamente mais próximo do usuário, este que pode ser melhorado quando usado em combinação com os dispositivos de borda.

No sensoriamento do estado emocional do usuário, a avaliação do desempenho das *Fog Layers 1* e 2 são apresentadas na Figura 28. É possível observar o mesmo comportamento das aplicações anteriores, em que existe um *trade-off* que determina o equilíbrio do processamento dos dados nas duas camadas *Fog Layers* da arquitetura proposta. Desta forma, é possível melhorar

Figura 26 – Tempo de processamento da aplicação de reconhecimento do usuário

Figura 27 – Tempo de processamento da aplicação de detecção de quedas

o desempenho do sistema e o uso dos recursos disponíveis em um ambiente IoT inteligente de saúde. Assim, é viável usar a abordagem de descarregamento de dados em névoa multicamadas para o compartilhamento e gestão de recursos no processamento de uma grande quantidade de dados (i.e., ambientes inteligentes que refletem o contexto de saúde e IoT).

Comprovando a viabilidade do compartilhamento de recursos em névoa de multicamadas, avaliamos o impacto das aplicações de sensoriamento de saúde na arquitetura de névoa proposta para a *Fog Layer 1*, em relação ao consumo dos recursos de memória e processador. Esta avaliação tem o objetivo de explorar o quanto de recursos os dispositivos podem consumir enquanto executam aplicações de saúde. Vale ressaltar que as aplicações de saúde utilizadas possuem um processamento custoso devido os modelos de Inteligência Artificial, o que reflete ambientes IoT de monitoramento.

Figura 28 – Tempo de processamento da aplicação de reconhecimento de emoções pela voz

As Figuras 29 e 30 apresentam o consumo de memória das aplicações de reconhecimento do usuário e detecção de quedas em ambos os ambientes de desempenho. Podemos observar que na execução da aplicação de reconhecimento do usuário na camada *Fog Layer 1* requer muito mais memória do que na camada *Fog Layer 2*. Isto ocorre devido os recursos de memória de um Raspberry PI serem limitados, porém abre espaço para a investigação de métodos para melhorar o consumo dos recursos dos dispositivos de borda. Para contornar esse problema, propomos uma névoa com mais de um Raspberry PI conectados de forma distribuída em forma de *cluster*, de maneira que o processamento dos dados sejam compartilhados entre eles por um ambiente virtualizado, formando um nó *fog* composto por múltiplos dispositivos (*Fog Layer 1*). Por outro lado, a aplicação de detecção de queda não teve tanto impacto na camada *Fog Layer 1*, uma vez que esta aplicação requer menos recursos para o carregamento e processamento dos dados no algoritmo de detecção.

A Figura 31 apresenta o desempenho da aplicação de reconhecimento de emoções pela voz. Assim como a aplicação de reconhecimento do usuário, a aplicação de reconhecimento de emoções pela voz também possui uma diferença elevada entre a camada *Fog Layer 1* e a camada *Fog Layer 2* no processamento dos dados de voz. Com isso, fica evidente o desafio de prover a gestão de recursos entre os dispositivos de borda e a próxima camada de processamento.

Podemos observar que para todas as aplicações avaliadas, a variação dos dados de entrada não houve diferença significativa para o consumo de memória da *Fog Layer 2*. Em outras palavras, quando temos apenas um bloco de dados o consumo de memória não diferencia tanto quando possuímos 16 bloco de dados.

Avaliamos também o impacto das aplicações de saúde em relação ao consumo do processador durante o sensoriamento do estado físico e emocional do usuário. As Figuras 32 e 33 apresentam o consumo de processamento das aplicações de reconhecimento do usuário e

Figura 29 – Desempenho de consumo de memória da aplicação de reconhecimento de usuário

Figura 30 – Desempenho de consumo de memória da aplicação de detecção de queda

detecção de quedas. Nesta avaliação, o consumo do processador no dispositivo embarcado é alto para a aplicação de reconhecimento do usuário e detecção de queda. Entretanto, é possível visualizar que existe uma alta variância no consumo do processador durante os experimentos. Quando variamos a quantidade de blocos de dados na *Fog Layer 2*, não houve tanta significância para aplicação de reconhecimento de usuário, porém destacamos a variância dos dados durante os experimentos. Por outro lado, houve um aumento de consumo do processador para a detecção de quedas a medida que os bloco de dados aumentavam.

Os resultados mostraram que os dispositivos embarcados suportam tais aplicações e podem ser utilizados em combinação com um servidor de névoa para aumentar o desempenho do sistema como um todo. Isto abre espaço para a investigação de algoritmos de alto desempenho

Figura 31 – Desempenho do consumo de memória da aplicação de reconhecimento de emoções pela voz

que exijam menos recursos dos dispositivos. Assim, o processamento máximo pode ser feito nas bordas da rede, o que pode economizar energia do dispositivo por causa da redução nas transmissões de mensagens para os servidores de névoa e nuvem.

Figura 32 – Desempenho do consumo de processamento da aplicação de reconhecimento do usuário

Fonte: Dados da pesquisa.

O desempenho do consumo do processador da aplicação de reconhecimento de emoções pela voz é apresentado na Figura 34. Nesta aplicação, o consumo do processador na *Fog Layer 1* também é alto em relação ao consumo do processador na *Fog Layer 2*. Porém, da mesma forma que as aplicações de monitoramento do estado físico anteriores, o consumo do processador no dispositivo embarcado houve uma alta variância durante os experimentos. Podemos ver também

Figura 33 – Desempenho do consumo de processamento da aplicação de detecção de queda

Fonte: Dados da pesquisa.

que, não houve tanta variação no consumo do processador quando variamos a quantidade de blocos no servidor de névoa. Apesar da variação e da necessidade de um poder de processamento mais elevado, os dispositivos de borda também são capazes de realizar o processamento dos dados de voz. As informações de variação do consumo do processador durante o sensoriamento auxilia o processo de decisão quando for necessário que o dispositivo faça o descarregamento dos dados para a *Fog Layer 2*.

Figura 34 – Desempenho do consumo de processamento da aplicação de reconhecimento de emoções pela voz

Fonte: Dados da pesquisa.

Portanto, os resultados apresentados mostraram que, além do sensoriamento, os dispositivos de borda também podem auxiliar no processamento dos dados de monitoramento de

saúde. Diante disso, realizamos experimentos para validar o *middleware* LOADING proposto durante o monitoramento do estado físico e emocional dos usuários pelas aplicações de saúde. Nesta avaliação, destacamos o papel *middleware* distribuído na camada *Fog Layer* 1 pelo *fog cluster*, realizando a tomada de decisão do descarregamento de dados antes de transmiti-los para a próxima camada de névoa (i.e., *Fog Layer* 2).

5.3.2 Avaliação da infraestrutura de comunicação

Considerando os cenários apresentados na Tabela 8, foi possível aumentar o desempenho da infraestrutura distribuindo o processamento entre as camadas *Fog Layers* 1 e 2 da arquitetura multicamadas de névoa pelo *middleware* LOADING. Os resultados mostram que o tempo de processamento diminui para todas as aplicações de sensoriamento de saúde quando combinamos as duas *Fog Layers*. Vale destacar a importância do coordenador no escalonamento das tarefas entre os dispositivos.

A Figura 35 apresenta os resultados da decisão de descarregamento de dados, considerando múltiplos dispositivos na *Fog Layer* 1. É possível observar que, quando utilizamos vários dispositivos na *Fog Layer* 1 e distribuimos o processamento entre a camada *Fog Layer* 1 e a camada *Fog Layer* 2, o tempo de processamento diminui em relação a abordagem tradicional (i.e., ao cenário em que os dados são processados apenas na camada *Fog Layer* 2). Isto mostra a importância de um *middleware* capaz de conectar os dispositivos locais em uma única rede, criando um único nó de névoa para prover mais recursos computacionais próximo do usuário final em um ambiente IoT.

Na aplicação de reconhecimento do usuário apresentada na Figura 35a, podemos ver uma diminuição de 1.27 vezes do tempo de processamento da aplicação em relação a abordagem tradicional de névoa, quando utilizamos três dispositivos na borda (*Fog Layer* 1). Quando aumentamos para cinco dispositivos na *Fog Layer* 1, o tempo de processamento da aplicação cai para 1.54 vezes.

Na Figura 35b, o tempo de processamento da aplicação de detecção de queda quando comparamos a *Fog Layer* 1 com a *Fog Layer* 2, é reduzido em até 1.15 vezes usando cinco dispositivos conectados ao *middleware* na camada *Fog Layer* 1. Os resultados mostram que a aplicação de detecção de quedas possui um bom desempenho quando processada somente na *Fog Layer* 2, porém ainda é possível melhorar esse desempenho quando existe a combinação das duas camadas de névoa por meio do *middleware* LOADING. Isto é possível por causa dos dois componentes principais presentes no LOADING: (1) módulo de comunicação e conexão dos dispositivos e (2) módulo de decisão adaptativa de descarregamento dos dados. O módulo de comunicação garante a interação entre os dispositivos de IoT usando o *publish-subscribe*, mesmo com dispositivos intermitentes. Enquanto que a decisão de descarregamento de dados, por meio do *framework* AMOK proposto, é baseada no modelo autônomo de controle de *feedback* da arquitetura MAPE-K.

Similarmente, utilizamos a aplicação de reconhecimento de emoções pela voz para avaliar o desempenho da decisão de descarregamento dos dados em ambientes IoT. Os resultados da Figura 35c mostram que quando aumentamos a quantidade de dispositivos na *Fog Layer 1* para cinco, obtemos uma diminuição de duas vezes em relação a execução somente na *Fog Layer 2*. No mesmo sentido, quando usamos três dispositivos na *Fog Layer 1*, ainda foi possível reduzir 1.43 vezes do tempo de processamento da aplicação. Portanto, a névoa multicamadas desempenha um papel crucial na garantia da eficiência dos sistemas de sensoriamento de saúde em ambientes inteligentes com poucos recursos, provendo respostas em tempo real para as aplicações de saúde. Desta forma, o *middleware* proposto auxilia a arquitetura de névoa multicamadas fornecendo uma comunicação adequada em ambientes IoT e adaptabilidade nas decisões de processamento do ambiente de acordo com o estado atual dos dispositivos.

Figura 35 – Impacto das aplicações no *LOADING* em relação ao descarregamento de dados

O comportamento discutido é semelhante quando observamos o número de mensagens a ser avaliado como uma métrica, como apresenta a Figura 36. Nesta avaliação não foi considerado as transmissões entre os dispositivos, somente as transmissões feitas para o servidor de névoa.

Os resultados mostraram que o mecanismo proposto foi capaz de aumentar a eficiência e o desempenho do processamento de dados no sistema. O mecanismo assegurou escalabilidade, alocando dados para processamento em toda a infraestrutura de névoa. Consequentemente, o número de mensagens enviadas pelas aplicações de sensoriamento do estado físico e emocional do usuário diminuiu quando o mecanismo de decisão de descarregamento de dados foi aplicado.

Figura 36 – Impacto das aplicações no LOADING em relação a quantidade de transmissões

(a) Quantidade de transmissões da aplicação de reconhecimento do usuário

(b) Quantidade de transmissões da aplicação de detecção de queda

(c) Quantidade de transmissões da aplicação de reconhecimento de emoções pela voz

Além disso, o mecanismo de descarregamento reduz o consumo de energia, uma vez que ocorrem menos transmissões no ambiente de IoT. A Figura 37 mostra os resultados da avaliação do consumo de energia das aplicações de saúde. A medida que a quantidade de RASPs na *Fog Layer 1* aumenta, menor é o consumo de energia para o processamento das aplicações. Isto ocorre porque o poder de processamento nas bordas da rede foi aumentado. Desta forma, é possível realizar mais processamento dos dados localmente sem a necessidade de sempre transmitir os dados para o servidor. O consumo de energia da aplicação de detecção de queda não houve diferença significativa porque a maioria das mensagens foram transmitidas para o servidor, uma vez que o *payload* é baixo e o tempo de processamento é curto.

Figura 37 – Impacto das aplicações no LOADING em relação ao consumo de energia

(a) Consumo de energia da aplicação de reconhecimento do usuário

(b) Consumo de energia da aplicação de detecção de queda

(c) Consumo de energia da aplicação de reconhecimento de emoções pela voz

Sendo assim, o *middleware* proposto teve um maior benefício para aplicações que requerem maior poder de processamento, como o reconhecimento de usuários e reconhecimento de emoções pela voz. Isto ocorre pela influência do *payload* dos dados a serem transmitidos pela rede. Ainda assim, o *middleware* proposto provê ganhos de comunicação para aplicações com alto e baixo poder de processamento. Além disso, os dados de entrada que necessitam de maior poder de processamento, tais como imagens e áudios, o descarregamento dos dados na arquitetura de névoa tiveram um maior impacto sobre o desempenho do sistema de saúde como um todo.

5.4 Considerações Finais

Esse trabalho apresentou um estudo de caso no monitoramento da saúde física e emocional do usuário. Este estudo foi utilizado para validar o *middleware* LOADING proposto implementado em uma arquitetura multicamadas de névoa. O *middleware* proposto foi capaz

de aumentar o desempenho no monitoramento por meio da implementação na arquitetura multicamadas em névoa baseada em ambientes IoT. Foi utilizado o *framework* AMOK proposto para a decisão adaptativa do descarregamento de dados na arquitetura de névoa. A decisão de descarregamento foi projetada para considerar as limitações arquitetônicas de seus dispositivos e a arquitetura de névoa. A sincronização da complexidade dos dados através do ambiente atingiu um processamento de dados elevado com baixa latência. Além disso, o *LOADING* teve grandes vantagens quando o processamento foi distribuído por meio de um *fog cluster*, uma vez que reduziu a concorrência de recursos entre as múltiplas aplicações, resultando em baixa latência e menos transmissões na comunicação.

Utilizamos três aplicações de saúde no sensoriamento do estado físico e emocional do usuário para validar e avaliar nosso *middleware*: (1) reconhecimento facial de usuários, (2) detecção de quedas usando uma combinação de sensores (acelerômetro e giroscópio) em um *smartwatch* e *smartphone*, e (3) reconhecimento de emoções pela voz. O *middleware* projetado foi capaz de fornecer recursos e melhorar o sensoriamento da saúde, além de garantir a comunicação assíncrona na interação dos dispositivos com as camadas de névoa e reduzir o tempo de respostas.

CONSIDERAÇÕES FINAIS

Foi apresentada uma arquitetura auto adaptativa e um *middleware* autônomo para a descarga dinâmica de dados em ambientes de névoa multicamadas. A arquitetura proposta baseia-se em *Fog-to-Cloud* (F2C), conceito que define uma arquitetura de névoa por meio de camadas hierárquicas, aumentando o poder de processamento próximo do usuário final. Vale destacar que foi utilizado o conceito de *Indie Fog* para a utilização completa dos recursos do ambiente IoT por meio dos próprios dispositivos do usuário na primeira camada de névoa. Assim, é possível obter respostas em tempo real e mais efetivas, diminuindo a latência de rede. Já o *middleware* baseia-se na arquitetura de referência para sistemas autônomos, MAPE-K e no paradigma de comunicação *publish-subscribe* para ambientes intermitentes.

A modularização do *middleware* permite uma fácil manutenção, tendo em vista que os módulos são independentes. Por se tratar de um *middleware* genérico, desenvolvedores de software podem construir aplicações sem preocupar em tratar aspectos de adaptação. Isso traz contribuições para o contexto de HSH e de Internet das Coisas, tendo em vista que normalmente são ambientes mutáveis, em que as aplicações necessitam se ajustar aos novos contextos com frequência, alcançando a dependabilidade, isto é, a capacidade de manter a confiança no sistema, mesmo com ocorrência de falhas ou indisponibilidade de recursos. Adicionalmente, uma política para o descarregamento dinâmico de dados foi definida com base no consumo de energia, processamento e tempo de resposta em cada camada da arquitetura F2C. Além da adaptabilidade, foi considerada também a volatilidade de ambientes IoT, onde os dispositivos tendem a perder a conexão com facilidade. Além disso, duas avaliações foram feitas para validar a proposta desta tese: (1) uma avaliação de desempenho utilizando a arquitetura proposta por [Azimi et al. \(2017\)](#) como *baseline*, e (2) uma avaliação qualitativa e quantitativa de um estudo de caso no sensoramento do estado físico e emocional do usuário. As aplicações escolhidas para o sensoramento do estado físico foram (1) reconhecimento facial de usuários e (2) detecção de quedas; enquanto que para o sensoramento do estado emocional, foi escolhido a aplicação de (3) reconhecimento de emoções pela voz. Nesta última, fomos além e construímos uma base de

dados emocional de voz atuada por atores em língua portuguesa. Uma das dificuldades na criação desta base foi encontrar atrizes e atores que aceitassem fazer as gravações, voluntariamente ou por um preço acessível.

6.1 Trabalhos Futuros

O desenvolvimento desta tese criou oportunidades e novas ideias surgiram para avançar o estado da arte em arquiteturas de névoa, adaptabilidade e descarga de dados. Outras pesquisas podem ser norteadas a partir das contribuições presentes nesta tese. A seguir são elencadas pesquisas que podem ser feitas futuramente:

- implementar nossa arquitetura proposta em dispositivos de ponta do mundo real para que apresente a colaboração real de múltiplas camadas de névoa em outros contextos de sistemas críticos e tempo real.
- novos modelos matemáticos podem ser criados para permitir a avaliação do impacto de algumas outras características peculiares da descarga de dados, considerando a influência de outros fatores na decisão de descarga dinâmica.
- algoritmos de aprendizado de máquina podem ser propostos para que os dispositivos aprendam quando realizar ou não a descarga dos dados. Neste sentido, vale a pena explorar métodos de aprendizado por reforço de forma distribuída e colaborativa para que o dispositivo seja capaz de criar um padrão do ambiente sem a necessidade de um coordenador.
- implementar um modelo tolerante a falha que possa prevê quando um dispositivo irá se desconectar, podendo assim realizar uma ação de prevenção do sistema, como atualizar um banco de dados no servidor com informações coletadas e não transmitidas.
- criar mecanismos de segurança para garantir a privacidade dos usuários. Neste cenário, o uso do conceito de *Indie fog* pode ser explorado por meio de uma rede *blockchain* se conectando com servidores multicamadas de névoa.
- explorar a adaptabilidade da solução proposta diante de diferentes cenários de névoa, como por exemplo a migração de VMs em redes oportunísticas e mobilidade de usuários em outros contextos de IoT.

6.2 Principais publicações

Nesta seção, serão apresentadas as publicações que resultaram diretamente desta pesquisa de doutorado, bem como as colaborações que este discente teve com outros trabalhos correlacionados:

6.2.1 Publicações científicas geradas desta tese

- **Torres Neto, J. R.** ; Giuntini, F. T. ; Rocha, G. P. ; Ueyama, J. . “LOADING: Um middleware autônomo baseado em ciclos de feedback para o *offloading* dinâmico de dados em *healthcare*”. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 2020. (**Submetido**)
- **Torres Neto, J. R.**, Geraldo P. Rocha Filho, Leandro Y. Mano, Leandro A. Villas, and Jó Ueyama. “Exploiting Offloading in IoT-Based Microfog: Experiments with Face Recognition and Fall Detection”. *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 2786837, 13 pages, 2019.
- **Torres Neto, J. R.**; Filho, Geraldo P. R.; Mano, Leandro Y.; Ueyama, Jó. “VERBO: voice emotion recognition database in portuguese language”. *Journal of Computer Science*, Deira, Dubai, United Arab Emirates, Science Publications, v. 14, n. 11, p. 1420-1430, 2018.
- **Torres Neto, J. R.** ; Rocha, G. P. ; Mano, L. Y. ; Ueyama, J. “INCA: Um sistema healthcare flexível baseado no paradigma fog computing e publish/subscribe”. In: Workshop de Computação Urbana (CoUrb), 2017, Belém, PA. I Workshop de Computação Urbana (CoUrb), 2017.
- **Torres Neto, J. R.**, Mazzo, A., Pereira Júnior, G. A. e Ueyama, J. “Modelo Computacional para o reconhecimento de emoções baseado na voz de alunos nos cenários de simulação clínica”. In: *3º Congresso de Graduação da Universidade de São Paulo*, São Paulo, BRA, 2017.

6.2.2 Publicações científicas complementares

- Gonçalves, Vinícius; Rocha Filho, Geraldo; Santis, Ligia; Faccioli, Juliana; **Torres Neto**, José; et al. “FlexPersonas - Flexible Design of IoT-based Home Healthcare Systems Targeted at the Older Adults”. In: *Transactions on Accessible Computing*. (**Submetido**)
- Mano, L. Y.; Mazzo, A; **Torres Neto, J. R.**, Meska, M.; Giancristofaro, G. T.; Ueyama, J.; Júnior, G. A. “Using emotion recognition to assess simulation-based learning”. *Nurse Education in Practice*. Elsevier, 2019. Volume 36, 2019, Pages 13-19, ISSN 1471-5953.
- Mano, Leandro; Mazzo, Alessandra; **Torres Neto**, José; et al. “The Relation of Satisfaction, Self-Confidence and Emotion in a Simulated Environment” *International Journal of Nursing Education Scholarship*, 16.1 (2019)
- Sawada, Luana Okino; Mano, Leandro; **Torres Neto**, José. “A module-based framework to emotion recognition by speech: a case study in clinical simulation”. In: *Journal of Ambient Intelligence and Humanized Computing*, 2019.

- Filho, Geraldo P. R.; **Torres Neto, José R.**; Valejo, Alan; Meneguette, Rodolfo I.; Villas, Leandro A.; Ueyama, Jó. “Um Sistema de Controle Neuro-Fog para Infraestruturas Residenciais via Objetos Inteligentes”. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 36. , 2018, Campos do Jordão, Porto Alegre.
- Sousa, Adriano B. de; **Torres Neto, José R.**; Filho, Geraldo P. R.; Ueyama, Jó. “Uma plataforma de IoT para integração de dispositivos baseada em nuvem com Apache Kafka”. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 1. , 2018, Campos do Jordão, Porto Alegre.
- Rocha, G. P. ; **Torres Neto, J. R.** ; Ueyama, J. . “Uma Solução para Tomadas de Decisões em um Ambiente Residencial baseado em fog computing e inteligência computacional”. In: Workshop de Computação Urbana (CoUrb), 2017, Belém, PA. I Workshop de Computação Urbana (CoUrb), 2017.
- Mano, L., Funes, M., Volpato, T., **Torres Neto, J. R.**. “Explorando tecnologias de IoT no contexto de Health Smart Home: uma abordagem para detecção de quedas em pessoas idosas”. Journal On Advances In Theoretical And Applied Informatics, 2(1), 46-57, 2016.

6.2.3 *Produção técnica*

- **Patente:** Ueyama, J.; Vieira, H. F.; **Torres Neto, J. R.**; Mano, L. Y. “Sistema de Monitoramento, Dispositivo de Aquisição e Transmissão de Dados e Método de Identificação e Classificação de Aspectos Físicos e Emocionais”. Número do Registro: BR102018016161-0, Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial, 2018.

REFERÊNCIAS

AAZAM, M.; ZEADALLY, S.; HARRAS, K. A. Fog computing architecture, evaluation, and future research directions. **IEEE Communications Magazine**, v. 56, n. 5, p. 46–52, May 2018. ISSN 1558-1896. Citado nas páginas 35 e 36.

_____. Offloading in fog computing for iot: Review, enabling technologies, and research opportunities. **Future Generation Computer Systems**, v. 87, p. 278 – 289, 2018. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X18301973>>. Citado na página 24.

AHLSÉN, M.; ASANIN, S.; KOOL, P.; ROSENGREN, P.; THESTRUP, J. Service-oriented middleware architecture for mobile personal health monitoring. **Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering**, v. 83 LNICST, p. 305–312, 2012. Cited By 5. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84869598660&doi=10.1007%2f978-3-642-29734-2_42&partnerID=40&md5=bcb201eb1ffdf738336e7353739e4a14>. Citado na página 48.

AIELLO, F.; FORTINO, G.; GALZARANO, S.; VITTORIOSO, A. Tinymaps: A lightweight java-based mobile agent system for wireless sensor networks. In: _____. **Intelligent Distributed Computing V: Proceedings of the 5th International Symposium on Intelligent Distributed Computing – IDC 2011, Delft, The Netherlands – October 2011**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 161–170. ISBN 978-3-642-24013-3. Disponível em: <http://dx.doi.org/10.1007/978-3-642-24013-3_16>. Citado na página 48.

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys Tutorials**, v. 17, n. 4, p. 2347–2376, Fourthquarter 2015. ISSN 1553-877X. Citado nas páginas 23, 28, 45 e 75.

ALI, S.; GHAZAL, M. Real-time heart attack mobile detection service (rhamds): An iot use case for software defined networks. In: **2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 26.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. Citado nas páginas 31 e 32.

AUGUSTO, J.; KRAMER, D.; ALEGRE, U.; COVACI, A.; SANTOKHEE, A. The user-centred intelligent environments development process as a guide to co-create smart technology for people with special needs. **Univers. Access Inf. Soc.**, Springer-Verlag, Berlin, Heidelberg, v. 17, n. 1, p. 115–130, mar. 2018. ISSN 1615-5289. Disponível em: <<https://doi.org/10.1007/s10209-016-0514-8>>. Citado na página 46.

AZIMI, I.; ANZANPOUR, A.; RAHMANI, A. M.; PAHIKKALA, T.; LEVORATO, M.; LILJEBERG, P.; DUTT, N. Hich: Hierarchical fog-assisted computing architecture for healthcare iot. **ACM Trans. Embed. Comput. Syst.**, ACM, New York, NY, USA, v. 16, n. 5s, p. 174:1–174:20,

set. 2017. ISSN 1539-9087. Disponível em: <<http://doi.acm.org/10.1145/3126501>>. Citado nas páginas 24, 25, 27, 51, 52, 53, 56, 68, 69, 73 e 99.

BANERJEE, T.; KELLER, J. M.; POPESCU, M.; SKUBIC, M. Recognizing complex instrumental activities of daily living using scene information and fuzzy logic. **Computer Vision and Image Understanding**, 2015. Citado nas páginas 23, 46 e 75.

BARTH, P.; GROSS, H. Blaubot—hassle-free multi-device applications. In: SPRINGER. **International Conference on Mobile Web and Information Systems**. [S.l.], 2017. p. 209–223. Citado na página 66.

BENIGNO, R.; PEDROSA, N.; TOMÉ, P.; ARAÚJO, W.; NETO, J. R. T.; JÚNIOR, E.; MOTA, S.; FILHO, J. B.; MARTIN, H. Annotation-based access control approach for ubiquitous systems. In: **2014 IEEE 28th International Conference on Advanced Information Networking and Applications**. [S.l.: s.n.], 2014. p. 218–225. ISSN 1550-445X. Citado nas páginas 23, 46 e 75.

BONOMI, F.; MILITO, R.; NATARAJAN, P.; ZHU, J. Fog computing: A platform for internet of things and analytics. In: _____. **Big Data and Internet of Things: A Roadmap for Smart Environments**. Cham: Springer International Publishing, 2014. p. 169–186. ISBN 978-3-319-05029-4. Disponível em: <http://dx.doi.org/10.1007/978-3-319-05029-4_7>. Citado na página 66.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: **Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing**. New York, NY, USA: ACM, 2012. (MCC '12), p. 13–16. ISBN 978-1-4503-1519-7. Disponível em: <<http://doi.acm.org/10.1145/2342509.2342513>>. Citado nas páginas 35 e 42.

BOTTA, A.; DONATO, W. D.; PERSICO, V.; PESCAPÉ, A. On the integration of cloud computing and internet of things. In: IEEE. **Future Internet of Things and Cloud (FiCloud), 2014 International Conference on**. [S.l.], 2014. p. 23–30. Citado na página 35.

BRUN, Y.; SERUGENDO, G. M.; GACEK, C.; GIESE, H.; KIENLE, H.; LITOIU, M.; MÜLLER, H.; PEZZÈ, M.; SHAW, M. Software engineering for self-adaptive systems. In: CHENG, B. H.; LEMOS, R.; GIESE, H.; INVERARDI, P.; MAGEE, J. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2009. cap. Engineering Self-Adaptive Systems Through Feedback Loops, p. 48–70. ISBN 978-3-642-02160-2. Citado nas páginas 15, 39, 40, 41 e 42.

CASILARI, E.; LUQUE, R.; MORÓN, M.-J. Analysis of android device-based solutions for fall detection. **Sensors**, v. 15, n. 8, p. 17827–17894, 2015. ISSN 1424-8220. Disponível em: <<http://www.mdpi.com/1424-8220/15/8/17827>>. Citado na página 80.

CAVALCANTE, E.; PEREIRA, J.; ALVES, M. P.; MAIA, P.; MOURA, R.; BATISTA, T.; DELICATO, F. C.; PIRES, P. F. On the interplay of internet of things and cloud computing: A systematic mapping study. **Computer Communications**, Elsevier, v. 89, p. 17–33, 2016. Citado nas páginas 34 e 35.

CERINA, L.; NOTARGIACOMO, S.; PACCANIT, M. G.; SANTAMBROGIO, M. D. A fog-computing architecture for preventive healthcare and assisted living in smart ambients. In: **2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 26.

CHANG, C.; SRIRAMA, S. N.; BUYYA, R. Indie fog: An efficient fog-computing infrastructure for the internet of things. **Computer**, v. 50, n. 9, p. 92–98, 2017. ISSN 1558-0814. Citado na página 25.

CHENG, B. H.; LEMOS, R. de; GIESE, H.; INVERARDI, P.; MAGEE, J.; ANDERSSON, J.; BECKER, B.; BENCOMO, N.; BRUN, Y.; CUKIC, B.; SERUGENDO, G. D. M.; DUSTDAR, S.; FINKELSTEIN, A.; GACEK, C.; GEIHS, K.; GRASSI, V.; KARSAI, G.; KIENLE, H.; KRAMER, J.; LITOIU, M.; MALEK, S.; MIRANDOLA, R.; MLLER, H.; PARK, S.; SHAW, M.; TICHY, M.; TIVOLI, M.; WEYNS, D.; WHITTLE, J. Software engineering for self-adaptive systems: A research roadmap. In: CHENG, B. H.; LEMOS, R.; GIESE, H.; INVERARDI, P.; MAGEE, J. (Ed.). **Software Engineering for Self-Adaptive Systems**. [S.l.]: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5525). p. 1–26. ISBN 978-3-642-02160-2. Citado na página 25.

CHIANG, M.; HA, S.; I, C.; RISSO, F.; ZHANG, T. Clarifying fog computing and networking: 10 questions and answers. **IEEE Communications Magazine**, v. 55, n. 4, p. 18–20, April 2017. ISSN 1558-1896. Citado na página 36.

C.K., Y.; HARIHARAN, M.; NGADIRAN, R.; ADOM, A. H.; YAACOB, S.; BERKAI, C.; POLAT, K. A new hybrid pso assisted biogeography-based optimization for emotion and stress recognition from speech signal. **Expert Systems with Applications**, v. 69, p. 149 – 158, 2017. ISSN 0957-4174. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417416305759>>. Citado na página 86.

COSTANTINI, G.; IADEROLA, I.; PAOLONI andrea; TODISCO, M. Emovo corpus: an italian emotional speech database. In: CHAIR), N. C. C.; CHOUKRI, K.; DECLERCK, T.; LOFTSSON, H.; MAEGAARD, B.; MARIANI, J.; MORENO, A.; ODIJK, J.; PIPERIDIS, S. (Ed.). **Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)**. Reykjavik, Iceland: European Language Resources Association (ELRA), 2014. ISBN 978-2-9517408-8-4. Citado nas páginas 82 e 84.

CROATTI, A.; MONTAGNA, S.; RICCI, A.; GAMBERINI, E.; ALBARELLO, V.; AGNOLETTI, V. Bdi personal medical assistant agents: The case of trauma tracking and alerting. **Artificial Intelligence in Medicine**, v. 96, p. 187 – 197, 2019. ISSN 0933-3657. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0933365717306000>>. Citado na página 24.

CRUZ, M. A. A. da; RODRIGUES, J. J. P. C.; AL-MUHTADI, J.; KOROTAEV, V. V.; ALBUQUERQUE, V. H. C. de. A reference model for internet of things middleware. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 871–883, April 2018. ISSN 2372-2541. Citado na página 58.

CUERVO, E.; BALASUBRAMANIAN, A.; CHO, D.-k.; WOLMAN, A.; SAROIU, S.; CHANDRA, R.; BAHL, P. Maui: Making smartphones last longer with code offload. In: **Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services**. New York, NY, USA: ACM, 2010. (MobiSys '10), p. 49–62. ISBN 978-1-60558-985-5. Disponível em: <<http://doi.acm.org/10.1145/1814433.1814441>>. Citado na página 43.

DANEELS, G.; MUNICIO, E.; SPAEY, K.; VANDEWIELE, G.; DEJONGHE, A.; ONGENAE, F.; LATRÉ, S.; FAMAÉY, J. Real-time data dissemination and analytics platform for challenging iot environments. In: **2017 Global Information Infrastructure and Networking Symposium (GIIS)**. [S.l.: s.n.], 2017. p. 23–30. ISSN 2150-329X. Citado nas páginas 49 e 52.

- DEERY, S.; IVERSON, R.; WALSH, J. Work relationships in telephone call centres: Understanding emotional exhaustion and employee withdrawal. **Journal of Management studies**, Wiley Online Library, v. 39, n. 4, p. 471–496, 2002. Citado nas páginas 85 e 86.
- EKMAN, P. **Cross-cultural studies of facial expression**. [S.l.]: New York: Academic, 1973. Citado na página 82.
- _____. Facial expression and emotion. **American psychologist**, American Psychological Association, v. 48, n. 4, p. 384, 1993. Citado na página 82.
- ELKHATIB, Y.; PORTER, B.; RIBEIRO, H. B.; ZHANI, M. F.; QADIR, J.; RIVIÈRE, E. On using micro-clouds to deliver the fog. **IEEE Internet Computing**, v. 21, n. 2, p. 8–15, Mar 2017. ISSN 1089-7801. Citado na página 48.
- ESCAMILLA-AMBROSIO, P. J.; RODRÍGUEZ-MOTA, A.; AGUIRRE-ANAYA, E.; ACOSTA-BERMEJO, R.; SALINAS-ROSALES, M. Distributing computing in the internet of things: Cloud, fog and edge computing overview. In: _____. **NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico**. Cham: Springer International Publishing, 2018. p. 87–115. ISBN 978-3-319-64063-1. Disponível em: <https://doi.org/10.1007/978-3-319-64063-1_4>. Citado na página 26.
- ETEMAD, M.; AAZAM, M.; ST-HILAIRE, M. Using devs for modeling and simulating a fog computing environment. In: **2017 International Conference on Computing, Networking and Communications (ICNC)**. [S.l.: s.n.], 2017. p. 849–854. Citado na página 25.
- EUGSTER, P. T.; FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A.-M. The many faces of publish/subscribe. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 35, n. 2, p. 114–131, jun. 2003. ISSN 0360-0300. Disponível em: <<http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/857076.857078>>. Citado nas páginas 28, 29 e 67.
- EYBEN, F.; SCHERER, K. R.; SCHULLER, B. W.; SUNDBERG, J.; ANDRÉ, E.; BUSSO, C.; DEVILLERS, L. Y.; EPPS, J.; LAUKKA, P.; NARAYANAN, S. S.; TRUONG, K. P. The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing. **IEEE Transactions on Affective Computing**, v. 7, n. 2, p. 190–202, April 2016. ISSN 1949-3045. Citado na página 83.
- FERSI, G. Middleware for internet of things: A study. In: **2015 International Conference on Distributed Computing in Sensor Systems**. [S.l.: s.n.], 2015. p. 230–235. ISSN 2325-2936. Citado nas páginas 58, 66 e 67.
- FIELDING, R. T.; TAYLOR, R. N. **Architectural styles and the design of network-based software architectures**. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. Citado na página 43.
- FILHO, G.; UEYAMA, J.; FAICAL, B.; GUIDONI, D.; VILLAS, L. Residi: An intelligent decision platform for residential infrastructure using wireless sensors and actuators. In: **Computer Networks and Distributed Systems (SBRC), 2015 XXXIII Brazilian Symposium on**. [S.l.: s.n.], 2015. p. 237–245. Citado na página 67.
- FILHO, G. P. R. **Soluções de tomadas de decisões inteligentes para infraestruturas residenciais**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação - Universidade de

São Paulo, São Carlos, Disponível em: <<https://www.teses.usp.br/teses/disponiveis/55/55134/tde-18102018-104924/>>, 2018. Citado na página 84.

FILHO, G. P. R.; MENEGUETTE, R. I.; MAIA, G.; PESSIN, G.; GONÇALVES, V. P.; WEI-GANG, L.; UYAMA, J.; VILLAS, L. A. A fog-enabled smart home solution for decision-making using smart objects. **Future Generation Computer Systems**, v. 103, p. 18 – 27, 2020. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X18322301>>. Citado na página 24.

FILHO, G. P. R.; UYAMA, J.; FAICAL, B. S.; PESSIN, G.; FARIAS, C. M. d.; PAZZI, R. W.; GUIDONI, D. L.; VILLAS, L. A. An energy-aware system for decision-making in a residential infrastructure using wireless sensors and actuators. In: **Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on**. [S.l.: s.n.], 2015. p. 9–16. Citado na página 68.

FOK, C.-L.; ROMAN, G.-C.; LU, C. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. **ACM Transactions on Autonomous and Adaptive Systems (TAAS)**, ACM, v. 4, n. 3, p. 16, 2009. Citado na página 50.

FORTINO, G.; GRAVINA, R. Fall-mobileguard: A smart real-time fall detection system. In: ICST (INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND ... **Proceedings of the 10th EAI International Conference on Body Area Networks**. [S.l.], 2015. p. 44–50. Citado na página 79.

FORTINO, G.; GUERRIERI, A.; RUSSO, W.; SAVAGLIO, C. Integration of agent-based and cloud computing for the smart objects-oriented iot. In: . [s.n.], 2014. p. 493–498. Cited By 42. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-84904664101&doi=10.1109%2fCSCWD.2014.6846894&partnerID=40&md5=4886139ea75226f26c840cde94f28043>>. Citado nas páginas 45 e 52.

GARLAN, D.; CHENG, S.-W.; HUANG, A.-C.; SCHMERL, B.; STEENKISTE, P. Rainbow: architecture-based self-adaptation with reusable infrastructure. **Computer**, v. 37, n. 10, p. 46–54, Oct 2004. Citado nas páginas 25 e 40.

GIA, T. N.; JIANG, M.; RAHMANI, A.; WESTERLUND, T.; LILJEBERG, P.; TENHUNEN, H. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In: **2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing**. [S.l.: s.n.], 2015. p. 356–363. ISSN null. Citado nas páginas 51 e 52.

GIUNTINI, F. T.; BEDER, D. M.; UYAMA, J. Exploiting self-organization and fault tolerance in wireless sensor networks: A case study on wildfire detection application. **International Journal of Distributed Sensor Networks**, v. 13, n. 4, p. 1550147717704120, 2017. Citado nas páginas 25 e 27.

GOMES, B.; MUNIZ, L.; SILVA, F. J. da Silva e; RÍOS, L. E. T.; ENDLER, M. A comprehensive cloud-based iot software infrastructure for ambient assisted living. In: **2015 International Conference on Cloud Technologies and Applications (CloudTech)**. [S.l.: s.n.], 2015. p. 1–8. Citado nas páginas 48 e 52.

GONÇALVES, V. P.; COSTA, E. P.; VALEJO, A.; FILHO, G. P.; JOHNSON, T. M.; PESSIN, G.; UEYAMA, J. Enhancing intelligence in multimodal emotion assessments. **Applied Intelligence**, Kluwer Academic Publishers, Hingham, MA, USA, v. 46, n. 2, p. 470–486, mar. 2017. ISSN 0924-669X. Disponível em: <<https://doi.org/10.1007/s10489-016-0842-7>>. Citado na página 82.

GONÇALVES, V. P.; NERIS, V. P.; UEYAMA, J.; SERAPHINI, S.; DIAS, T. C.; FILHO, G. R. Senior citizens in interaction with mobile phones: A flexible middleware approach to support the diversity. **Proc. of the SERP 2013**, v. 1, p. 56–62, 2013. Citado na página 46.

GRAPHQL. 2019. <<http://facebook.github.io/graphql/October2016/>>. Accessed: 2019-01-06. Citado na página 43.

GROUP, O. C. A. W. *et al.* Openfog reference architecture for fog computing. **OPFRA001**, v. 20817, p. 162, 2017. Citado nas páginas 15, 56 e 57.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645 – 1660, 2013. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X13000241>>. Citado nas páginas 31 e 32.

HABAK, K.; SHI, C.; ZEGURA, E. W.; HARRAS, K. A.; AMMAR, M. H. Elastic mobile device clouds: Leveraging mobile devices to provide cloud computing services at the edge. In: . [S.l.: s.n.], 2017. Citado na página 35.

HARTMANN, C.; VARSHNEY, P.; MEHROTRA, K.; GERBERICH, C. Application of information theory to the construction of efficient decision trees. **IEEE Transactions on Information Theory**, v. 28, n. 4, p. 565–577, July 1982. ISSN 1557-9654. Citado na página 77.

HE, S.; CHENG, B.; WANG, H.; HUANG, Y.; CHEN, J. Proactive personalized services through fog-cloud computing in large-scale iot-based healthcare application. **China Communications**, v. 14, n. 11, p. 1–16, Nov 2017. Citado nas páginas 26 e 75.

HOLMAN, D.; CHISSICK, C.; TOTTERDELL, P. The effects of performance monitoring on emotional labor and well-being in call centers. **Motivation and Emotion**, Springer, v. 26, n. 1, p. 57–81, 2002. Citado nas páginas 85 e 86.

HONG, K.; LILLETHUN, D.; RAMACHANDRAN, U.; OTTENWÄLDER, B.; KOLDEHOFE, B. Mobile fog: a programming model for large-scale applications on the internet of things. In: **MCC@SIGCOMM**. [S.l.: s.n.], 2013. Citado na página 36.

HORN, P. **Autonomic computing: IBM's Perspective on the State of Information Technology**. [S.l.: IBM, 2001. <http://people.scs.carleton.ca/~soma/biosesec/readings/autonomic_computing.pdf>. Citado na página 37.

HSIEH, S. L.; CHEN, C. C.; WU, S. H.; YUE, T. W. A wrist -worn fall detection system using accelerometers and gyroscopes. In: **Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control**. [S.l.: s.n.], 2014. p. 518–523. Citado na página 80.

HUANG, L.; FENG, X.; ZHANG, C.; QIAN, L.; WU, Y. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. **Digital Communications and Networks**, v. 5, n. 1, p. 10 – 17, 2019. ISSN 2352-8648. Artificial Intelligence for Future Wireless Communications and Networking. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2352864818301469>>. Citado nas páginas 44, 63 e 65.

HUEBSCHER, M. C.; MCCANN, J. A. A survey of autonomic computing—degrees, models, and applications. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 40, n. 3, p. 7:1–7:28, ago. 2008. ISSN 0360-0300. Citado nas páginas 37 e 38.

IBM. **The Tivoli software implementation of autonomic computing guidelines**. 2002. <ftp://ftp.software.ibm.com/software/tivoli/brochures/br-autonomic-guide.pdf>. Citado na página 38.

_____. **An architectural blueprint for autonomic computing**. USA: IBM, 2005. <<http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf>>. Citado nas páginas 15, 28, 40, 41 e 60.

JIN, Y.; LEE, H. On-demand computation offloading architecture in fog networks. **Electronics**, v. 8, n. 10, 2019. ISSN 2079-9292. Disponível em: <<https://www.mdpi.com/2079-9292/8/10/1076>>. Citado nas páginas 57, 65 e 69.

JONES, N. B.; GRAHAM, C. M. Virtual teams in business and distance education: Reflections from an mba class. **Journal of Business & Economic Policy**, v. 2, n. 1, p. 49–59, 2015. Citado na página 86.

KANG, B.; KIM, D.; CHOO, H. Internet of everything: A large-scale autonomic iot gateway. **IEEE Transactions on Multi-Scale Computing Systems**, v. 3, n. 3, p. 206–214, July 2017. ISSN 2332-7766. Citado nas páginas 49 e 52.

KEPHART, J. O.; CHESS, D. The vision of autonomic computing. **Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 36, n. 1, p. 41–50, jan. 2003. ISSN 0018-9162. Citado nas páginas 37 e 40.

KHAN, M.; DIN, S.; JABBAR, S.; GOHAR, M.; GHAYVAT, H.; MUKHOPADHYAY, S. Context-aware low power intelligent smarthome based on the internet of things. **Computers Electrical Engineering**, v. 52, n. Supplement C, p. 208 – 222, 2016. ISSN 0045-7906. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0045790616300933>>. Citado nas páginas 45 e 52.

KOSTOULAS, T.; MPORAS, I.; KOCSIS, O.; GANCHEV, T.; KATSAOUNOS, N.; SANTAMARIA, J. J.; JIMENEZ-MURCIA, S.; FERNANDEZ-ARANDA, F.; FAKOTAKIS, N. Affective speech interface in serious games for supporting therapy of mental disorders. **Expert Systems with Applications**, v. 39, n. 12, p. 11072 – 11079, 2012. ISSN 0957-4174. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417412005908>>. Citado na página 86.

KUMAR, K.; LIU, J.; LU, Y.-H.; BHARGAVA, B. K. A survey of computation offloading for mobile systems. **MONET**, v. 18, n. 1, p. 129–140, 2013. Disponível em: <<http://dblp.uni-trier.de/db/journals/monet/monet18.html#KumarLLB13>>. Citado na página 42.

LA, Q.; NGO, M.; DINH, T.; QUEK, T.; SHIN, H. Enabling intelligence in fog computing to achieve energy and latency reduction. **Digital Communications and Networks**, v. 5, n. 1, p. 3–9, 2019. Cited By 4. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85056659319&doi=10.1016%2fj.dcan.2018.10.008&partnerID=40&md5=b849a734ed0ae16dbabe2e6f7f46d4a8>>. Citado nas páginas 51 e 52.

- LANE, J. **The 10 Most Spoken Languages In The World**. 2016. <<https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world>>. Citado na página 86.
- LANGLEY, P.; IBA AND, W.; THOMPSON, K. An analysis of bayesian classifiers. In: **Proceedings of the Tenth National Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 1992. (AAAI'92), p. 223–228. ISBN 0262510634. Citado na página 77.
- LE, X. H. B.; MASCOLO, M. D.; GOUIN, A.; NOURY, N. Health smart home for elders - a tool for automatic recognition of activities of daily living. In: **2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society**. [S.l.: s.n.], 2008. p. 3316–3319. ISSN 1094-687X. Citado na página 46.
- LI, J.; GAO, H.; LV, T.; LU, Y. Deep reinforcement learning based computation offloading and resource allocation for mec. In: **2018 IEEE Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2018. p. 1–6. ISSN 1558-2612. Citado nas páginas 44, 63, 64, 65 e 69.
- LINGARAJ, K.; BIRADAR, R. V.; PATIL, V. Eagilla: An enhanced mobile agent middleware for wireless sensor networks. **Alexandria engineering journal**, Elsevier, v. 57, n. 3, p. 1197–1204, 2018. Citado na página 50.
- LIOUANE, Z.; LEMLOUMA, T.; ROOSE, P.; WEIS, F.; MESSAOUD, H. A markovian-based approach for daily living activities recognition. **CoRR**, abs/1603.03251, 2016. Disponível em: <<http://arxiv.org/abs/1603.03251>>. Citado nas páginas 23, 46 e 75.
- LUCEY, P.; COHN, J. F. *et al.* The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In: **Computer Vision and Pattern Recognition Workshops, IEEE Computer Society Conf. on**. [S.l.: s.n.], 2010. Citado na página 80.
- Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. **IEEE Communications Surveys Tutorials**, v. 19, n. 3, p. 1628–1656, thirdquarter 2017. ISSN 2373-745X. Citado nas páginas 42, 43 e 44.
- MAIA, P.; BAFFA, A.; CAVALCANTE, E.; DELICATO, F. C.; BATISTA, T.; PIRES, P. F. A middleware platform for integrating devices and developing applications in e-health. In: **Computer Networks and Distributed Systems (SBRC), 2015 XXXIII Brazilian Symposium on**. [S.l.: s.n.], 2015. p. 10–18. Citado nas páginas 47 e 52.
- MANE, Y. V.; SURVE, A. R. Capm: Context aware provisioning middleware for human activity recognition. In: **2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)**. [S.l.: s.n.], 2016. p. 661–665. Citado nas páginas 47 e 52.
- MANO, L. Y.; FAIÇAL, B. S.; NAKAMURA, L. H.; GOMES, P. H.; LIBRALON, G. L.; MENEQUETE, R. I.; FILHO, G. P.; GIANCRISTOFARO, G. T.; PESSIN, G.; KRISHNAMACHARI, B.; UEYAMA, J. Exploiting iot technologies for enhancing health smart homes through patient identification and emotion recognition. **Computer Communications**, v. 89-90, p. 178 – 190, 2016. ISSN 0140-3664. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0140366416300688>>. Citado nas páginas 15, 46, 47, 52, 77, 78 e 82.
- MANO, L. Y.; VASCONCELOS, E.; UEYAMA, J. Identifying emotions in speech patterns: Adopted approach and obtained results. **IEEE Latin America Transactions**, v. 14, n. 12, p. 4775–4780, Dec 2016. ISSN 1548-0992. Citado na página 82.

- MASIP-BRUIN, X.; MARÍN-TORDERA, E.; TASHAKOR, G.; JUKAN, A.; REN, G. Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. **IEEE Wireless Communications**, v. 23, n. 5, p. 120–128, October 2016. ISSN 1558-0687. Citado nas páginas 24 e 27.
- MELL, P.; GRANCE, T. *et al.* The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011. Citado nas páginas 33 e 34.
- MIORANDI, D.; SICARI, S.; PELLEGRINI, F. D.; CHLAMTAC, I. Internet of things: Vision, applications and research challenges. **Ad Hoc Networks**, Elsevier, v. 10, n. 7, p. 1497–1516, 2012. Citado nas páginas 31 e 32.
- MOHAN, N.; KANGASHARJU, J. Edge-fog cloud: A distributed cloud for internet of things computations. In: **2016 Cloudification of the Internet of Things (CIoT)**. [S.l.: s.n.], 2016. p. 1–6. Citado nas páginas 27 e 35.
- MOURADIAN, C.; NABOULSI, D.; YANGUI, S.; GLITHO, R. H.; MORROW, M. J.; POLAKOS, P. A. A comprehensive survey on fog computing: State-of-the-art and research challenges. **IEEE Communications Surveys Tutorials**, v. 20, n. 1, p. 416–464, Firstquarter 2018. ISSN 2373-745X. Citado na página 35.
- MUTLAG, A. A.; GHANI, M. K. A.; ARUNKUMAR, N.; MOHAMMED, M. A.; MOHD, O. Enabling technologies for fog computing in healthcare iot systems. **Future Generation Computer Systems**, v. 90, p. 62 – 78, 2019. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X18314006>>. Citado nas páginas 23, 24, 26, 27, 32, 49, 58, 64, 66 e 75.
- NÉMETH, B.; SZALAY, M.; DÓKA, J.; ROST, M.; SCHMID, S.; TOKA, L.; SONKOLY, B. Fast and efficient network service embedding method with adaptive offloading to the edge. In: . [s.n.], 2018. p. 178–183. Cited By 2. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85050691390&doi=10.1109%2fINFCOMW.2018.8406882&partnerID=40&md5=47c445c2a6376e2478682b24e51fc4bd>>. Citado nas páginas 49 e 52.
- NETO, J. L. D.; YU, S.; MACEDO, D. F.; NOGUEIRA, J. M. S.; LANGAR, R.; SECCI, S. Uloof: A user level online offloading framework for mobile edge computing. **IEEE Transactions on Mobile Computing**, v. 17, n. 11, p. 2660–2674, Nov 2018. ISSN 2161-9875. Citado nas páginas 43, 50 e 52.
- NETO, J. R. T.; FILHO, G. P.; MANO, L. Y.; UEYAMA, J. Verbo: Voice emotion recognition database in portuguese language. **Journal of Computer Science**, v. 14, n. 11, p. 1420 – 1430, 2018. ISSN 1549-3636. Citado na página 84.
- NETO, J. R. T.; FILHO, G. P. R.; MANO, L. Y.; UEYAMA, J. Inca: um sistema healthcare flexível baseado no paradigma fog computing e publish/subscribe. In: **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC**. [S.l.]: SBC, 2017. Citado na página 24.
- NETO, J. R. T.; FILHO, G. P. R.; MANO, L. Y.; VILLAS, L. A.; UEYAMA, J. Exploiting offloading in iot-based microfog: Experiments with face recognition and fall detection. **Wireless Communications and Mobile Computing**, Hindawi, v. 2019, 2019. Citado na página 26.

NILSSON, N. J. **Principles of Artificial Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1980. ISBN 0-934613-10-9. Citado na página 39.

OLIVIERI, A. C.; RIZZO, G.; MORARD, F. A publish-subscribe approach to iot integration: The smart office use case. In: **Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on**. [S.l.: s.n.], 2015. p. 644–651. Citado na página 67.

PATEL, M.; HU, Y.; P., H.; JOUBERT, J.; THORNTON, C.; NAUGHTON, B.; RAMOS, J. R.; CHAN, C.; YOUNG, V.; TAN, S. J.; LYNCH, D.; SPRECHER, N.; MUSIOL, T.; MANZANARES, C.; RAUSCHENBACH, U.; ABETA, S.; CHEN, L.; SHIMIZU, K.; NEAL, A.; COSIMINI, P.; POLLARD, A.; KLAS, G. Mobile-edge computing. **ETSI White Paper**, v. 11, n. 1, p. 1–36, 2014. Citado na página 24.

PATEL, P.; ALI, M. I.; SHETH, A. On using the intelligent edge for iot analytics. **IEEE Intelligent Systems**, v. 32, n. 5, p. 64–69, September 2017. ISSN 1541-1672. Citado nas páginas 46 e 52.

PESSIN, G. **Estratégias inteligentes aplicadas em robôs móveis autônomos e em coordenação de grupos de robôs**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, University of São Paulo, São Carlos, Disponível em: <<http://www.teses.usp.br/teses/disponiveis/55/55134/tde-25062013-160156/>>, 2013. Citado na página 84.

PLUMMER, D. Top 10 strategic predictions for 2016 and beyond: The future is a digital thing. 2015. Citado na página 23.

PRADO, A. G. de; ORTIZ, G.; PUIG, J. B. Collect: Collaborative context-aware service oriented architecture for intelligent decision-making in the internet of things. **Expert Systems with Applications**, v. 85, n. Supplement C, p. 231 – 248, 2017. ISSN 0957-4174. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417417303573>>. Citado nas páginas 46 e 52.

QIAN, H.; ANDRESEN, D. Jade: Reducing energy consumption of android app. **the International Journal of Networked and Distributed Computing (IJNDC)**, Atlantis press, v. 3, n. 3, p. 150–158, 2015. Citado nas páginas 43, 44, 49 e 52.

RA, M.-R.; SHETH, A.; MUMMERT, L.; PILLAI, P.; WETHERALL, D.; GOVINDAN, R. Odessa: Enabling interactive perception applications on mobile devices. In: **Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services**. New York, NY, USA: ACM, 2011. (MobiSys '11), p. 43–56. ISBN 978-1-4503-0643-0. Disponível em: <<http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/1999995.2000000>>. Citado na página 66.

RA, M.-R.; SHETH, A.; MUMMERT, L. B.; PILLAI, P.; WETHERALL, D.; GOVINDAN, R. Odessa: enabling interactive perception applications on mobile devices. In: AGRAWALA, A. K.; CORNER, M. D.; WETHERALL, D. (Ed.). **MobiSys**. ACM, 2011. p. 43–56. ISBN 978-1-4503-0643-0. Disponível em: <<http://dblp.uni-trier.de/db/conf/mobisys/mobisys2011.html#RaSMPWG11>>. Citado na página 50.

RAJAGOPALAN, A.; JAGGA, M.; KUMARI, A.; ALI, S. T. A ddos prevention scheme for session resumption sea architecture in healthcare iot. In: **2017 3rd International Conference on**

Computational Intelligence Communication Technology (CICT). [S.l.: s.n.], 2017. p. 1–5. Citado na página 26.

RAZZAQUE, M. A.; MILOJEVIC-JEVRIC, M.; PALADE, A.; CLARKE, S. Middleware for internet of things: A survey. **IEEE Internet of Things Journal**, v. 3, n. 1, p. 70–95, Feb 2016. ISSN 2327-4662. Citado nas páginas 58 e 66.

REED, R. Pruning algorithms-a survey. **IEEE Transactions on Neural Networks**, v. 4, n. 5, p. 740–747, Sep. 1993. Citado na página 84.

RODRIGUES, D. O.; BOUKERCHE, A.; SILVA, T. H.; LOUREIRO, A. A.; VILLAS, L. A. Smaframework: Urban data integration framework for mobility analysis in smart cities. In: **Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems**. New York, NY, USA: ACM, 2017. (MSWiM '17), p. 227–236. ISBN 978-1-4503-5162-1. Disponível em: <<http://doi.acm.org/10.1145/3127540.3127569>>. Citado na página 24.

ROMERO, E.; ARAUJO, A.; MOYA, J. M.; GOYENECHE, J.-M. de; VALLEJO, J. C.; MALAGON, P.; VILLANUEVA, D.; FRAGA, D. Image processing based services for ambient assistant scenarios. In: **Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living**. [S.l.]: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5518). p. 800–807. ISBN 978-3-642-02480-1. Citado na página 23.

SAKR, S.; ELGAMMAL, A. Towards a comprehensive data analytics framework for smart healthcare services. **Big Data Research**, v. 4, n. Supplement C, p. 44 – 58, 2016. ISSN 2214-5796. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S221457961530023X>>. Citado na página 48.

SALMAN, H.; RAHMAN, M. S.; TAREK, M. A. Y.; WANG, J. The design and implementation of gps controlled environment monitoring robotic system based on iot and arm. In: **2019 4th International Conference on Control and Robotics Engineering (ICCRE)**. [S.l.: s.n.], 2019. p. 93–98. Citado na página 24.

SAQIB, M. T.; HAMID, M. A. Fogr: A highly reliable and intelligent computation offloading on the internet of things. In: **2016 IEEE Region 10 Conference (TENCON)**. [S.l.: s.n.], 2016. p. 1039–1042. ISSN 2159-3450. Citado nas páginas 24 e 48.

SHI, C.; HABAK, K.; PANDURANGAN, P.; AMMAR, M.; NAIK, M.; ZEGURA, E. Cosmos: computation offloading as a service for mobile devices. In: **ACM. Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing**. [S.l.], 2014. p. 287–296. Citado nas páginas 43 e 44.

SHUKLA, R. M.; MUNIR, A. An efficient computation offloading architecture for the internet of things (iot) devices. In: **2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2017. p. 728–731. ISSN 2331-9860. Citado nas páginas 64 e 65.

Artif. Intell. Rev., Kluwer Academic Publishers, Norwell, MA, USA, v. 11, n. 1-5, 1997. ISSN 0269-2821. Citado na página 77.

- SOUZA, A. M. de; PEDROSA, L. L. C.; BOTEAGA, L. C.; VILLAS, L. Itssafe: An intelligent transportation system for improving safety and traffic efficiency. In: **2018 IEEE 87th Vehicular Technology Conference (VTC Spring)**. [S.l.: s.n.], 2018. p. 1–7. Citado nas páginas 23 e 56.
- SOUZA, V.; MASIP-BRUIN, X.; MARIN-TORDERA, E.; SANCHEZ-LOPEZ, S.; GARCIA, J.; REN, G.; JUKAN, A.; FERRER, A. J. Towards a proper service placement in combined fog-to-cloud (f2c) architectures. **Future Generation Computer Systems**, v. 87, p. 1 – 15, 2018. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X17323051>>. Citado nas páginas 15, 23, 24, 25, 26, 27, 36, 48, 49, 51, 53, 55, 56, 58 e 64.
- STANFORD-CLARK, A.; NIPPER, A. Mq telemetry transport. 2014. Disponível em: <<http://mqtt.org>>. Citado nas páginas 59 e 67.
- STOJKOSKA, B. L. R.; TRIVODALIEV, K. V. A review of internet of things for smart home: Challenges and solutions. **Journal of Cleaner Production**, v. 140, Part 3, p. 1454 – 1464, 2017. ISSN 0959-6526. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S095965261631589X>>. Citado nas páginas 27 e 32.
- STONE, M. Cross-validatory choice and assessment of statistical predictions. **Journal of the royal statistical society. Series B (Methodological)**, JSTOR, p. 111–147, 1974. Citado na página 79.
- SYSTEM, M. A. **QMedic - Medical Alert System - Full Review**. 2015. Disponível em: <<http://medicalalertsystemreviews.net/qmedic-full-review/>>. Citado nas páginas 15 e 79.
- TALEB, T.; BOTTAZZI, D.; NASSER, N. A novel middleware solution to improve ubiquitous healthcare systems aided by affective information. **IEEE Transactions on Information Technology in Biomedicine**, v. 14, n. 2, p. 335–349, March 2010. ISSN 1089-7771. Citado nas páginas 23, 29 e 47.
- TEKIN, Y.; SAHINGOZ, O. K. A publish/subscribe messaging system for wireless sensor networks. In: **2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)**. [S.l.: s.n.], 2016. p. 171–176. Citado na página 67.
- THANGARAJ, M.; PONMALAR, P. P.; SUJATHA, G.; ANURADHA, S. Agent based semantic internet of things (iot) in smart health care. In: **Proceedings of the The 11th International Knowledge Management in Organizations Conference on The Changing Face of Knowledge Management Impacting Society**. New York, NY, USA: ACM, 2016. (KMO '16), p. 41:1–41:9. ISBN 978-1-4503-4064-9. Disponível em: <<http://doi.acm.org/10.1145/2925995.2926023>>. Citado nas páginas 47 e 52.
- VAPNIK, V. **The nature of statistical learning theory**. [S.l.]: Springer science & business media, 1995. Citado na página 77.
- VINH, T. L.; BOUZEFRANE, S.; FARINONE, J.-M.; ATTAR, A.; KENNEDY, B. P. Middleware to integrate mobile devices, sensors and cloud computing. **Procedia Computer Science**, Elsevier, v. 52, p. 234–243, 2015. Citado nas páginas 46 e 52.
- WEN, Y.; ZHANG, W.; LUO, H. Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. In: **2012 Proceedings IEEE INFOCOM**. [S.l.: s.n.], 2012. p. 2716–2720. ISSN 0743-166X. Citado na página 64.

- WIRSING, M. **Software Intensive Systems**. 2006. <http://www.pst.ifi.lmu.de/rauschma/interlink/cannes/groups/TG6.pdf>. Citado na página 39.
- WOOLDRIDGE, M.; JENNINGS, N. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 1995. Citado na página 38.
- XIA, F.; DING, F.; LI, J.; KONG, X.; YANG, L. T.; MA, J. Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. **Information Systems Frontiers**, Springer, v. 16, n. 1, p. 95–111, 2014. Citado nas páginas 43, 44, 49 e 52.
- XU, D.; LI, Y.; CHEN, X.; LI, J.; HUI, P.; CHEN, S.; CROWCROFT, J. A survey of opportunistic offloading. **IEEE Communications Surveys Tutorials**, v. 20, n. 3, p. 2198–2236, thirdquarter 2018. ISSN 2373-745X. Citado nas páginas 43 e 44.
- ZHAO, S.; RUDZICZ, F.; CARVALHO, L. G.; MARQUEZ-CHIN, C.; LIVINGSTONE, S. Automatic detection of expressed emotion in parkinson’s disease. In: **2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2014. p. 4813–4817. ISSN 1520-6149. Citado na página 86.
- ZHOU, H.; WANG, H.; LI, X.; LEUNG, V. C. M. A survey on mobile data offloading technologies. **IEEE Access**, v. 6, p. 5101–5111, 2018. Citado na página 24.
- ZHU, H.; ZHU, Z.; LUO, X.; QIAN, H. Distributed computation offloading in resource limited fog computing. In: . [s.n.], 2019. v. 2019-September. Cited By 0. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85075253987&doi=10.1109%2fVTCFall.2019.8891125&partnerID=40&md5=93cad85b405b15d5a52dc1626fabb7c2>>. Citado na página 52.
- ZHU, Q.; SI, B.; YANG, F.; MA, Y. Task offloading decision in fog computing system. **China Communications**, IEEE, v. 14, n. 11, p. 59–68, 2017. Citado nas páginas 44 e 65.