UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

EXEHDA-FC: Uma Abordagem para Processamento Distribuído de Contexto Explorando a Integração de Fog e Cloud Computing

Leonardo da Rosa Silveira João

Pelotas, 2020

Leonardo da Rosa Silveira João

EXEHDA-FC: Uma Abordagem para Processamento Distribuído de Contexto Explorando a Integração de Fog e Cloud Computing

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Adenauer Corrêa Yamin
Coorientador: Profa. Dra. Ana Marilza Pernas Fleischmann

Pelotas, 2020

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

J62e João, Leonardo da Rosa Silveira

EXEHDA-FC : uma abordagem para processamento distribuído de contexto explorando a integração de Fog e Cloud computing / Leonardo da Rosa Silveira João ; Adenauer Corrêa Yamin, orientador ; Ana Marilza Pernas, coorientadora. — Pelotas, 2020.

92 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2020.

1. Internet das coisas. 2. Ciência de contexto. 3. Fog computing. 4. Cloud computing. I. Yamin, Adenauer Corrêa, orient. II. Pernas, Ana Marilza, coorient. III. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Leonardo da Rosa Silveira João

EXEHDA-FC: Uma Abordagem para Processamento Distribuído de Contexto Explorando a Integração de Fog e Cloud Computing

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 27 de novembro de 2020

Banca Examinadora:

Prof. Dr. Adenauer Corrêa Yamin (Orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Profa. Dra. Ana Marilza Pernas Fleischmann (Coorientadora)
Doutora em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. João Ladislau Barbará Lopes
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Rafael Iankowski Soares
Doutor em Computação pela Pontifícia Universidade Católica do Rio Grande do Sul.

É muito melhor lançar-se em busca de conquistas grandiosas, mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito, que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta, onde não conhecem nem vitória, nem derrota.

— Theodore Roosevelt

RESUMO

DA ROSA SILVEIRA JOÃO, Leonardo. **EXEHDA-FC: Uma Abordagem para Processamento Distribuído de Contexto Explorando a Integração de Fog e Cloud Computing**. Orientador: Adenauer Corrêa Yamin. 2020. 92 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2020.

A Computação Ubíqua tem por premissa disponibilizar aos usuários, da maneira o mais transparente possível, funcionalidades computacionais integradas ao ambiente de seu interesse. Essa integração exige a utilização de dispositivos para sensoriamento e/ou atuação distribuídos no ambiente. A disseminação de alternativas para interligar estes dispositivos tem sido potencializada com a Internet das Coisas (IoT).

A IoT visa prover a intercomunicação entre dispositivos inteligentes com o uso da Internet, provendo a coleta de informações destes dispositivos heterogêneos, viabilizando assim o processamento contextual das mesmas, provendo suporte para decisões de atuação da maneira o mais autônoma possível.

As soluções baseadas na IoT contemplavam, via de regra, um processamento centralizado das informações contextuais sensoriadas, explorando para isto equipamentos em uma perspectiva de *Cloud Computing*. Por sua vez, a abordagem de *Fog Computing*, em ampla disseminação, visa a realização do processamento contextual de forma descentralizada e mais próxima de aonde ocorre a aquisição dos dados.

O trabalho de estudo e pesquisa desenvolvido nesta Dissertação tem como objetivo central a concepção de uma nova arquitetura para o Servidor de Borda do *middleware* EXEHDA, denominada *EXEHDA - Fog Computing* (EXEHDA-FC), capacitando os Servidores de Borda para aquisição e processamento distribuído de dados contextuais na IoT, contribuindo assim para o uso da abordagem de *Fog Computing*.

Este trabalho faz parte das atividades de pesquisa associadas ao Subsistema de Adaptação e Reconhecimento do Contexto (SARC) do EXEHDA e tem o propósito de agregar esforços nos estudos para qualificação do *middleware*. As avaliações realizadas do EXEHDA-FC consideraram um Cenário de Uso na área de Viticultura de Precisão, o qual explora situações típicas nas quais as premissas de *Fog* e *Cloud Computing* podem ser exploradas de forma combinada. Os resultados atingidos apontam para a continuidade dos esforços de estudo e pesquisa no tema.

Palavras-chave: Internet das Coisas. Ciência de Contexto. *Fog Computing*. *Cloud Computing*.

ABSTRACT

DA ROSA SILVEIRA JOÃO, Leonardo. **EXEHDA-FC: An Approach to Distributed Context Processing Exploring the Integration of Fog and Cloud Computing**. Advisor: Adenauer Corrêa Yamin. 2020. 92 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2020.

Ubiquitous Computing has as a premise to provide users, in the most transparent way possible, computational functionalities integrated to the environment of their interest. This integration requires the use of devices for sensing and/or acting distributed in the environment. The dissemination of alternatives to interconnect these devices has been enhanced with the Internet of Things (IoT).

The IoT aims to provide intercommunication between smart devices with the use of the Internet, providing the collection of information from these heterogeneous devices, thus enabling their contextual processing, providing support for acting decisions in the most autonomous way possible.

The solutions based on the IoT included, as a rule, a centralized processing of the contextual sensed information, exploring for this equipments in a perspective of Cloud Computing. On the other hand, the Fog Computing approach, in wide dissemination, aims to perform contextual processing in a decentralized way, in a layer closer to where the context data acquisition is performed.

The study and research work developed in this Dissertation has as its central objective the conception of a new architecture for the EXEHDA middleware Edge Server, called EXEHDA-FC, enabling Edge Servers for the acquisition and distributed processing of contextual data in IoT, thus contributing to the use of the Fog Computing approach.

This work is part of the research activities associated with the EXEHDA Context Recognition and Adaptation Subsystem and has the purpose of aggregating efforts in studies for the qualification of *middleware*. The evaluations carried out by EXEHDA-FC considered a Scenario of Use in the area of Precision Viticulture, which explores typical situations in Fog Computing. The results achieved point to the continuity of study and research efforts on the subject.

Keywords: Internet of Things. Context Awareness. Fog Computing. Cloud Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APScheduler *Advanced Python Scheduler*

BRT *Boeing Research and Development Division*

CES *Consumer Electronics Show*

CoT *Cloud of Things*

CORE *Common Open Research Emulator*

CEP *Complex Event Processing*

DNR *Distributed Node-RED*

ECA Evento-Condição-Ação

Embrapa Empresa Brasileira de Pesquisa Agropecuária

EXEHDA *Execution Environment for Highly Distributed Applications*

EXEHDA-FC *EXEHDA - Fog Computing*

GPS *Global Positioning System*

GW-SB Gateways Nativos

GWV-SB Gateway Virtual

IA Inteligência Artificial

IBM International Business Machines

IoT *Internet of Things*

JS *JavaScript*

JSON *JavaScript Object Notation*

KP *Knowledge Processors*

LASO Laboratório de Análise de Sementes Oficial

MQTT *Message Queue Telemetry Transport*

PoCo *Portability Core*

RD *Resource Directory*

REST *Representational State Transfer*

RR Programação Reativa

SB Servidor de Borda

SC Servidor de Contexto

SIB *Semantic Information Broker*

SoC *System on Chip*

SARC Subsistema de Adaptação e Reconhecimento do Contexto

TCP/IP *Transmission Control Protocol / Internet Protocol*

UbiComp Computação Ubíqua

VP Viticultura de Precisão

WoT *Web of Things*

SUMÁRIO

1 INTRODUÇÃO

A popularização crescente da Internet e a qualificação dos sistemas distribuídos, associado ao fato das tecnologias modernas potencializaram o emprego de soluções baseadas em hardware embarcado, criaram um cenário oportuno para a Computação Ubíqua (UbiComp). Este conceito, introduzido por Mark Weiser (WEISER, 1991), tem como objetivo transpor as funcionalidades dos computadores convencionais (centralizados) apontando para um cenário constituído por diversos objetos inteligentes, cada um com suas funcionalidades, dotados de dispositivos computacionais distribuídos e interligados por diferentes canais de comunicação (QIU et al., 2018).

A popularização de dispositivos computacionais embarcados e sua miniaturização, somada a disseminação das redes de computadores, consolidou a evolução da Internet, dando base a infraestrutura computacional que hoje é denominada *Internet of Things* (IoT) (DOBSON; CARLEY, 2018).

Nesse sentido, a IoT tem como característica central a capacidade de comunicação e transferência de dados entre objetos, pessoas, animais e recursos computacionais de diferentes portes, onde cada dispositivo possui um identificador único, sendo a Internet o meio de interoperação utilizado. Como exemplo de cenários com potencial uso da IoT, temos as casas inteligentes, os sistemas de transporte autônomos, as redes de energia, equipamentos de uso pessoal e sistemas agrícolas, dos quais podem ser coletadas informações e compartilhadas com sistemas computacionais de modo autônomo para tratamento.

Alguns desafios surgem no suporte ao desenvolvimento das aplicações, tais como: (i) a capacidade de processamento e gerenciamento das informações coletadas por meio de dispositivos heterogêneos; (ii) o tratamento da escalabilidade; (iii) a interpretação das informações contextuais coletadas considerando os interesses envolvidos; entre outros.

Considerando a elevada escalabilidade associada a IoT Outros desafios surgem em relação a infraestrutura para comunicação dos dados entre as bordas, onde os dados são coletados, e o local aonde serão armazenados, processados e/ou dispo-

nibilizados aos usuários para visualização. Torna-se em muitos cenários, essencial a utilização de técnicas de filtragem e/ou fusão de dados antes de transferi-los, visando otimizar o emprego dos recursos de rede, dos recursos de armazenamento, e em casos de conexões empregando canais de baixa velocidade, minimizar sua chance de sobrecarga.

Em alguns casos, também é necessária uma política de atuação sobre o ambiente Ubíquo provido pela IoT, baseada nos dados coletados, com atuação próxima ao tempo real. Para a proposta, em que o esforço para Ciência de Contexto parte também acontece nas bordas computacionais dá-se o nome de *Fog Computing* (HU et al., 2017). Como premissa central, o *Fog Computing* prevê uma gerência autônoma dos recursos, com regras disparadas a partir dos eventos que acontecem nos equipamentos de borda envolvidos.

Os dados coletados na infraestrutura computacional nem sempre precisam ser transmitidos para processamento em um equipamento centralizado, localizado na *Cloud*. Eles podem ser processados localmente nos próprios dispositivos embarcados, com o objetivo de economizar recursos de rede e armazenamento (NI et al., 2017). Visando melhorar a capacidade de agregação e abstração, os dispositivos embarcados podem requisitar dados de outros dispositivos em um mesmo espaço geográfico, colaborando para o processamento contextual e materializando assim as premissas do *Fog Computing* (YI; LI; LI, 2015).

Diferentes desafios estão presentes no provimento de suporte para as aplicações direcionadas a IoT, considerando o paradigma de *Fog Computing*. Dentre estes, destaca-se o gerenciamento das informações coletadas empregando de dispositivos distribuídos e a interpretação destas informações considerando o contexto de interesse das aplicações (GAZIS et al., 2015). As informações sensoriadas, no sentido de promover a compreensão do contexto das aplicações, denominam-se neste trabalho de dados contextuais.

1.1 Objetivos

Considerando este cenário, bem como a natureza operacional da IoT, o objetivo geral desta Dissertação de Mestrado é a concepção de uma arquitetura para os dispositivos de borda do SARC, denominada EXEHDA-FC, que faculte suporte ao processamento distribuído de contexto, integrando abordagens de *Fog* e *Cloud Computing*.

Para o atendimento deste objetivo geral foram identificas algumas metas, dentre as quais destacam-se as relacionadas a seguir:

- prover interoperabilidade entre os equipamentos que integram a camada de *Fog Computing*, facultando a sua cooperação, seja pela troca de informações sensoriadas, como pelo disparo de comandos remotos de atuação;

- explorar Programação *Dataflow* na definição das regras para processamento de contextual;
- conceber a proposta do EXEHDA-FC considerando sua integração no SARC do EXEHDA;
- divulgar as contribuições técnico-científicas do trabalho ante veículos da área de computação, sejam eventos ou revistas.

Dentre as possíveis contribuições do trabalho, considerando a escalabilidade inerente a quantidade de recursos conectados nas infraestruturas computacionais modernas, providas pela IoT, destacamos as otimizações quanto ao consumo de banda para tráfego de informações sensoriadas, bem como um potencial a redução no tempo de resposta ante aos dados provindos do meio.

1.2 Estrutura do Texto

O Capítulo 2 apresenta os conceitos trabalhados durante o desenvolvimento desta Dissertação de Mestrado, o que inclui a contextualização de Internet das Coisas, Contexto e Ciência de Contexto, *Fog Computing*, Programação Dataflow e Middleware EXEHDA. O Capítulo 3 apresenta o estudo realizado em pesquisas que discutem o uso de *Fog Computing* na perspectiva da Internet das Coisas *Dataflow*, os quais foram selecionados a partir de uma Revisão Sistemática de Literatura. A concepção do EXEHDA-FC é apresentada no Capítulo 4, sendo descrita uma visão da arquitetura proposta, suas premissas de concepção e tecnologias empregadas. No Capítulo 5, é apresentado o cenário de uso utilizado para avaliar as funcionalidades do EXEHDA-FC, bem como as avaliações realizadas. Por fim, o Capítulo 6 apresenta as considerações finais sobre o trabalho, revisando as principais contribuições dessa Dissertação de Mestrado, bem como, uma discussão de possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO CONCEITUAL

Neste Capítulo são abordados conceitos que constituem a fundamentação teórica necessária ao desenvolvimento desta Dissertação de Mestrado. A primeira Seção revisa conceitos sobre a Internet das Coisas e seus desafios. Na continuidade do Capítulo, na segunda Seção, são discutidos tópicos referentes a Contexto e Ciência de Contexto. Os conceitos e características da *Fog Computing* são discutidos na terceira Seção. Na sequência, é apresentado na quarta Seção o modelo de Programação *Dataflow* e suas aplicações. Por fim, a quinta Seção é destinada a apresentar os conceitos relacionados ao *middleware* EXEHDA e seus subsistemas, apresentando sua arquitetura de software e o ambiente ubíquo na IoT provido pelo mesmo.

2.1 Internet das Coisas: Revisando Conceitos

A IoT é um paradigma que vem mantendo um crescimento constante no cenário tecnológico mundial. Apresenta como característica central a capacidade de interoperação e transferência de dados entre "coisas", as quais podem ser objetos, pessoas, animais e até mesmo sistemas, onde cada elemento, quando necessário empregando próteses, possui um identificador único, sendo a Internet o meio de comunicação utilizado (AYDOS, 2016).

Nesta perspectiva, a IoT vem ganhando destaque como uma abordagem para promover a ubiquidade das soluções computacionais no mundo real, particularmente em relação aos diferentes aspectos estruturais que precisam ser atendidos para que se tenha uma solução computacional efetiva. Nesta perspectiva, a computação e seus diversos sistemas interagem com o ser humano, provendo informação de dispositivos, a todo o momento, independente de localização, constituindo um ambiente altamente distribuído, heterogêneo, dinâmico, móvel, mutável e com forte interação entre homem e máquina (DOBSON; CARLEY, 2018). Nesta Sessão são abordados os principais conceitos e características da IoT, bem como os desafios de pesquisa que estão sendo considerados pela comunidade técnico-científica.

2.1.1 Definições

A IoT tem por base um ambiente composto por uma variedade de “coisas”, materializadas na forma de objetos inteligentes (*smart objects*) os quais são embarcados com dispositivos computacionais capazes de prover: (i) conexão à Internet, por meio de esquemas de endereçamentos únicos; (ii) funcionalidades de sensoriamento; e (iii) processamento de dados contextuais. Estas funcionalidades os capacitam a interagir uns com os outros e cooperar de forma a criar novas aplicações e serviços para alcançar objetivos comuns.

Diversos desafios surgem no suporte ao desenvolvimento de aplicações no cenário computacional provido pela IoT, particularmente, considerando natureza do trabalho de pesquisa desta Dissertação de Mestrado destacaríamos: (i) a capacidade de processamento e gerenciamento das informações coletadas através de dispositivos heterogêneos; (ii) o tratamento da escalabilidade; e, (iii) a interpretação das informações contextuais coletadas considerando os interesses dos usuários.

Na literatura é possível encontrar mais do que uma definição para IoT, segundo (PERERA et al., 2013), isto ocorre pelo fato da visão da IoT ser muito ampla, além da existência de uma gama de desafios ainda a serem vencidos. A seguir apresenta-se três diferentes definições encontradas na literatura referente ao termo IoT.

- Definição 1: "As coisas (objetos, pessoas ou animais) têm identidades e personalidades virtuais que operam em espaços inteligentes utilizando interfaces capazes de se conectar e comunicar dentro de contextos sociais, ambientais e de usuários" (TAN; WANG, 2010).

- Definição 2: "Internet das Coisas pode ser definida como uma rede mundial de objetos interligados que possuem endereçamento único e que se comunicam através da Internet por meio de protocolos de comunicação padronizados." (BASSI; HORN, 2008).

- Definição 3: "A Internet das Coisas permite que pessoas e objetos possam se conectar a qualquer momento, em qualquer lugar, com qualquer coisa, de preferência usando qualquer caminho ou rede e qualquer serviço." (GUILLEMIN; FRIESS, 2009).

Para o âmbito deste trabalho, será utilizado a definição 3, visto que esta definição transparece a premissa da UbiComp. O conceito introduzido por Mark Weiser (WEISER, 1993), tem como objetivo transpor as funcionalidades dos computadores convencionais, que operam de forma centralizada, apontando para um cenário constituído por diversos objetos inteligentes, cada um com suas funcionalidades, dotados

de dispositivos computacionais distribuídos e interligados por diferentes canais de comunicação (LOPES et al., 2014). Considerando isto, na perspectiva desta Dissertação de Mestrado, a IoT é entendida como uma abordagem para a materialização da UbiComp.

2.1.2 Visões

A IoT possui três visões que se diferenciam, dependendo do foco conferido a pesquisa sendo desenvolvida (ČOLAKOVIć; HADžIALIć, 2018):

- Visão orientada à Internet: o foco da pesquisa sob o ponto de vista da Internet busca criar modelos e técnicas para a interoperabilidade de dispositivos em rede;
- Visão orientada as Coisas: foco desta visão de pesquisa é relacionada integração de objetos (Coisas) presentes em um cenário IoT. Pesquisas nesta visão apresentam propostas que buscam a garantia do melhor esforço no aproveitamento dos recursos dos dispositivos e sua comunicação;
- Visão orientada a Semântica: foco da pesquisa nesta visão está relacionado a comunicação e troca de informação dos diversos dispositivos interconectados. A comunicação ocorre pela Internet, tendo por base de protocolos de comunicação padronizados.

A Figura 1 apresenta as três visões contendo seus componentes primordiais. Ao observar as definições da IoT, as mesmas descrevem uma ou mais visões, por exemplo a Definição 2, da Seção anterior, descreve a semântica das expressões Internet e Coisas, e a Definição 3, descreve a união das três visões, por conter uma definição mais abrangente.

2.1.3 Principais Características da Infraestrutura Provida pela IoT

Um fator comum em aplicações IoT é a inerente inteligência da infraestrutura computacional, que reflete na denominação dos seus domínios, como por exemplo *smart home, smart health care, smart agriculture* e outros. Enquanto parte destas "*smart*" aplicações, os dispositivos podem coletar dados automaticamente, compartilhar informações entre si, e iniciar e executar serviços com o mínimo de intervenção humana.

Um dos principais desafios está em como interligar, reconhecer e armazenar o grande número de informações geradas por ambientes IoT, de forma a constituírem informações relevantes, ditas informações de contexto para as aplicações e seus usuários, bem como reagirem de forma inteligente a estas informações. Com base

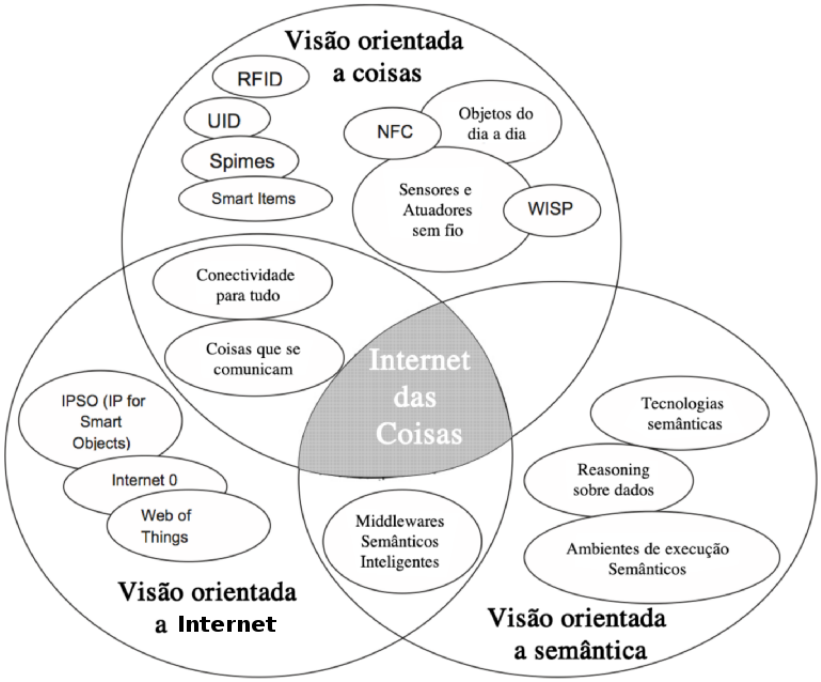

Figura 1 – Visões que Integralizam a Internet das Coisas
Fonte: (ATZORI; IERA; MORABITO, 2010)

neste cenário, algumas características são importantes para as infraestruturas, entre elas:

- Automatização: componente chave que implica que uma infraestrutura IoT deva suportar coleta de dados, processamento e inferência contextual de forma autônoma;
- Dinamicidade: um dispositivo pode se mover de um lugar a outro, necessitando que sua infraestrutura esteja apta para reconhecer esta mudança e consequentemente realizar a adaptação necessária baseada no estado atual do seu ambiente;
- Ciência de Contexto e Situação: dispositivos que possuam inteligência devem estar capacitados a operar de forma adaptativa a diferentes condições. A Ciência de Contexto e Situação vem se mostrando um componente chave para viabilizar sistemas inteligentes na IoT;
- Zero Configuração: para suportar a fácil integração de dispositivos, recursos *plug and play* devem estar disponíveis, otimizando a gerência dos dispositivos e possibilitando o crescimento descentralizado de sistemas IoT.

O grande facilitador para a integração destes diversos dispositivos independentemente de sua localização, por ser uma rede global, foi a Internet. Porém alguns fatores são tidos como importantes para a efetiva materialização da IoT, como os

avanços tecnológicos na miniaturização de dispositivos embarcados e inclusão de novos e diferentes sensores, atuadores e tags inteligentes. No entanto, há ainda uma série de desafios a serem superados para alavancar a ampla disseminação da IoT.

Um dos principais desafios inerentes em um cenário IoT, está relacionado com a alta heterogeneidade decorrente da diversidade de tecnologias de *hardware* e *software* presentes neste ambiente. Esta situação demanda que haja uma busca de soluções que permitam a interoperabilidade e integração destes diferentes componentes.

Uma alternativa de solução promissora que vem sendo amplamente utilizada, de forma a tratar desafios IoT, como o da heterogeneidade, está na utilização de plataformas de *middleware*. As plataformas de *middleware* são inseridas entre as aplicações e a infraestrutura (sensoriamento, comunicação e processamento) subjacente, provendo um meio padronizado para o acesso aos dados e serviços fornecidos pelos recursos (*Smart Objects*) por meio de uma interface com bom nível de abstração (BANDYOPADHYAY et al., 2011).

2.1.4 Desafios de Pesquisa

De acordo com a literatura (MACHADO; MORENO; RIBEIRO, 2017), alguns desafios de pesquisa inerentes aos ambientes IoT merecem destaque e constituem requisitos importantes para uma frente de trabalho como a contemplada nesta Dissertação de Mestrado.

Interfaces de Alto Nível

A adoção de uma plataforma de *middleware* também pode contribuir para facilitar a construção de aplicações para IoT. Desta forma, o desafio reside no fato de que, a fim de permitir a criação de aplicações que combinem recursos do mundo físico disponibilizados via redes de computadores, são necessários modelos que abstraiam os serviços e recursos físicos subjacentes. Com isso, usuários e aplicações consumidores dos dados originados dos dispositivos conectados, possuem acesso aos dados de forma padronizada, por meio de interfaces de alto nível, não necessitando assim lidar com funcionalidades de baixo nível para a manipulação de tais objetos.

Interoperabilidade

Dentre os requisitos, o considerado primordial e que deve ser imperativamente endereçado por uma plataforma de *middleware* para a IoT, diz respeito a interoperabilidade entre os diversos dispositivos e plataformas disponíveis neste ambiente. A sua importância é devida sobretudo por um número crescente de dispositivos a serem integrados neste novo cenário, sendo estes heterogêneos tanto em termos de

hardware e *software*, como também quanto aos protocolos que utilizam. Ressalte-se que muitos protocolos ainda são proprietários, o que caracteriza a interoperabilidade no cenário da IoT ainda como um grande desafio. Em (PIRES et al., 2015) é destacado que a integração de dispositivos no contexto de IoT acontece em múltiplos níveis:

- no nível mais baixo é necessário integrar de maneira transparente, uma grande quantidade de dispositivos físicos heterogêneos de modo a ocultar detalhes em relação à rede, formatos de dados e à semântica das informações;
- no nível intermediário é necessário integrar e disponibilizar dados providos por esses dispositivos, a fim de prover serviços de valor agregado aos usuários, podendo incluir funcionalidades com diferentes complexidades;
- no nível mais alto de um modelo padronizado de programação pode promover integração no que se refere à agregação e transformação de informações providas pelos dispositivos, a fim dos desenvolvedores de aplicações não necessitem ter qualquer conhecimento das especificações dos dispositivos e ambientes de rede.

Com dispositivos de diferentes naturezas interoperando, pode-se fazer uso de informações providas por diferentes meios e aplicações, permitindo que sejam criadas aplicações com maior valor agregado para os usuários.

Escalabilidade

A IoT é uma área da computação que registra um aumento de dispositivos conectados à Internet, com uma estimativa de 50 bilhões de dispositivos até o final de 2020 (AL-GARADI et al., 2020). As tecnologias envolvidas na IoT desempenham um papel fundamental de natureza transversal e em larga escala para atender o crescente número de dispositivos.

Como consequência, as plataformas de *middleware* para IoT devem atender ao requisito de escalabilidade, possuindo a capacidade para suportar requisições provindas de inúmeros dispositivos, mesmo em situações de uso intenso. Soluções de nuvem vem sendo utilizadas para este fim, devido à sua facilidade de provisão de recursos computacionais, que podem ser alocados e liberados sob demanda, proporcionando o surgimento da chamada “Nuvem das Coisas” (do inglês, *Cloud of Things* (CoT)) (SOLDATOS; SERRANO; HAUSWIRTH, 2012).

Raciocínio distribuído

Este desafio diz respeito ao emprego de técnicas para distribuir a capacidade do ambiente ubíquo para reconhecimento de contexto, bem como para promover as necessárias atuações. A IoT tem sido apontada como a viabilizadora de diversas

aplicações inteligentes, dessa forma são necessárias técnicas que possibilitem transformar os dados coletados em conhecimento sobre os ambientes. Este tratamento das informações via de regra é realizado considerando a perspectiva de reduzir a latência em transmissões, bem como o aumento da qualidade das mesmas.

As bordas computacionais, nesta perspectiva, devem ser capazes de utilizar do seu poder computacional, visando a coordenação e colaboração nas tomadas de decisões locais, processando esses dados em busca do atendimento das diferentes demandas das aplicações na IoT.

2.2 Contexto e Ciência de Contexto

Em sistemas computacionais informações de contexto traduzem aspectos de um mundo físico e lógico, capturadas por meio de sensores de diferentes naturezas.

Uma definição, entre as mais referenciadas é a de (DEY, 2001), a qual descreve que contexto é “qualquer informação que caracteriza a situação de uma entidade, sendo que uma entidade pode ser uma pessoa, um lugar ou um objeto. Considerados relevantes para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação. O contexto é tipicamente a localização, a identidade e o estado das pessoas, grupos ou objetos físicos e computacionais”.

O conceito de contexto vem sendo objeto de pesquisa em diversas áreas, destacando-se Psicologia Cognitiva, Linguística e Computação. Na Computação as áreas pioneiras na pesquisa do termo contexto foram a Computação Ubíqua e a Inteligência Artificial (IA) (DAVET, 2016), mostrando o potencial de sua utilização nos sistemas computacionais modernos.

A Ciência de Contexto refere-se a um modelo de computação no qual o sistema computacional é capaz de verificar as características que sejam do seu interesse e, quando necessário reagir sobre o mesmo. Quando utilizado na IoT, possui destaque no quesito do sistema tomar diversas decisões autônomas envolvendo os inúmeros dispositivos conectados, considerando as especificações do sistema ou do próprio usuário.

A Ciência de Contexto é a capacidade de um sistema utilizar o contexto para fornecer serviços e informações relevantes para o usuário (TEMDEE; PRASAD, 2018). Sistemas Cientes de Contexto devem ser flexíveis, adaptativos e capazes de atuar automaticamente para auxiliar o usuário na realização de suas tarefas (DAVET, 2016).

De acordo com (ALEGRE; AUGUSTO; CLARK, 2016) (Sezer; Dogdu; Ozbayoglu, 2018), para a construção de sistemas cientes de contexto em ambientes altamente distribuídos, como o da IoT, alguns desafios precisam ser tratados: (i) aquisição do contexto a partir de fontes heterogêneas e distribuídas; (ii) processamento dos dados

contextuais adquiridos e a respectiva atuação sobre o meio físico; e (iii) disponibilização distribuída dos dados contextuais processados, os quais são organizados em três componentes: aquisição, modelagem e processamento.

2.2.1 Aquisição do Contexto

A aquisição das informações contextuais é usualmente realizada por meio de um conjunto de sensores heterogêneos, não sendo limitada a coleta de grandezas físicas do ambiente, ocorrendo também a partir de qualquer fonte de dados capaz de prover informações contextuais (ALEGRE; AUGUSTO; CLARK, 2016). De acordo com (MOTTA; de Oliveira; TRAVASSOS, 2019), os sensores podem ser classificados em três grupos:

- Sensores Físicos: é o tipo de sensor mais utilizado. Trata-se de dispositivos capazes de capturar grandezas físicas diretamente do ambiente. Exemplos deste tipo de sensores são: sensores de movimento, temperatura, umidade, Ph, localização, entre outros.

- Sensores Virtuais: neste tipo de sensor as informações são provenientes de softwares ou serviços. Como exemplo pode-se citar uma identificação de atividade de usuário através de um software de monitoramento dos movimentos do mouse ou das entradas do teclado.

- Sensores Lógicos: este tipo de sensor disponibiliza informações contextuais geradas através da combinação de dados provenientes de diversas fontes, como sensores físicos, virtuais ou informações armazenadas em bancos de dados. Por exemplo, um sensor lógico poderia fornecer a localização de um usuário utilizando a posição do Sistema de Posicionamento Global (do inglês, *Global Positioning System* (GPS)) do seu *smartphone* combinado com informações de login em suas contas de e-mail armazenadas em banco de dados.

O desenvolvimento de sistemas computacionais para os cenários de IoT devem adotar estratégias que promovam a escalabilidade, pois nos ambientes de IoT a quantidade de dispositivos pode atingir números elevados (WANT et al., 2010). Por isso no momento da coleta das informações contextuais deve ser considerada a natureza do elemento monitorado para que não se façam requisições desnecessárias. Para apoiar essa decisão, algumas técnicas de coleta são apontadas por (PERERA et al., 2014), as quais são classificadas, tanto quanto a responsabilidade de quem promove a coleta, quanto a sua frequência de realização.

Quanto à responsabilidade, o processo de coleta das informações de contexto pode ocorrer de duas formas:

- *Pull*: nessa situação, o procedimento de coleta da informação contextual é disparado a partir do middleware, o qual faz uma consulta ao sensor.

- *Push*: nesse caso o procedimento de coleta é disparado a partir do sensor, o qual tem a iniciativa de enviar os dados capturados para o middleware empregando um procedimento de publicação.

A coleta das informações contextuais também pode ser classificada em dois tipos, quanto à frequência de aquisição:

- Instantânea: essa coleta é disparada por meio de eventos que ocorrem no ambiente. Por exemplo, a abertura de uma porta, o desligamento de uma lâmpada, uma temperatura atingindo um determinado valor, entre outros. Essa frequência de coleta associada ao método *push* promove uma reação rápida e mostra-se conveniente em eventos críticos onde se faz necessária uma tomada de decisão rápida.

- Periódica: essa coleta obedece a períodos temporais específicos, via de regra atrelados a aplicação. Podendo estar associado a horários e dias pré-estabelecidos e intervalos de tempo. Ambas abordagens podem ser implementadas tanto por meio do método *pull*, quanto pelo método *push*.

2.2.2 Modelagem do Contexto

Devido a significativa quantidade de informações capturadas e manipuladas pelos sistemas que tratam contexto, existe um crescente estudo de técnicas de representação das informações contextuais. Um modelo de contexto é a representação de como as informações contextuais são consideradas em um domínio ou aplicação e de que modo se relacionam com o sistema (MACHADO et al., 2018).

O processamento do modelo contextual consiste na elaboração de um modelo de entidades do mundo real, considerando suas propriedades, estados de seus ambiente e situações que podem ser utilizadas como referência para a aquisição, interpretação e raciocínio de informações contextuais (El Kadiri et al., 2016). A modelagem de contexto prove benefícios, dentre estes como facilitar o acesso às informações de forma eficaz, reduz a complexidade das aplicações cientes de contexto, e qualifica a capacidade de manutenção e evolução da aplicação.

Diversos modelos vem sendo empregados a fim de representar contextos (PERERA et al., 2013), tais como: chave-valor, linguagem de marcação, gráfico, orientado a objetos, ontológico. Neste contexto, modelos híbridos vem se mostrando promissores por combinarem diferentes técnicas de modelagem, com diferentes níveis de interpretação, para diversos aspectos.

2.2.3 Processamento do Contexto

O Processamento do Contexto considera aspectos referentes à interpretação, agregação, armazenamento, consulta e inferência das informações contextuais que foram concebidas a partir da aquisição de dados do contexto, tendo em vista possibilitar a compreensão do contexto de interesse através da geração de informações. Assim, o processamento do contexto pode ser estabelecido como um mecanismo de raciocínio para inferir novos conhecimentos e melhorar a compreensão das informações de contexto adquiridas (BIBRI, 2015).

A interpretação de contexto consiste em um conjunto de métodos e processos que realizam a abstração, mapeamento, manipulação, agregação, derivação, inferência e outras ações sobre as informações contextuais, com o princípio de facilitar o entendimento de um determinado contexto e auxiliá-los na tomada de decisão, esse processo de interpretação consiste, genericamente, na manipulação e refinamento das informações contextuais obtidas do ambiente.

Neste cenário, um dos grandes desafios na utilização de informações contextuais é consequência do processamento destas informações necessitar realizar deduções de contexto de alto nível, sobre conjuntos de contexto de baixo nível, a partir de informações muitas vezes dispersas e desconexas (ALEGRE; AUGUSTO; CLARK, 2016).

2.3 Fog Computing

A *Fog Computing* é um novo paradigma computacional, que tem por base “Servidores de Borda”, que estendem a Computação em Nuvem até os pontos terminais da rede. Estes servidores, então fornecem recursos de computação, comunicação, controle, armazenamento computacional e serviços nas bordas entre os mundos computacionais e físicos (HU et al., 2017). Além disto, é usual este tipo de equipamento fornecer a inteligência lógica para os dispositivos finais, como sensores e atuadores, e filtrar os dados produzidos pelos mesmos.

A *Cloud Computing*, por sua vez, vem sendo uma das tecnologias mais promissoras para a materialização da IoT, por oportunizar sistemas escaláveis sob demanda, garantindo capacidade de armazenamento de dados e processamento (De Donno; Tange; Dragoni, 2019). Mesmo oferecendo essas funcionalidades, soluções baseadas exclusivamente na nuvem necessitam de uma conexão ativa com a Internet, visando o processamento dos dados contextuais e atuações sob o meio.

Neste sentido, para atender a necessidade de garantir confiabilidade operacional, mesmo quando de eventuais desconexões, surge a motivação central para a *Fog Computing*.

2.3.1 Fog Computing: Arquitetura Hierárquica

A arquitetura hierárquica da *Fog Computing* é um tópico de pesquisa significativo na área de sistemas distribuídos. Nos últimos anos várias propostas de arquitetura para *Fog Computing* surgiram na literatura, as quais são derivadas principalmente da arquitetura de três camadas (HU et al., 2017). Nesta proposta a *Cloud* estende a capacidade de computação, comunicação, armazenamento e serviço na borda da rede, introduzindo uma camada de névoa entre os dispositivos finais e a *Cloud* propriamente dita. Devido a coexistência dos dispositivos em conexões de rede próximas, a *Fog Computing* potencialmente tem um tempo de resposta menor o que a *Cloud*. Outro aspecto importante a ser considerado, que os dados brutos gerados pelos dispositivos podem ser pré-processados na infraestrutura da *Fog*, sendo enviadas para a *Cloud* somente as informações que precisam de processamento adicional e/ou registro histórico. Consequentemente, a utilização da *Fog Computing* pode promover a diminuição do consumo de rede, bem como, contribuir para a redução da carga computacional na infraestrutura de *Cloud*. A Figura 2 apresenta a arquitetura hierárquica de uma infraestrutura que combina *Fog* e *Cloud Computing*.

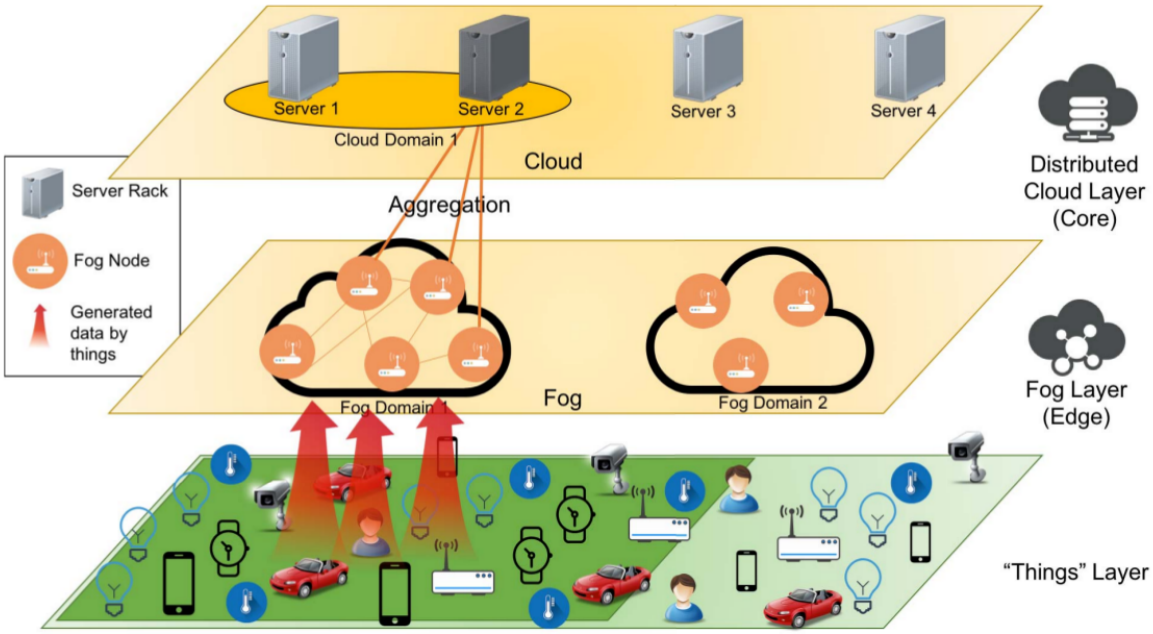

Figura 2 – Visão Geral da Integração: IoT, *Fog* e *Cloud Computing*
Fonte: (YOUSEFPOUR; ISHIGAKI; JUE, 2017)

A arquitetura hierárquica da IoT-Fog-Cloud é organizada em três camadas:

- *Things*: essa camada mais próxima do usuário final e do ambiente físico, consiste em dispositivos para IoT. Os dispositivos podem ser: (i) sensores; (ii) atuadores; (iii) smartphone; (iv) smartwatch; (v) carros inteligentes, entre outros.

Mesmo que alguns dispositivos possuem poder computacional para o processamento dos dados contextuais, o objetivo é coletar os dados do ambiente e os transmitir à camada superior, a qual irá efetivar o processamento e o armazenamento dos dados contextuais.

- *Fog Computing*: essa camada intermediaria, localizada entre as camadas *Things* e *Cloud*, é composta por diversos nodos que incluem: (i) roteadores; (ii) gateways; (iii) comutadores; (iv) pontos de acesso, entre outros. Os nodos que compõem esta camada podem compartilhar colaborativamente recursos de processamento e armazenamento. A análise em tempo real é realizada nessa camada e os nodos que estão conectados com os servidores na nuvem são responsáveis pela interação e cooperação com outros cenários em estudo.

- *Cloud Computing*: esta camada consiste em diversos servidores de alta desempenho e dispositivos de armazenamento. Possui recursos poderosos de computação e armazenamento para suportar a análises computacionais complexas, bem como o armazenamento permanente de uma quantidade significativa de dados contextuais oriundos dos ambientes em monitoramento pelos dispositivos (SARKAR, 2016).

Como pode ser visto na Figura 2, na arquitetura hierárquica explorada, cada dispositivo final na camada *Things* é conectado a um dos nós da camada imediatamente superior, *Fog Computing*, principalmente por meio de tecnologias de acesso sem fio. Por sua vez, os nós da *Fog Computing* podem operar conectados por tecnologias de comunicação com ou sem fio, sendo cada um dos mesmos vinculados à camada de *Cloud* utilizando um endereço de IP.

2.3.2 Fog Computing: Principais Características

Por premissa, a *Fog Computing* realiza tarefas de computação, comunicação e armazenamento em dispositivos de borda. A mesma trata não somente as requisições das aplicações locais, como daquelas externas. A *Fog Computing* possui a capacidade de “raciocínio” e assim pode auxiliar nas tomadas de decisão considerando os dados contextuais dos ambientes em monitoramento. Essas características da *Fog Computing* constituem a vantagem mais significativa em comparação com outros modelos. Além das citadas, esta Seção apresenta outras características consideradas essenciais da *Fog Computing*(HU et al., 2017)(Peralta et al., 2017):

- Proximidade da borda: a *Fog Computing* possui a localização dos serviços mais próximo dos usuários finais e integrados à rede local. Estes serviços são responsáveis por adquirir os dados gerados por sensores ou outros dispositivos,

processando e armazenando os mesmos. Dessa forma, fica reduzido significativamente o tráfego de dados pela Internet, sendo fornecidos serviços localizados e potencialmente com melhores níveis de QoS;

- Aplicações próximas ao tempo real: com uma latência baixa as aplicações da *Fog Computing* podem se beneficiar deste modelo computacional cujos serviços estão próximos em termos de latência. Neste sentido, é possível atender às demandas de interações em tempos curtos, o que é especialmente importante para aplicativos sensíveis à latência ou ao tempo;

- Organização hierárquica: a *Fog Computing* é estruturada de forma distribuída, com uma hierarquia entre os equipamentos, para prover uma baixa latência e alta escalabilidade, de forma que o gerenciamento possa ser coordenado pelo topo da hierarquia;

- Distribuição geográfica: a *Fog Computing* emprega dispositivos amplamente distribuídos ao londo dos ambientes em monitoramento. Essa característica provê suporte a análises de dados mais rápidas, pois os serviços de processamento e armazenamento das informações contextuais, estão distribuídos de maneira descentralizada, garantindo a análise dos dados de contexto de uma maneira mais "próxima"do dispositivo final;

- Escalabilidade: devido a organização hierárquica é possível utilizar uma grande quantidade de dispositivos, ampliando a escala da infraestrutura computacional envolvida em soluções de *Fog Computing*;

- Suporte a mobilidade: a *Fog Computing* contempla mobilidade espacial na última camada de sua arquitetura. Para tanto, existem diversos cenários que são compostos por dispositivos conectados a uma rede, seja ela Wi-Fi ou provida por operadoras de celular (2G, 3G ou 4G);

- Interoperabilidade: os recursos de uma arquitetura *Fog Computing* devem possuir a capacidade de interoperar e compartilhar recursos e serviços com os mais diferentes dispositivos, mesmo estes sendo heterogêneos tanto em hardware como software;

- Comunicação com a nuvem: os nodos da *Fog Computing* devem possuir comunicação com a infraestrutura de nuvens computacionais, para eventual troca de dados.

A *Fog Computing* não tem por premissa substituir a *Cloud Computing*, e sim, estendê-la em uma perspectiva de utilização em conjunto, para atendimento com as aplicações de IoT, tratando da economia de tráfego e latência de envio para a

nuvem, fazendo com que processamento dessas informações ocorra o maior parte na borda, ou processadas o mais perto possível dos pontos de sensoriamento.

Na perspectiva da *Fog Computing* a computação deve realizar de maneira distribuída, tendo em vista obter otimizações nos resultados da análise de dados contextuais e também utilizar-se dos recursos do ambiente da melhor maneira possível. Os dados mais críticos que não possuem a necessidade de análise histórica, porém necessitam serem tratados quanto a sua ocorrência deverão serem analisados nas bordas, estes dados das análises podem serem oriundas de diferentes nodos *Fog*.

2.4 Programação *Dataflow*

O modelo de Programação *Dataflow* fornece flexibilidade e, ao mesmo tempo, promove uma facilidade de uso em sistemas que facultem o emprego de um paradigma de programação baseado em fluxo de dados (VAQUERO; RODERO-MERINO, 2014).

Seu emprego está ganhando destaque no desenvolvimento de aplicações em diversas áreas do conhecimento. Uma das razões para este crescimento está alicerçada pela necessidade de os usuários finais manipularem diretamente os recursos disponibilizados pelos complexos ambientes computacionais modernos, os quais, muitas vezes, manipulam inúmeras variáveis de controle (Giang et al., 2015a).

Neste modelo, em que as aplicações são construídas empregando grafos que conectam nós que trocam dados, a lógica da aplicação é expressa como um grafo direcionado. Para cada nó é possível existir entradas, saídas e unidades de processamento independentes. Para isto, existem nós que produzem apenas saídas e aqueles que consomem apenas entradas, que geralmente representam o início e o fim de um determinado fluxo de dados. As unidades de processamento dos nós, computam as entradas de dados associadas a elas, produzindo saídas para os próximos nós. A unidade de processamento de um nó é executada independentemente, não afetando a execução de outros nós. Estas características, dentre outros aspectos, potencializam a possibilidade de reutilização de nós no desenvolvimento de novas aplicações (Giang et al., 2015b).

Com a finalidade de facilitar a transição entre projeto e a implementação, reduzindo o tempo de desenvolvimento, vários sistemas modernos fornecem uma interface gráfica para a construção dos fluxos de dados das aplicações. Embora o modelo de Programação *Dataflow* tenha sido originalmente concebido com o objetivo de ser utilizado na programação de aplicações paralelas, em hardwares dotados de multiprocessadores, o mesmo tem sido bastante adotado em sistemas distribuídos. Nesta Dissertação de Mestrado, nos referimos ao modelo de Programação *Dataflow*, principalmente como uma linguagem de coordenação para o desenvolvimento de

aplicativos cientes de contexto na IoT (BLACKSTOCK; LEA, 2014).

Uma característica oportuna do modelo de Programação *Dataflow* é o fato deste aumentar o nível de abstração na interconexão entre os módulos que compõem uma determinada aplicação. Este aumento da abstração auxilia a tarefa dos desenvolvedores na concepção dos diferentes fluxos de dados, bem como, permite flexibilidade na criação de novos fluxos. Isto ocorre porque que o hardware, os protocolos de comunicação e as funcionalidades dos sistemas de IoT são abstraídas em nós, fica implícita a manipulação das conexões entre os nós de entradas, saídas e processamento (Giang et al., 2015b).

A modelagem das aplicações neste caso, como irá consistir em grafos direcionados, sua concepção e adequação para os ambientes dinâmicos da IoT, pode ser gerenciada com flexibilidade.

2.5 Middleware EXEHDA

Esta Seção apresenta os principais conceitos relacionados ao *middleware Execution Environment for Highly Distributed Applications* (EXEHDA), bem como os subsistemas que compõe a sua arquitetura. O EXEHDA é um *middleware* baseado em serviços que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução, sob este ambiente de aplicações distribuídas em infraestruturas como a IoT. Estas aplicações são cientes do contexto de seu interesse, estando disponíveis para serem acessadas a partir de qualquer lugar, todo o tempo (LOPES, 2016a).

2.5.1 Aspectos Funcionais

O *middleware* EXEHDA tem como premissa definir a arquitetura para um ambiente de execução destinado às aplicações da computação ubíqua na plataforma computacional da IoT, no qual as condições de contexto são proativamente monitoradas e o suporte à execução deve permitir que tanto a aplicação como o próprio middleware utilizem estas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. Entende-se por adaptação funcional aquela que implica a modificação do código sendo executado. Por sua vez, adaptação não funcional é aquela que atua sobre a gerência da execução distribuída. Também a premissa siga-me das aplicações ubíquas deverá ser suportada, garantindo a execução da aplicação do usuário em qualquer tempo, lugar e dispositivo (LOPES et al., 2012).

As aplicações alvo são distribuídas, adaptativas ao contexto em que executam e compreendem a mobilidade lógica e a física. Na perspectiva do EXEHDA, entende-se por mobilidade lógica a movimentação entre dispositivos de artefatos de software e seu contexto, e por mobilidade física o deslocamento do usuário, portando ou não seu dispositivo.

2.5.2 Arquitetura de Software

Os principais requisitos que o EXEHDA se propõe atender são: (i) gerenciar tanto aspectos não funcionais como funcionais das aplicações, de modo independente; (ii) dar suporte à adaptação dinâmica de aplicações; (iii) disponibilizar mecanismos para obter e tratar informações de contexto; (iv) empregar informações de contexto na tomada de decisões; (v) decidir as ações adaptativas de forma colaborativa com a aplicação; e (vi) disponibilizar a semântica siga-me, permitindo ao usuário iniciar as aplicações e acessar dados a partir de qualquer lugar, e executar as aplicações continuamente mesmo em deslocamento. A arquitetura de software do *middleware* EXEHDA pode ser vista na Figura 3.

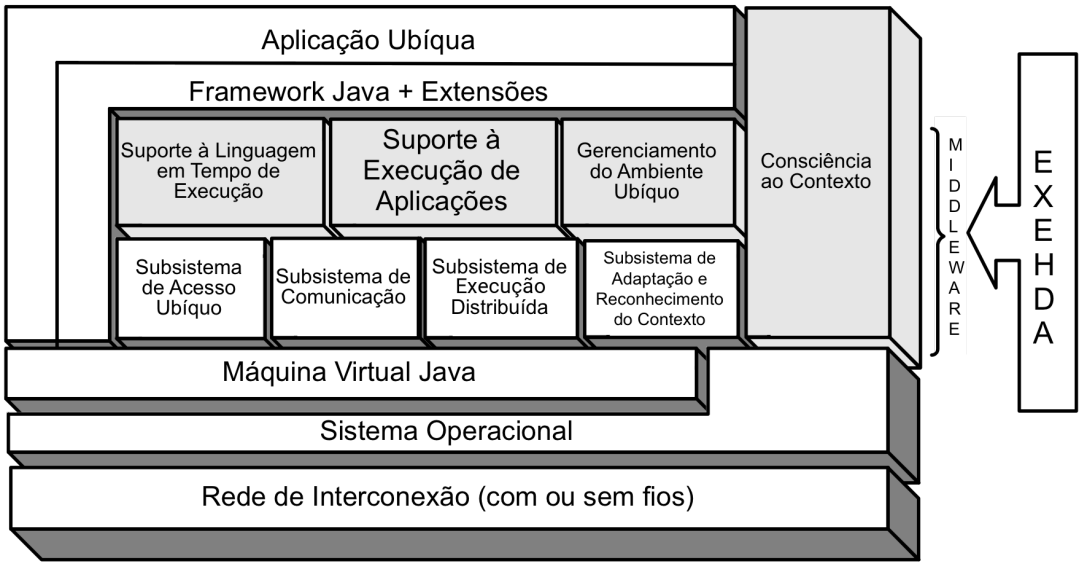

Figura 3 – Arquitetura de Software do Middleware EXEHDA
Fonte: (LOPES et al., 2014)

O *middleware* EXEHDA é organizado em quatro subsistemas (LOPES, 2016b), entre eles: (i) Subsistema de Acesso Ubíquo; (ii) Subsistema de Comunicação; (iii) Subsistema de Execução Distribuída; e (iv) Subsistema de Adaptação e Reconhecimento de Contexto. A seguir uma descrição resumida do mesmo:

- Subsistema de Acesso Ubíquo: comprometido com a premissa de acesso em qualquer lugar e todo o momento, provê suporte ao *middleware* para a recuperação de dados disponibilizados remotamente. Para isto, é construído um Ambiente Virtual para prover suporte aos usuários para acesso às suas aplicações e os seus dados de forma distribuída. Dentre outros aspectos, provê suporte a gerência de sessão dos usuários. Dentre outros aspectos, deverá possibilitar recursos para acesso por meio de um *logins*;

- Subsistema de Comunicação: a natureza do hardware na IoT, e também muitas vezes, do software, não garante a comunicação contínua entre os componentes das aplicações ubíquas. Em alguns casos as desconexões são comuns, não sendo um problema estreitamente de rede de comunicações, mas principalmente por estratégias para economia de energia nos dispositivos ou pelo perfil de uso dos usuários;

- Subsistema de Execução Distribuída: é responsável pelo suporte ao processamento distribuído no EXEHDA. No propósito de promover uma execução transparente, este subsistema interage com outros subsistemas do EXEHDA, em particular com o Subsistema de Reconhecimento de Contexto e Adaptação para prover um comportamento que se adapte às demandas das aplicações ubíquas no cenário da IoT;

- Subsistema de Adaptação e Reconhecimento de Contexto: compreende serviços que tratam a coleta de dados contextuais do ambiente monitorado, caracterizando o que os dados significam naquele ambiente, até o disparo das ações de adaptação como decorrência de modificações nos estados dos elementos de contexto.

2.5.3 Ambiente Ubíquo Disponibilizado

O ambiente ubíquo na IoT provido pelo middleware, corresponde a uma infraestrutura computacional onde recursos e serviços são gerenciados pelo EXEHDA com o propósito de atender os requisitos da aplicação. A composição deste ambiente envolve tanto os dispositivos dos usuários, como os equipamentos empregados como suporte, todos instanciados conforme seu respectivo perfil de execução no *middleware*. A infraestrutura do ambiente distribuído é mapeada em uma organização composta por um conjunto de células de execução, conforme pode ser observado na Figura 4.

O meio físico sobre o qual o ambiente ubíquo é definido e provido por uma rede infraestruturada, cuja composição final pode ser alterada pela agregação dinâmica de nodos. Os recursos físicos do meio computacional são mapeados para três abstrações, as quais são empregadas na composição dos diferentes dispositivos do ambiente ubíquo (SOUZA et al., 2018):

- EXEHDAcels: indica a área de atuação de uma EXEHDAbase e é composta por esta e por EXEHDAnodos. Os principais aspectos considerados na definição da abrangência de uma célula são: o escopo institucional, a proximidade geográfica e o custo de comunicação entre os nodos;

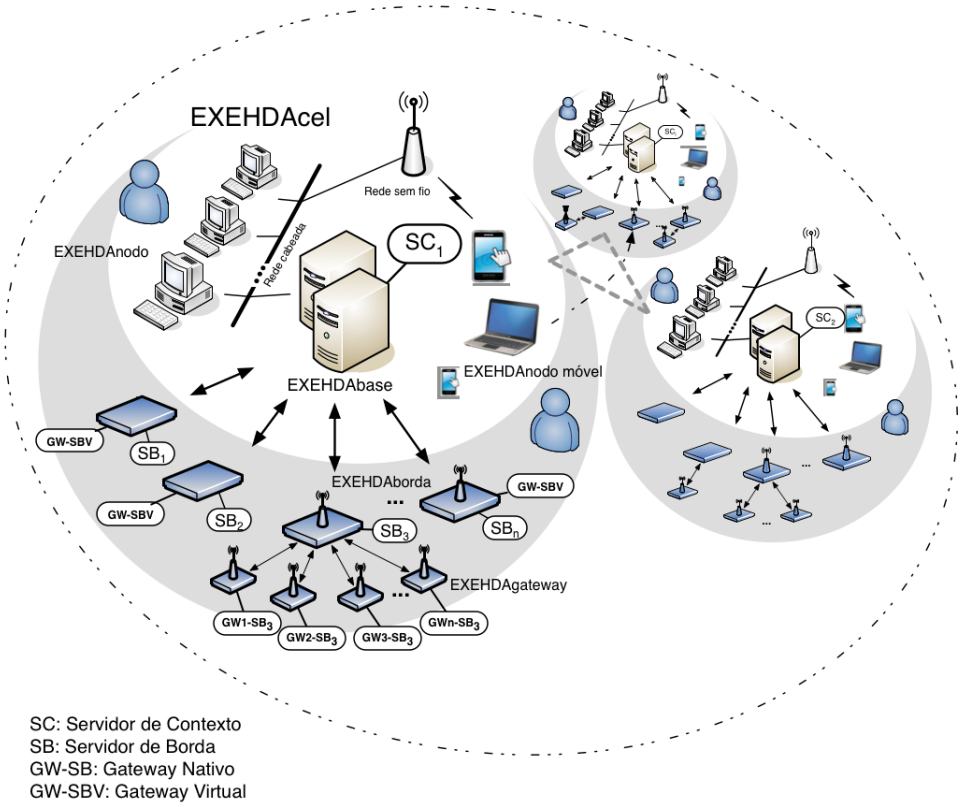

Figura 4 – Ambiente Ubíquo Gerenciado pelo EXEHDA
Fonte: (SOUZA et al., 2018)

- EXEHDAbase: é o ponto de convergência para os EXEHDAnodos. Este nodo é responsável por todos os serviços básicos do ambiente ubíquo e, embora constitua uma referência lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre os vários dispositivos;

- EXEHDAnodo: são os dispositivos de processamento disponíveis no ambiente ubíquo, sendo responsáveis pela execução das aplicações. Um subcaso deste tipo de recurso é o EXEHDAnodo móvel. São os nodos do sistema com elevada portabilidade, tipicamente dotados de interface de rede para operação sem fio e, neste caso, integram a célula a qual seu ponto-de-acesso está subordinado. São funcionalmente análogos aos EXEHDAnodos, porém eventualmente com uma capacidade mais limitada (por exemplo, smartphones, tablets, etc.).

Na composição do ambiente podem estar presentes sistemas embarcados, os quais são instanciados como EXEHDAnodos junto ao *middleware*.

Conforme pode ser observado na Figura 4, o EXEHDA possui uma organização composta por um conjunto de células de execução. Estas células, no que diz respeito ao provimento de Ciência de Contexto, são compostas por Servidores de Contexto, Servidores de Borda e Gateways.

O Servidor de Contexto, é alocado no EXEHDAbase e fornece as funcionalidades para a Ciência de Contexto, para isto provê o armazenamento e o processamento das informações contextuais, integrando dados provenientes de diferentes Servidores de Borda, distribuídos ao longo do ambiente ubíquo. O Servidor de Borda, por sua vez, é responsável por interagir com o ambiente por meio de Gateways que gerenciam sensores e atuadores, sendo instanciado em um equipamento do tipo EXEHDAborda.

Os Gateways são hardwares com capacidades computacionais limitadas, sendo dedicados para tratar tecnologias específicas, realizando a intercomunicação de protocolos e o gerenciamento dos dispositivos, possuem a finalidade de tratar os diversos protocolos existentes para conectar fisicamente os sensores e atuadores. Além disso, realizam comunicação com o Servidor de Borda para a troca de dados contextuais do ambiente em monitoramento. Atualmente, vários modelos de Gateways IoT vem sendo disponibilizados no mercado (GAZIS et al., 2015).

Com o objetivo de garantir a interoperabilidade com as tecnologias de mercado, e potencializar a distribuição dos hardwares dos Gateways, foram previstos dois tipos distintos, entre eles:

- Gateways Nativos (GW-SB): cujas funcionalidades são mapeadas em um hardware embarcado específico, cujas funcionalidades são instanciadas em equipamentos do tipo EXEHDAgateway;
- Gateway Virtual (GWV-SB): instanciado no mesmo hardware em que processa o Servidor de Borda, constituindo-se em uma virtualização do Gateway Nativo, dispensando, quando possível, a necessidade de um hardware específico para implementação das funcionalidades de um Gateway.

2.6 Considerações Sobre o Capítulo

Este Capítulo apresentou os principais aspectos pertinentes a revisão conceitual realizada para o desenvolvimento desta Dissertação de Mestrado. Foram discutidos os principais conceitos relativos a Internet das Coisas, o emprego de Ciência de Contexto e Situação enquanto mecanismo para prover comportamento autônomo nas aplicações ubíquas da IoT. Também foram introduzidos os principais desafios relacionados a *Fog Computing*, as características da Programação *Dataflow*, e por fim uma visão geral da arquitetura de software do middleware EXEHDA e seu ambiente de execução.

3 TRABALHOS RELACIONADOS

Este Capítulo tem como objetivo apresentar o estudo bibliográfico feito, o qual buscou identificar pesquisas que exploram *Fog* e *Cloud Computing* enquanto infraestruturas computacionais distribuídas para Processamento de Contexto, bem como *Dataflow* enquanto modelo de programação.

Neste sentido, foi realizada uma Revisão Sistemática de Literatura sobre o tema. Na Seção seguinte é apresentado o método empregado e posteriormente são discutidos os trabalhos relacionados identificados.

3.1 Revisão Sistemática da Literatura

Foi realizada uma revisão sistemática baseada no processo proposto por (PETERSEN; VAKKALANKA; KUZNIARZ, 2015), o qual estabelece uma série de atividades a serem executadas e registradas, permitindo que o procedimento desenvolvido possa ser reproduzido por outros pesquisadores.

Para a implementação da revisão sistemática utilizou-se a ferramenta StArt[1] para auxiliar na organização dos vários artigos envolvidos. Na primeira etapa do processo mencionado, foram definidas as seguintes questões de pesquisa:

- Quais trabalhos direcionados aos cenários de execução distribuídos como a IoT, exploram o uso da programação Dataflow?
- Quais trabalhos exploram *Fog Computing* e consideram o emprego de programação Dataflow?

Na pesquisa para identificação dos estudos primários, inicialmente foram estabelecidos os seguintes critérios para seleção dos artigos, entre eles:

- Disponibilidade em bibliotecas digitais e bases científicas, acessíveis pela Web;
- Publicados em periódicos e conferências, relacionados a IoT, *Edge* e *Fog Computing*, e *Programação Dataflow*;

[1] http://lapes.dc.ufscar.br/tools/start_tool

- Estarem escritos em inglês.

As bases acadêmicas selecionadas para a revisão foram as usualmente mais consideradas para publicações científicas na área de computação: ACM, IEEE, Science Direct, Scopus e Web of Science. Considerando as questões de pesquisa elencadas, as palavras-chave definidas para a busca de artigos foram: *Fog Computing*, *Edge Computing*, *IoT* e *Dataflow*.

O processo de busca pelos artigos seguiu um fluxo de execução onde inicialmente foi estabelecida uma *string* de pesquisa, conforme apresentado na Figura 5. Esta *string* teve por base as palavras-chave definidas e sua sintaxe foi ajustada para cada base acadêmica empregada.

```
(("fog computing") OR ("Internet of Things") OR ("edge computing")) AND ("dataflow"))
```

Figura 5 – Revisão Sistemática de Literatura: String de Pesquisa Utilizada

A *string* de pesquisa foi aplicada em cada base considerada para identificação dos artigos. A Figura 6 apresenta o número de artigos identificados por ano em cada base utilizada. Um fato a ser destacado é que a aplicação da *string* de busca na base ACM, não promoveu a localização de artigos.

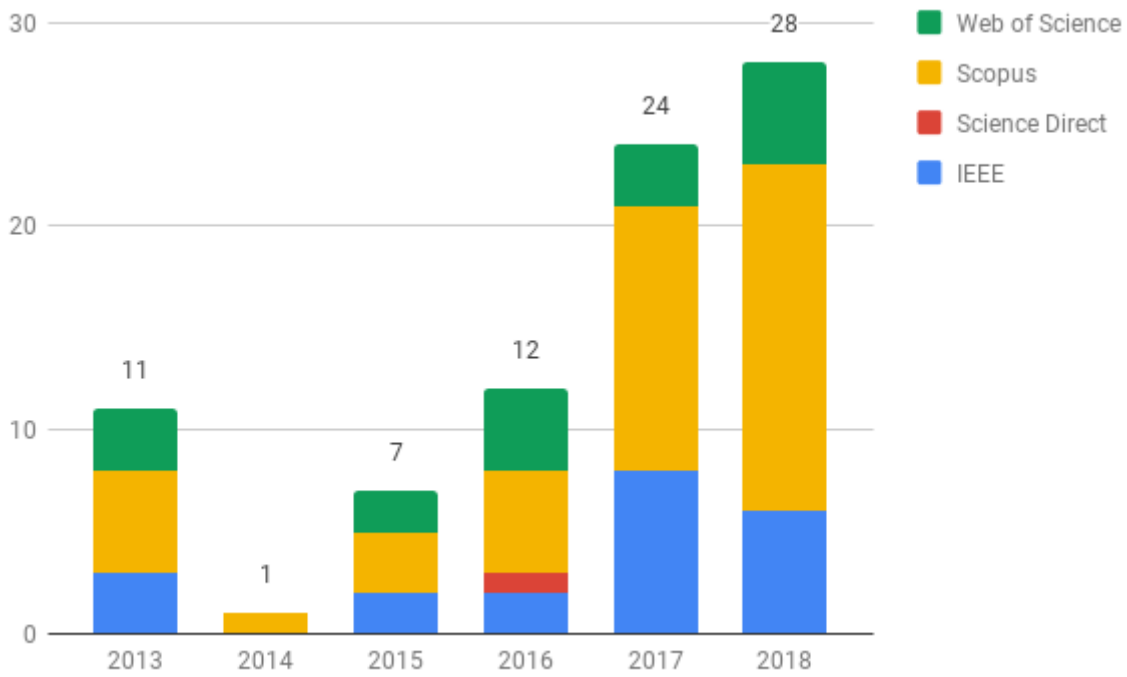

Figura 6 – Número de Artigos Publicados por Ano em cada Base Considerada

Após a pesquisa nas bases científicas, foram aplicados os seguintes critérios de Inclusão (I) e Exclusão (E), sobre os artigos retornados:

- (E) Artigos que não exploram Dataflow;
- (E) Artigo prévio de um artigo mais novo;
- (E) Documentos que não são artigos de conferência ou periódicos;
- (E) Artigos que não estão em inglês;
- (I) Artigos que utilizam Dataflow para processamento de contexto.

A Tabela 1 apresenta a *string* de busca com sua sintaxe ajustada para cada base acadêmica, bem como o número de artigos retornados.

Tabela 1 – Aplicação da String de Busca nas Bases Acadêmicas

Base	String	Artigos
IEEE	((("fog computing") OR ("Internet of Things") OR ("edge computing")) AND ("dataflow"))	21
Science Direct	TITLE-ABSTR-KEY ((("fog computing") OR ("Internet of Things") OR ("edge computing")) AND ("dataflow"))	1
Scopus	((("fog computing") OR ("Internet of Things") OR ("edge computing")) AND ("dataflow"))	44
Web of Science	((("fog computing") OR ("Internet of Things") OR ("edge computing")) AND ("dataflow"))	17

Após o emprego da string na *engine* de busca das bases acadêmicas, foi realizada a exportação do resultado para o formato BibTex e sua importação para a ferramenta StArt. Um total de 83 artigos foram identificados. A Tabela 2 apresenta o número de artigos incluídos ou excluídos, de acordo com os critérios apresentados.

Os documentos apresentados na Tabela 2 foram identificados como norteadores das análises posteriormente realizadas. Sendo assim, como resultado da revisão sistemática da literatura foram selecionados 5 artigos, os quais são apresentados na Seção 3.2.

3.2 Trabalhos Selecionados

Esta Seção tem o objetivo descrever os trabalhos selecionados pelo processo de revisão sistemática. São apresentados os objetivos da proposta de cada trabalho junto a uma análise crítica dos mesmos.

Tabela 2 – Total de Artigos por Critério de Inclusão (I) ou Exclusão (E)

Critério	Número de Artigos
(E) Artigos que não exploram dataflow	27
(E) Artigos que explorar dataflow para aquisição de dados	7
(E) Artigos prévio de artigo mais novo	2
(E) Documentos que não são artigos de conferência ou periódicos	6
(E) Artigo que não estão em inglês	1
(E) Artigo duplicado	34
(I) Artigos que utilizam dataflow para processamento de contexto	5

3.2.1 A Web-Based Approach Using Reactive Programming for Complex Event Processing in Internet of Things Applications

O artigo (ZIMMERLE; GAMA, 2018) apresentas um trabalho inicial para o desenvolvimento de componentes que permitem o uso de padrões para *Complex Event Processing* (CEP) nas plataformas de desenvolvimento *Web of Things* (WoT). A pesquisa, denominada NR-CEP, desenvolveu uma extensão de CEP para a plataforma Node-RED, a qual consiste de uma ferramenta de programação visual direcionada para a IoT, prototipada sobre Node.js.

A NR-CEP adicionou a capacidade de integrar os operadores de CEP aos dados associados a diferentes fluxos na plataforma Node-RED, os eventos de alto nível criados visualmente e configurados pelo usuário na ferramenta são disparados no momento que as informações chegam nos diversos fluxos de dados.

Os autores no seu esforço de estudo e pesquisa realizam um mapeamento sistematizado as diferentes funcionalidades de CEP na perspectiva da IoT e identificaram que algumas operações apresentam alguma sobreposição, contemplando a mesma proposta geral (ZIMMERLE; GAMA, 2018).

Desta forma, foram selecionadas três funcionalidades para serem implementadas junto ao Node-RED, as quais são apresentadas na Figura 7.

O trabalho desenvolveu os três conjuntos de funcionalidades que estendem a plataforma Node-RED para CEP: (i) *aggregation*; (ii) *join*; e (iii) *pattern matching*. Dessa forma, a arquitetura baseada proposta permitiu colocar a manipulação das informações que chegam em componentes de *mashup* reutilizáveis, os quais irão dar suporte para o processamento de eventos complexos em fluxos de dados da WoT (ZIMMERLE; GAMA, 2018).

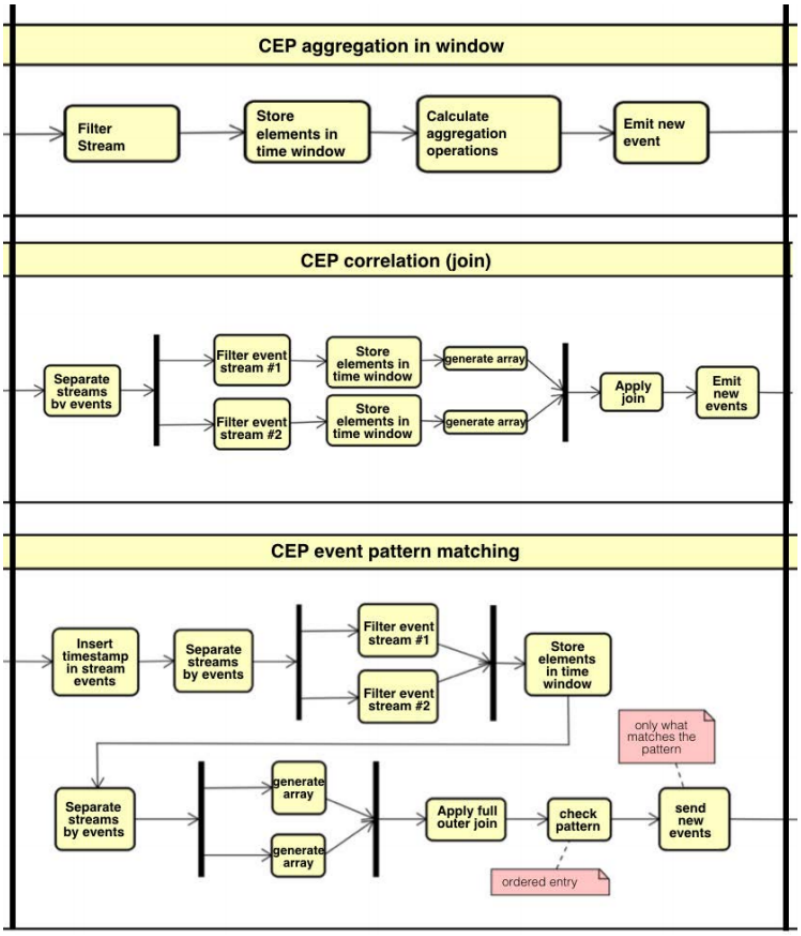

Figura 7 – Visão Geral da Plataforma NR-CEP
Fonte: (ZIMMERLE; GAMA, 2018)

3.2.2 Appdaptivity: An Internet of Things Device-decoupled System for Portable Applications in Changing Contexts

No artigo (MARTÍN et al., 2018) os autores apresentam o *Appdaptivity*, uma plataforma que permite o desenvolvimento de aplicativos que podem ser adaptados a diversos contextos. Com o emprego do *Appdaptivity*, os desenvolvedores dispõem de recursos para criarem aplicações portáveis e personalizadas.

Considerando os objetivos contemplados no trabalho, o desenvolvimento empregando o *Appdaptivity* concentra-se na lógica do aplicativo, e não na concepção de aplicações para dispositivos específicos, permitindo um desenvolvimento independente dos equipamentos disponíveis da infraestrutura computacional. Os dispositivos típicos da IoT, que são o foco da arquitetura do *Appdaptivity*, muitas vezes contém recursos limitados para o processamento, portanto um padrão leve de comunicação foi adotado. O *Appdaptivity* adotou o CoAP como protocolo de comunicação para interagir com os dispositivos finais, os quais usualmente lidam com sensores e atuadores (MARTÍN et al., 2018).

O trabalho integra o *Portability Core* (PoCo), sendo o componente responsável

pela programação do fluxo de dados na interface Web. O PoCo permite o desenvolvimento dos aplicativos baseados em localização, que podem ser dinamicamente portáveis para diferentes hardwares. Para isto, oferece uma variedade de implementações considerando diferentes casos de uso, a lógica da aplicação concebe a integração de diferentes fluxo de dados, enquanto os aplicações são um conjunto de clientes que recebem e enviam dados com as informações necessárias para o processamento do dados. Uma visão geral da arquitetura do Appdaptivity, pode ser vista na Figura 8.

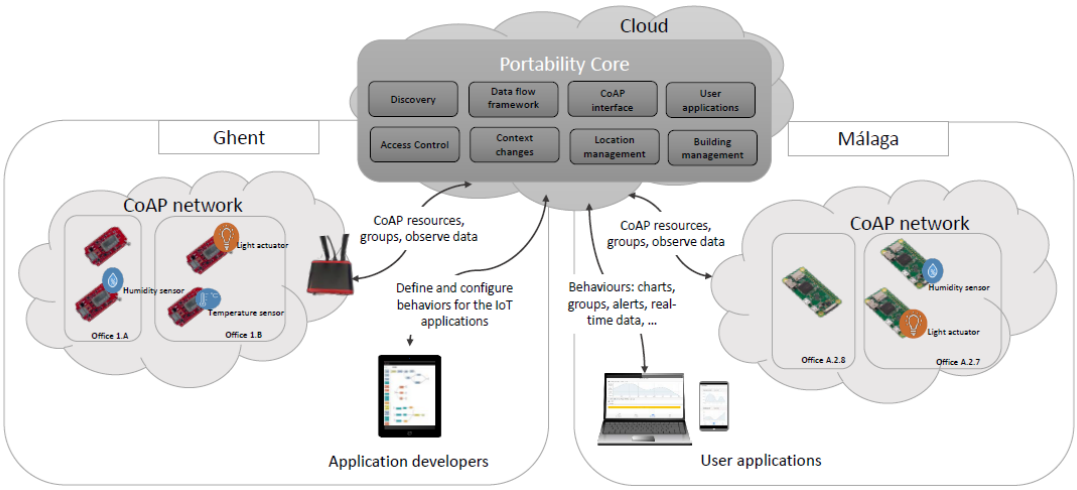

Figura 8 – Visão Geral da Plataforma Adaptability
Fonte: (MARTÍN et al., 2018)

A portabilidade e o acesso dos aplicativos resultantes são diferentes para cada caso de uso. A arquitetura proposta para o *Appdaptivity* foi projetada para contemplar diferentes casos de uso e permite três formatos:

- *Cloud Deployment*: consiste de uma implementação em nuvem, permitindo a portabilidade de aplicativos e o acesso de qualquer lugar que disponha de uma conexão com a Internet. Os dispositivos para *Resource Directory* (RD) devem estar localizados na rede local e serem acessíveis por IP para permitir a descoberta e interação, respectivamente.

- *Local Deployment*: neste formato os aplicativos de usuário são por concepção implantados em diferentes dispositivos. O Appdaptivity será executado em um servidor dedicado, mas na mesma rede que os aplicativos do usuário. A portabilidade do aplicativo pode ser feita com as redes CoAP disponíveis na rede local e a descoberta de RDs pode ser realizada neste tanto pelos aplicativos do usuário quanto pela arquitetura de software associada ao Appdaptivity.

- *Embedded Deployment*: os aplicativos do usuário e a arquitetura de software do Appdaptivity, neste formato, são implantados no mesmo dispositivo. Isso

permite uma solução portátil e incorporada que pode ser aplicada a áreas controladas, por exemplo, casas inteligentes controladas pelos proprietários.

O trabalho explorou um modelo de programação baseado em fluxo de dados para definir intuitivamente as aplicações dos usuários, os quais definem os fluxos por meio de um gráfico direcionado de nós, modelando um fluxo de ações a partir dos dados.

Os nós têm funções definidas que não dependem de outros nós, assim, formando componentes portáveis e reutilizáveis nas aplicações. Assim, os desenvolvedores prioritariamente precisam se concentrar na lógica das aplicações, uma vez que do ponto de vista da prototipação o *Appdaptivity* fornece um grande conjunto de nós para serem utilizados na definição das aplicações dos usuários.

3.2.3 ECHO: An Adaptive Orchestration Platform for Hybrid Dataflows Across Cloud and Edge

No artigo (RAVINDRA et al., 2017) os autores propõem uma plataforma adaptável para gerenciar fluxos de dados entre recursos distribuídos, bem como a gerência do ciclo de vida da aplicação. Para tanto, é previsto o emprego da abstração de *containers* e um registro dinâmico para o controle de estado das aplicações. A plataforma ECHO possibilita a troca de mensagens entre os diversos níveis da aplicação, dentre eles: (i) *Edge*; (ii) *Fog Computing*; e (iii) *Cloud Computing*.

A Figura 9 apresenta uma visão geral da arquitetura concebida para a plataforma ECHO. Os dispositivos identificados na cor amarela, constituem os seguintes módulos:

- *Device Service*: executado como parte da infraestrutura da aplicação;
- *Platform Service*: executado em um *contêiner* ou uma Máquina Virtual e realiza a interface com uma instância local do Apache NIFI[2], que é utilizado como mecanismo *Dataflow* para tratamento do fluxo de dados;
- *Resouce Directory* e *Platform Master Service*: formam os principais serviços da plataforma, usualmente hospedados em uma Máquina Virtual disponibilizada de forma pública em uma Nuvem Computacional (*Cloud VM*). Os dispositivos, seus *conteiners* e a própria Cloud VM, são registrados no *Resouce Directory* podendo ser gerenciados para dar suporte à execução de fluxos de dados dos usuários.

[2]https://nifi.apache.org/

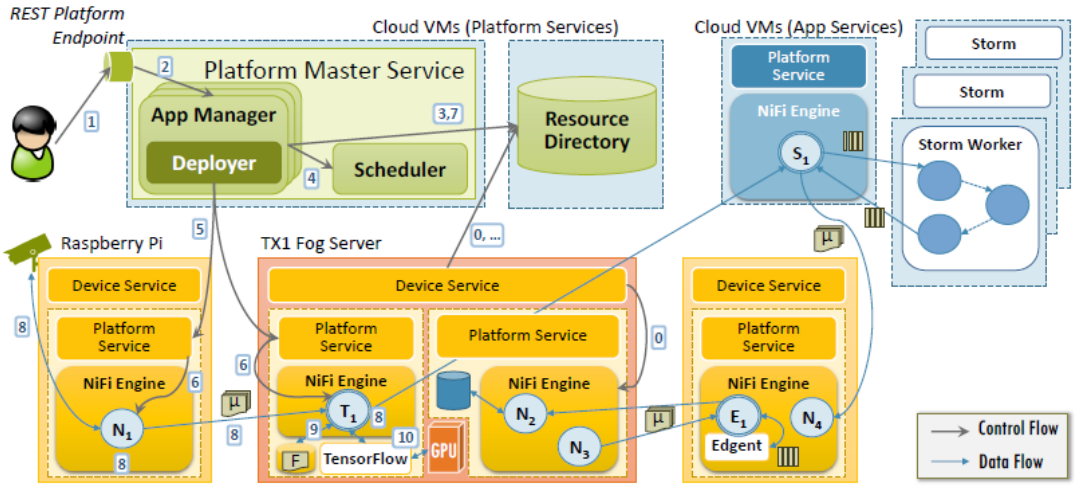

Figura 9 – Visão Geral da Plataforma ECHO
Fonte: (RAVINDRA et al., 2017)

O *Master*, identificado na cor azul, é responsável por gerenciar o fluxo de dados da aplicação. Os módulos que o compõem, podem ser observados a seguir:

- *Resouce Directory*: este módulo armazena o estado do sistema, sendo utilizado para registrar os recursos da IoT e os seus fluxos de dados. Para isto, é utilizado o padrão *Hypercat* 3.0 BSI é baseado em JSON leve para os ativos da IoT, onde cada dispositivo registrado é identificado por uma URI. Com o emprego do *Hypercat* o recurso pode disponibilizar seus registros (POST), bem como aceitar consultas (GET), operações estas baseadas no padrão REST.

- *Device Service*: é um serviço de infraestrutura da arquitetura, que monitora os dispositivos e os *conteiners* que são gerados. O serviço disponibiliza uma API REST que pode ser usada para lançar novos contêineres, com imagens de aplicativos específicos e desativar os contêineres não utilizados. Este componente também registra a utilização da CPU e da memória de cada dispositivo em seus contêineres, isso caracteriza a capacidade do dispositivo e também o desempenho de aplicativos em seus contêineres.

- *Platform Master and Dataflow Lifecycle*: é um serviço REST responsável por gerenciar o ciclo de vida de um fluxo de dados para o usuário, usando outros componentes da arquitetura ECHO. O serviço oferece três ações principais: (i) iniciar um fluxo de dados, (ii) pará-lo e (iii) reequilibrá-lo dinamicamente. Estas ações podem realizar outros procedimentos: pausar, alterar parâmetros de entrada ou mesmo modificar a estrutura do fluxo de dados.

- *Platform Service and Distributed Orchestration*: este módulo é responsável por gerenciar os *conteiners* que hospedam a aplicação que executa um serviço

para gerenciar a orquestração dos fluxos de dados. Dependendo da disponibilidade de recursos e do compartilhamento permitido entre diferentes fluxos de dados, cada contêiner pode executar todo ou partes de um, ou mais, fluxos de dados.

Resumidamente, o trabalho propôs uma plataforma de *middleware* para compor dados considerando os recursos da *Cloud Computing*, *Fog Computing* e *Edge*. O ECHO é uma plataforma orientada a serviços que atende os requisitos da gerência dos recursos presentes nas diversas camadas da IoT.

3.2.4 Fog at the Edge: Experiences Building an Edge Computing Platform

O artigo (GIANG et al., 2018) apresenta uma plataforma de Computação de Borda chamada *Distributed Node-RED* (DNR), que utiliza o modelo de programação de fluxo de dados distribuído, baseado na plataforma Node-RED. O emprego do Node-RED faculta que a programação possa ser baseada em um paradigma visual.

O trabalho considera como oportuno que a concepção das aplicações explore a computação nos níveis de *Cloud*, *Fog* e *Edge*. Para tanto, o trabalho considerou que os dispositivos de borda seriam acessíveis por IPs válidos e que possuiriam recursos computacionais suficientes para executar as tarefas em intervalos de tempo oportunos para as aplicações.

O trabalho explorou o uso do Node-Red para o desenvolvimento das aplicações utilizadas pelos nodos, bem como para implementar a sua distribuição pela rede. Para o suporte dessas operações, foram implementadas duas funcionalidades principais: (i) *Remote Wires*; e (ii) *Remote Arcs*. Estas funcionalidades são centrais para a proposta, pois a computação pode acontecer em diversos dispositivos, distribuindo o processamento, bem como o fluxo de dados deve poder ser interrompido a qualquer momento.

Deste modo, como os nodos podem ser executados em diferentes dispositivos, os Node-Red *Wires* necessitam que exista suporte para comunicação entre dispositivos para lidar com a situação em que um fluxo de dados é interrompido. Para tratar isto foi adotada uma estratégia de comunicação baseada em *publish/subscribe* para promover a interoperação dos nós.

Neste sentido, os dispositivos realizam a conexão em um *broker* para que possam trocar dados com outros diapositivos.

A Figura 10 apresenta o processo de suporte à implantação distribuída de um fluxo de dados em vários dispositivos. De modo geral, pode-se dizer que a arquitetura do DNR abstrai os detalhes de comunicação, facilitando o desenvolvimento das aplicações orientadas a *Fog Computing*.

O trabalho apresentou uma plataforma que permite aos desenvolvedores criarem aplicações IoT distribuídas que abordem diversos dispositivos. O desenvolvimento

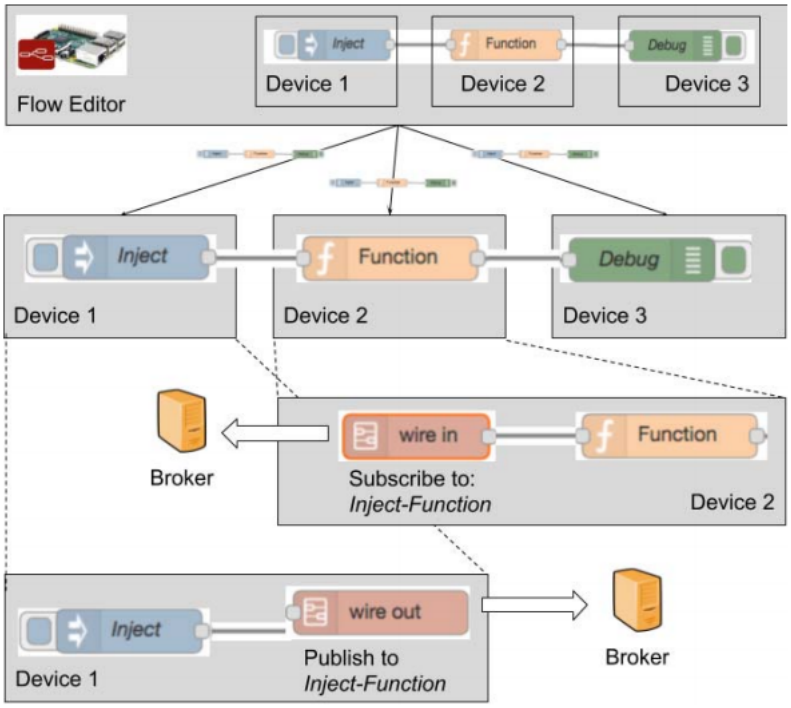

Figura 10 – Visão Geral da Plataforma DNR
Fonte: (GIANG et al., 2018)

da aplicação fornece um modelo de aplicativo reutilizável e escalável, pois ocorre à separação explícita das atividades de comunicação e computação.

3.2.5 Mechanism for context-aware substitution of Smart-M3 agents based on dataflow network model

O artigo (VASILEV et al., 2013) estudou a confiabilidade das redes de fluxo de dados na plataforma Smart-M3. Esta plataforma é um *middleware* de código aberto para criação de *Smart Space Applications*, provendo o recurso de substituição de agentes comprometidos.

O trabalho desenvolveu um novo mecanismo para a substituição de agentes, que permite a transferência do processamento de dados do agente comprometido para um substituto. Dessa forma, este mecanismo desenvolvido permite preservar o contexto nas situações de desconexão de um agente em específico, estas situações são comuns em ambientes de IoT.

O Smart-M3 fornece infraestrutura para a criação de aplicações onipresentes de acordo com o paradigma de espaços inteligentes. O elemento central da plataforma, *Semantic Information Broker* (SIB), permite aos agentes de aplicação interagirem por meio do compartilhamento de informações.

Os dados armazenados no SBI estão de acordo com o modelo *Resource Description Framework*. No nível inferior, o agente manipula dados inserindo, excluindo, atualizando e consultando no SIB. O SIB pode notificar o agente sobre o evento de

modificação de dados, isto se o agente estiver inscrito na alteração de dados correspondente.

A implementação do modelo de rede de fluxo de dados na plataforma Smart-M3 consiste em agentes que seguem regras. Os agentes utilizam assinaturas para recuperar as informações. Quando um agente externo altera o elemento monitorado, o SIB envia a notificação para todos os inscritos. O agente modificado processa e atualiza as informações.

O trabalho apresentou pesquisa sobre o aumento da confiabilidade dos recursos presentes no ambiente da IoT. Para isto foi desenvolvido um método que permite substituir temporariamente agentes desconectado por outros, até o momento que o agente original reconecte-se à rede. Para que essa substituição ocorra de forma efetiva, o agente ativo recebe o contexto computacional do agente comprometido, o que torna esse processo não perceptível para o usuário.

3.3 Considerações Sobre o Capítulo

Neste Capítulo estão registrados os esforços de sistematização das características dos trabalhos técnico-científicos cujas tecnologias exploradas se interseccionam com aquelas entendidas como oportunas para o EXEHDA-FC.

Como abordagem para identificação destes trabalhos foi empregada uma revisão sistemática de literatura. O emprego deste mecanismo além de sistematizar o processo, facultando sua reprodução no LUPS no momento do retorno desta frente de pesquisa no Grupo de Pesquisa, também teve por premissa entender o perfil das publicações em veículos consolidados, envolvendo tópicos entendidos como de interesse para a concepção do EXEHDA-FC.

Neste sentido, como consequência da área de *Fog Computing* estar em franca disseminação quando da realização dos esforços de estudo e pesquisa do EXEHDA-FC, a busca dos Trabalhos Relacionados levou a identificação de esforços de pesquisa com diferentes objetivos no que diz respeito aos cenários de uso e/ou perfis de aplicações explorados.

Esta situação naturalmente dificultou a categorização destes trabalhos quanto aos métodos e/ou algoritmos computacionais empregados, mas entretanto permitiu identificar que algumas abordagens podiam ser entendidas como uma tendência na área.

Dentre estas abordagens destacaríamos duas prioritariamente: (i) o emprego do Node-RED como ferramenta para implementação a do processamento *Dataflow*; e (ii) a tendência de empregar mais de um nível para organizar a gerência e o processamento de informações em infraestruturas distribuídas.

O esforço para realização da revisão sistemática de literatura, bem como o estudo

dos Trabalhos Relacionados contemplados, foi central para a concepção do EXEHDA-FC apresentada no Capítulo 4, na qual uma hierarquia de dois níveis arquiteturais é prevista, com a flexibilização do processamento nas bordas poder acontecer de modo cooperativo entre os diferentes Servidores de Borda do middleware EXEHDA que estão comprometidos em atender uma determinada aplicação em particular.

4 EXEHDA-FC: ARQUITETURA E FUNCIONALIDADES

Neste capítulo são discutidas as premissas de concepção, bem como as funcionalidades previstas para o EXEHDA-FC. Dentre as quais destacaríamos o suporte para emprego de *Fog Computing* nos equipamentos responsáveis pelo gerenciamento da interação com o meio no *middleware* EXEHDA

4.1 Contribuições Realizadas no Middleware EXEHDA

No EXEHDA, os eventos de contexto são proativamente monitorados e o suporte à execução deve permitir que tanto as aplicações, como o próprio *middleware*, utilizem as informações na gerência de seus aspectos funcionais e não-funcionais.

A arquitetura do *middleware* EXEHDA é apresentada na Figura 3, divide-se em três camadas: (i) camada inferior, responsável pela gerencia da infraestrutura computacional (máquina virtual, sistema operacional e hardware); (ii) camada intermediária, parte responsável pelas funcionalidades de mais alto nível do *middleware*; e a camada superior, formada pelo *framework* destinado a prover suporte à programação das aplicações ubíquas na IoT.

A abordagem proposta nesta Dissertação tem por foco contribuir especificamente com o Subsistema de Adaptação e Reconhecimento do Contexto, ampliando suas funcionalidades. Esse módulo tem por finalidade gerenciar os contextos de interesse das aplicações e do próprio ambiente de execução.

As premissas de concepção do EXEHDA-FC incorporam as características do mecanismo para Ciência de Contexto adotado no EXEHDA, o qual emprega uma estratégia colaborativa entre a aplicação e o ambiente de execução. O EXEHDA-FC têm por finalidade qualificar o *middleware* EXEHDA, mais especificamente em contribuir com o Subsistema de Adaptação e Reconhecimento de Contexto, de maneira a qualificar o atendimento às demandas inerentes às modernas infraestruturas computacionais da IoT, incorporando neste cenário suporte a *Fog Computing* e *Dataflow* combinados.

Sendo assim, o EXEHDA-FC foi concebido empregando os seguintes pressupostos, todos comprometidos em reduzir a latência na tomada de decisões com base

nas informações contextuais e também aumentar a confiabilidade operacional do *middleware*

- Processamento distribuído de eventos: por meio de políticas de publicação e subscrição os dispositivos de borda podem se comunicar e processar os eventos no momento em que eles são gerados, auxiliando na distribuição necessária para a operação em regime de *Fog Computing*;
- Serviço de diretório: tem por finalidade prover suporte as ações de subscrição e publicação dos equipamentos de borda. Um diretório com as informações dos recursos de sensoriamento e atuação será mantido atualizado na arquitetura a partir dos próprios equipamentos de borda, a medida que novos sensores e atuadores são instanciados e/ou removidos;
- Operação de borda desconectada: mesmo com a perda da conexão com o Servidor de Contexto via Internet (conexão de longa distância), os equipamentos de borda, operando em rede local, devem manter os procedimentos de coleta de dados contextuais, bem como o processamento de regras de contingência locais, disparando os respectivos procedimentos de atuação previstos;
- Eventos da arquitetura tratados por meio de API: os diferentes eventos inerentes a operação do EXEHDA-FC, entre eles a coleta de informações sensoreadas, subscrições, publicação, etc., devem ser tratados por uma interface de programação (API) que permita uma manipulação em mais alto nível dos mesmos.

Considerando estes pressupostos, foram concebidos os módulos que compõem o EXEHDA-FC. Suas particularidades arquiteturais e funcionais estão descritas nas próximas Seções.

4.2 Servidor de Contexto

O Servidor de Contexto concentra as funcionalidades de processamento e comunicação necessárias para operação do Subsistema de Adaptação e Reconhecimento de Contexto.

No EXEHDA-FC, é mantida a organização prevista para o Servidor de Contexto, apresentada na Seção 2.5, permanecendo as funcionalidades organizadas por meio dos componentes arquiteturais que podem ser observados na Figura 11.

Toda comunicação com os *Gateways* é gerenciada pelos Servidores de Borda, deste modo os diferentes sensores a atuadores conectados aos *Gateways* sob a gerência de um determinado Servidor de Borda, se mostram ao Servidor de Contexto associados ao mesmo.

4.2.1 Configurador

O como acontece com outros trabalhos envolvendo o EXEHDA componente Configurador concentra todos os procedimentos de configuração necessários para o funcionamento do EXEHDA-FC (FILHO, 2019) (TABIM, 2018).

As configurações inerentes ao Servidor de Contexto, que são locais, permanecem acontecendo de modo análogo aos outros trabalhos do grupo de pesquisa. O procedimento novo diz respeito ao repasse de configurações para os Servidores de Borda de modo remoto, utilizando o componente Interoperador, como será discutido na continuidade.

4.2.2 Processador de Contexto

O componente Processador de Contexto é responsável por realizar inferências a partir das informações sensoriadas pela infraestrutura computacional de borda. Os valores sensoreados gerenciados pelos Servidores de Borda, quando enviados ao Servidor de Contexto, são processados por regras definidas pelo usuário.

Esta submissão de dados no Servidor de Contexto gera eventos, cuja ação primária é o disparo das regras de processamento contextual. Estas regras utilizam o modelo Evento-Condição-Ação (ECA) e utilizam para sua especificação o modelo *Dataflow* de programação, sendo possível contar com o auxílio de uma interface gráfica para sua escrita (GROUP, 2018).

Com a finalidade de prover suporte para o processamento de históricos de informações contextuais, bem como para viabilizar a visualização das informações ao longo do tempo por parte dos usuários, todas as informações submetidas ao Servidor de Contexto são persistidas em um Banco de Dados, denominado Repositório de Informações Contextuais (RIC).

4.2.3 Interoperador

O componente Interoperador, considerando a natureza inerentemente distribuída das aplicações alvo do EXEHDA-FC, foi concebido para gerenciar a troca de informações entre os Servidores de Borda e de Contexto considerando redes de diferentes naturezas.

Este componente também provê suporte às comunicações entre os serviços do *middleware* eventualmente em execução em equipamentos diferentes, como também às aplicações, oferecendo para isso uma interface padronizada e bem definida.

A concepção deste componente teve como foco realizar a substituição do padrão arquitetural *Representational State Transfer* (REST) empregado em trabalhos anteriores do EXEHDA, o qual utiliza o protocolo HTTP e suas operações (GET, PUT, POST, DELETE) no suporte às trocas de informações. Com isso, a interoperação ocorria por meio de operações HTTP sobre URIs, as quais constituíam o caminho para o acesso

a todos os recursos disponibilizados pela arquitetura.

Para viabilizar a troca de informações entre dispositivos localizados em redes diferentes e protegidas por *firewalls*, as quais não permitem que ocorram conexões a partir de dispositivos externos, foi necessário trocar o padrão REST de comunicação, baseado em HTTP, utilizado pelos Servidores de Contexto e Borda.

O componente Interoperador teve uma nova concepção com o objetivo de utilizar como protocolo físico de comunicação o *Message Queue Telemetry Transport* (MQTT), com isto, ficaram viabilizadas comunicações assíncronas entre o Servidor de Contexto e os Servidores de Borda, com a iniciativa de interoperação partindo de qualquer um dos lados.

Este componente se vale de um *Broker* MQTT de acesso aberto na Internet, no qual os Servidores de Borda e Contexto, realizam a subscrição em tópicos que são criados dinamicamente de acordo com a identificação (UUID) dos Servidores de Borda envolvidos.

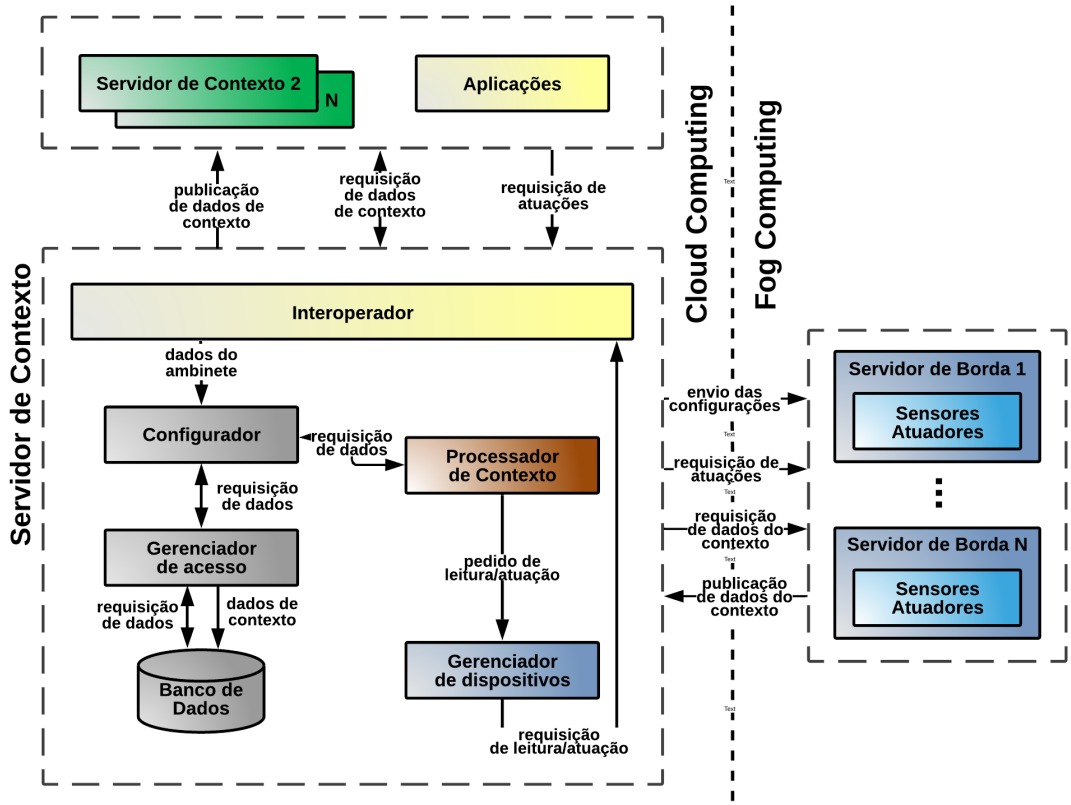

Figura 11 – Arquitetura do Servidor de Contexto
Fonte: O Autor

4.2.4 Especificação da Concepção das Regras de Borda

O Servidor de Contexto oferece uma aplicação que permite a concepção das regras de borda. Estas regras especificas são criadas na interface *Dataflow* do Node-

Red, pode ser observado na Figura 12, o qual o usuário pode selecionar sensores, adicionar condições associados a valores. Além disso, é possível combinar sensores, associados a operadores lógicos, entre eles *and* e *or*. Por fim, com a condição ou condições criadas, o usuário pode criar a ação de seu interesse.

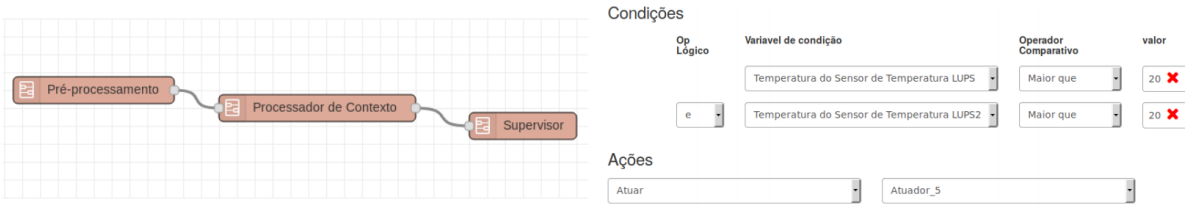

Figura 12 – Interface Dataflow para Criação de Regras de Borda
Fonte: O Autor

A regra de exemplo apresentada na Figura 12 , é descrita para o seguinte código na Figura 13.Toda regra é associada a um sensor no Banco de Dados, e quando um dado de um sensor chega no componente Processador, o mesmo verifica a regra associada ao sensor e aos realiza a execução. Essa regra armazenada é lida e executada como um arquivo Python. Nesse caso, é realizado uma combinação de dois sensores para posteriormente que a ação possa ser executada quando necessária.

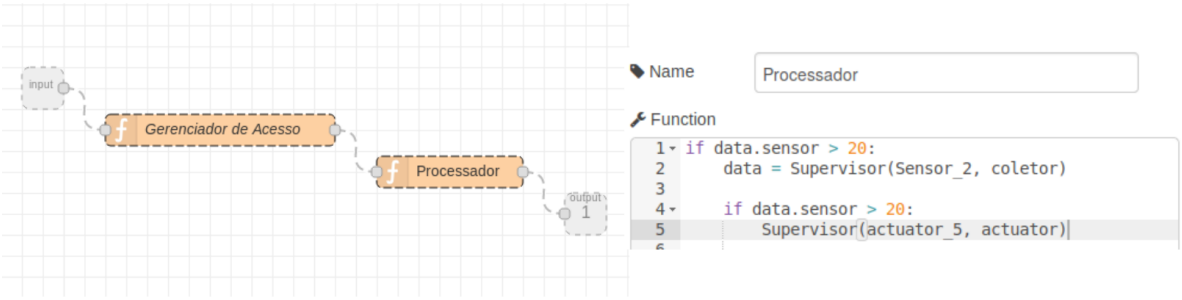

Figura 13 – Instanciação de Processamento Dataflow na Borda
Fonte: O Autor

4.3 Servidor de Borda

No EXEHDA-FC, a arquitetura adotada no desenvolvimento do Servidor de Borda é formada por componentes. Cada componente foi desenvolvido para ter uma funcionalidade exclusiva, quando o componente necessita de recursos disponíveis em outros, o mesmo requisita as informações ou envia os dados. A Figura 14 apresenta a arquitetura de software do Servidor de Borda desenvolvido, ilustrando os componentes que o compõem, bem como a transição dos dados.

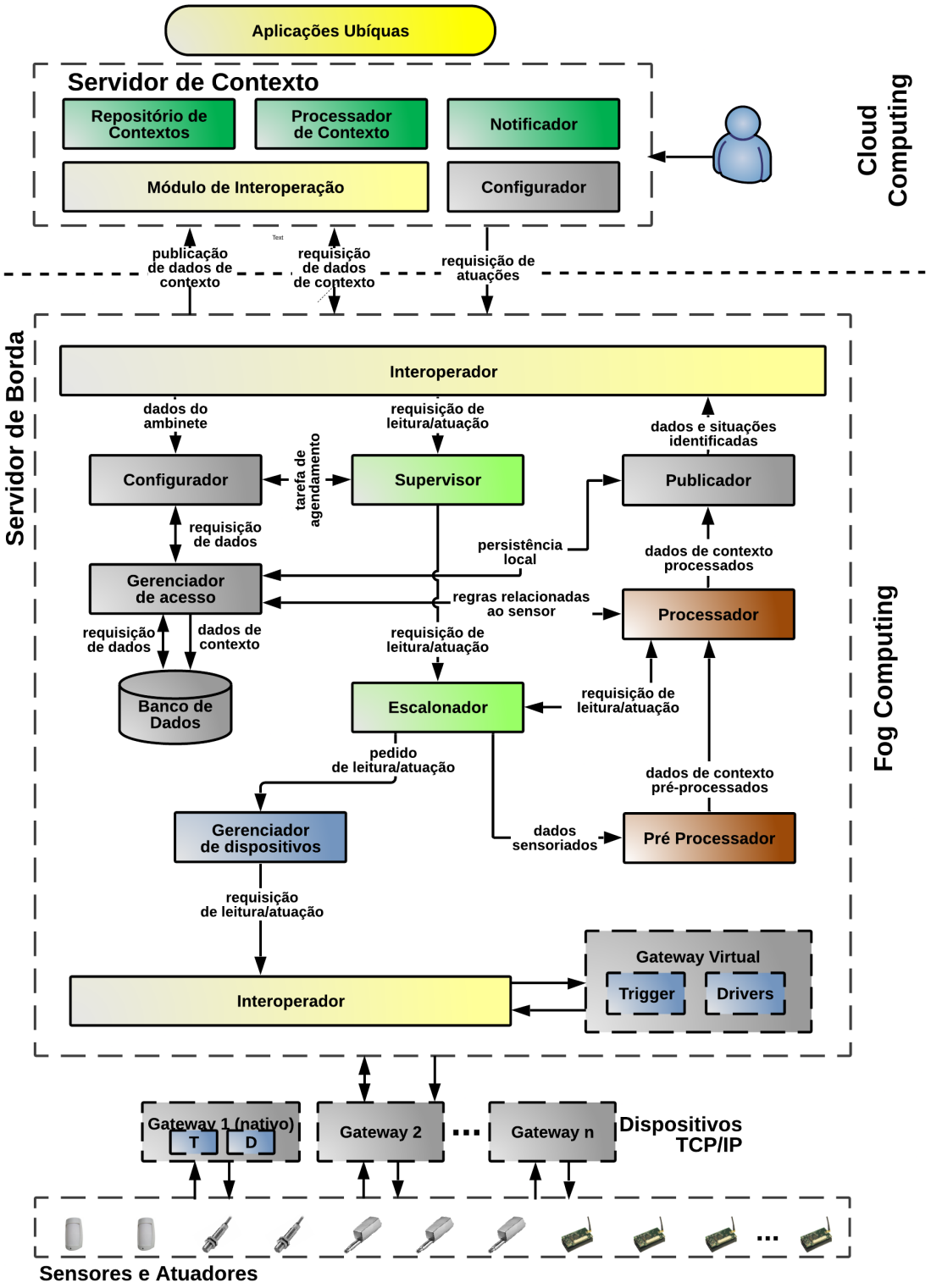

Figura 14 – Arquitetura do Servidor de Borda

Fonte: O Autor

4.3.1 Interoperador

Na perspectiva de atender o perfil distribuído das aplicações do EXEHDA-FC, o componente Interoperador segue os mesmos princípios de concepção do Interoperado do Servidor de Contexto, substituindo a arquitetura REST e suas URIs, pelo protocolo de comunicação MQTT. O padrão arquitetural REST que foi substituído tem suas particularidades disponíveis em (João et al., 2018).

Este componente foi concebido para gerenciar a troca de informações com os diferentes dispositivos presentes na infraestrutura computacional gerenciada pelo EXEHDA-FC, particularmente destacaríamos a interoperação com o Servidor de Contexto, Gateways e mesmo outros Servidores de Borda. É possível observar na Figura 14, que o Interoperador comunicando tanto com o Servidor de Contexto, como com os Gateways. Sua visualização em dois pontos distintos da Figura é somente para maior clareza do seu perfil operacional.

4.3.2 Supervisor

O componente Supervisor foi projetado para atender as demandas de agendamento de eventos periódicos. Como exemplos típicos de eventos periódicos temos aqueles relacionados a leitura de sensores e a ativação de atuadores.

Após avaliação das diferentes alternativas existentes, com o intuito de flexibilizar os perfis de agendamento foi selecionada uma biblioteca que suporte o padrão Crontab/POSIX[1] de agendamento de eventos.

A biblioteca utilizada é implementada em Python, e denominada *Advanced Python Scheduler* (APScheduler) (APSCHEDULER, 2020). O *APScheduler* permite o cadastro de eventos, permitindo que os mesmos sejam executados periodicamente ou mesmo apenas uma vez. Também é facultada a adição de novas tarefas a qualquer momento e/ou a gerência de tarefas já cadastradas.

Os eventos são definidos pelo usuário utilizando o componente configurador do Servidor de Contexto e posteriormente enviados ao Servidor de Borda.

O formato da mensagem a ser transferida pode ser visualizada no *Listing* 4.1. O *type* recebe como argumento *APSscheduler*, tendo ficado definido que este argumento refere-se a uma tarefa de agendamento para os Servidores de Borda. Os argumentos do campo *mode*, foram desenvolvidos considerando recursos da biblioteca *APScheduler*, oferecendo aos Servidores de Borda três modos de agendamento, como apresentado a seguir:

- *Cron-style scheduling*: é o modo mais completo, possui todos requisitos do *CRON* e funcionalidades de adicionar horário de início e de fim nas atuações periódicas das tarefas;

[1] http://crontab.org/

- *Interval-based execution*: executa as tarefas com intervalos regulares, com a opção de adicionar horário de início e término;
- *One-off delayed execution*: executa uma única tarefa, determinada por data e horário específicos.

Listing 4.1 – Mensagem de cadastro de agendamento enviada pelo Servidor de Contexto

```
{
  "type"=> "type",
  "mode"=> "mode",
  "task"=> "task",
  "event"=> "event",
  "minute"=> "minute",
  "hour"=> "hour",
  "day"=> "day",
  "month"=> "month",
  "year"=> "year",
  "second"=> "second",
}
```

As especificações de agendamento das tarefas são armazenadas no banco de dados local do Servidor de Borda, pois pode ocorrer queda de energia ou até mesmo uma reinicialização do sistema para eventuais atualizações. Quando o Servidor de Borda está inicializando, é lido do banco de dados as configurações dos eventos cadastrados. A Figura 15 apresenta o fluxo de dados pertinente a realização de um agendamento em um Servidor de Borda.

4.3.3 Configurador

O componente Configurador aglutina os procedimentos de configuração necessários para o funcionamento do Servidor de Borda. Quando ocorre uma inicialização do Servidor de Borda é realizada a leitura do JSON apresentado no *Listing* 4.2.

Alguns parâmetros são necessários para permitir as conexões nos *brokers*, no Servidor de Contexto e Servidor de Borda. Após a leitura dos dados, é inicializada a conexão em ambos *brokers*.

Neste JSON estão especificados os parâmetros para conexão na rede Wi-Fi, bem como nos Brokers MQTT. Um Broker utilizado para o Servidor de Borda trocar mensagens com os Gateways e outro para trocar mensagens com o Servidor de contexto.

Após etapa de conexão são anunciados ao Servidor de Contexto os recursos de sensoriamento e atuação presentes no Banco de Dados do Servidor de Borda.

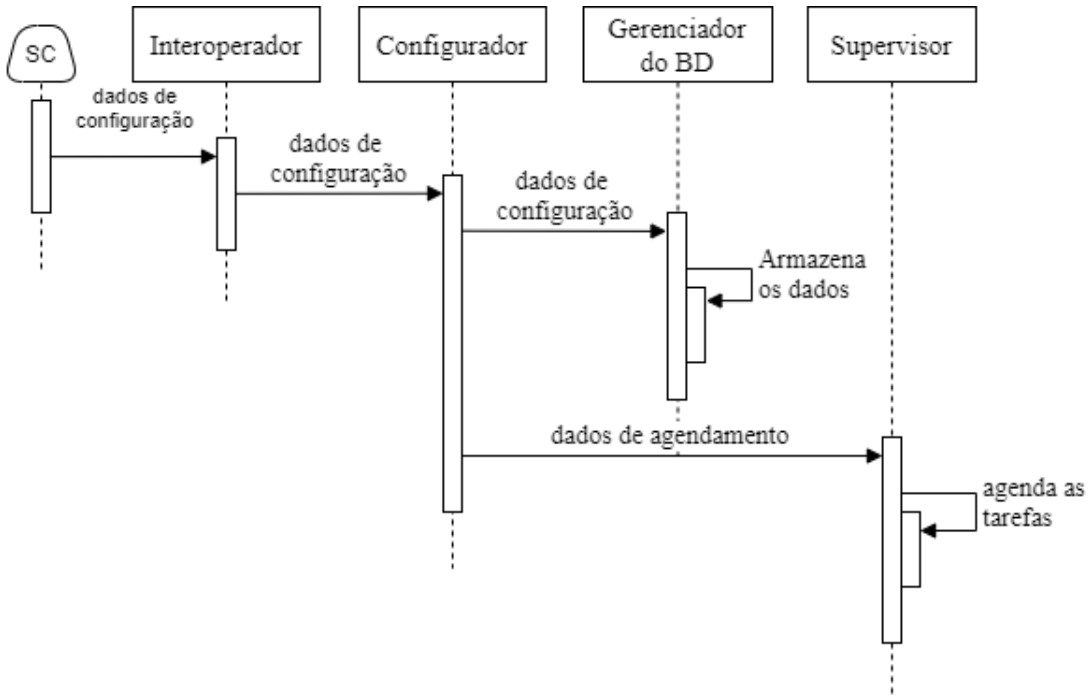

Figura 15 – Fluxo de Dados do Agendamento de Eventos em um Servidor de Borda
Fonte: O Autor

Após o Servidor de Contexto ter o conhecimento de um determinado Servidor de Borda e seus recursos, uma terceira etapa é inicializada. Nesta etapa, todos os agendamentos de eventos cadastrados são inicializados.

Por fim, após estabilizada a etapa de configurações, ainda no procedimento de reinício, é disparada uma rotina que verifica a Tabela *Persistence* do Banco de Dados do Servidor de Borda, na qual são registrados os valores sensoreados. Em existindo conteúdos nesta tabela, os mesmo são transferidos ao Servidor de Contexto antes do Servidor de Borda entrar em regime de operação.

Listing 4.2 – JSON de Configuração Carregado na Inicialização do Servidor de Borda

```
{
  "type": "type",
    "version": "version",
    "network":{
      "name": "network_name",
      "password": "network_password",
        },
        "edge_server":{
      "name": "edge_server_name",
      "uuid": "edge_server_uuid",
    },
    "broker_mqtt_ES":{
```

```
      "ip": "broker_mqtt_ES_ip",
      "port": "broker_mqtt_ES_port",
      "user": "broker_mqtt_ES_user",
      "password": "broker_mqtt_ES_password",
      "topic": "broker_mqtt_ES_topic",
    },
    "broker_mqtt_CS":{
      "ip": "broker_mqtt_CS_ip",
      "port": "broker_mqtt_CS_port",
      "user": "broker_mqtt_CS_user",
      "password": "broker_mqtt_CS_password",
      "topic": "broker_mqtt_CS_topic",
    },
    "topics": ["topics"],
}
```

4.3.4 Escalonador

O componente Escalonador foi projetado para atender as demandas da arquitetura da forma mais genérica possível:

As ações previstas para serem disparadas pelo Escalonador são as seguintes:

- (i) agendamento de dispositivo, os dados recebidos são enviados ao componente Supervisor;
- (ii) coleta ou atuação em um dispositivo, a tarefa é disparada pelo componente Supervisor e enviados os dados para o componente Gerenciador de Dispositivos;
- (iii) processamento de uma regra agendada, a tarefa é disparada pelo componente Supervisor;
- (iv) anúncio de recursos ao Servidor de Contexto, toda e qualquer momento que o Servidor de Borda inicializar, os dados de cadastrados no Banco de Dados são enviados ao Servidor de Contexto;
- (v) anúncio de recursos recebidos pelo gateway, ocorre quando os gateways estão anunciando os recursos disponíveis ao Servidor de Borda, armazenando estes dados no Banco de Dados local;
- (vi) publicação, os dados de contexto provenientes dos sensores são armazenados localmente e enviados ao Servidor de Contexto.

Estas ações são recebidas pelo Escalonador a partir de diferentes componentes da arquitetura do Servidor de Borda, na forma de eventos.

4.3.5 Gerenciador de Acesso

O componente Gerenciador de Acesso implementa as funcionalidades de acesso ao Banco de Dados local ao Servidor de Contexto, bem como as requisições dos dados e o armazenamento das informações. O Banco de Dados foi modelado com o objetivo de armazenar as informações contextuais típicas do cenário da IoT, sendo formadas prioritariamente por strings, valores numéricos inteiros e de ponto flutuante. Para isto, foi utilizado a tecnologia apresentada na Seção 4.5.3, o Banco de Dados SQLite.

4.3.6 Gerenciador de Dispositivos

O componente Gerenciador de Dispositivo foi modelado para abstrair os comandos para requisições de coleta de dados e/ou atuações.

As requisições são realizadas por meio do componente Interoperador, enviando aos Gateways uma requisição de dado contextual pelo emprego da mensagem *Listing* 4.3. Por sua vez, quando do disparo de uma atuação ambiente, a mensagem recebe outro campo (´device_status´) como pode ser observado no *Listing* 4.4.

Listing 4.3 – Formato da mensagem de requisição de dados contextuais

```
{
  "collect_uid": "collect_uid",
  "device_uid": "device_uid",
  "requested_data": "requested_data",
}
```

Listing 4.4 – Formato da mensagem de atuação do atuadores no ambiente

```
{
  "collect_uid": "collect_uid",
  "device_uid": "device_uid",
  "device_status" "device_status",
  "requested_data": "requested_data",
}
```

4.3.7 Pré Processador

O componente Pré Processador foi desenvolvido para realizar a filtragem dos dados antes de serem enviados para o componente Processador. Sendo assim, alguns

dados podem ser descartados. Um exemplo neste sentido, seriam dados coletados cuja variação for menor que a precisão do sensor.

4.3.8 Processador

O componente Processador foi concebido para atender as demandas de processamentos de dados contextuais nos Servidores de Borda. As regras adotadas são do tipo ECA e armazenadas no Banco de Dados local do Servidor de Borda.

Estas regras são criadas no Servidor de Contexto pelo usuário, e devem ser construídas considerando o fato de que os Servidores de Borda geralmente são alocados fisicamente próximos ao ambiente sendo sensoreado, permitindo uma atuação bastante confiável sobre o meio.

4.3.9 Publicador

O componente Publicador tem como princípio efetuar a publicação do dado contextual no Servidor de Contexto. O formato da mensagem enviada pode ser observada no *Listing* 4.5.

Este componente é responsável por coordenar o principal fluxo de dados entre os Servidores de Borda e de Contexto, promovendo a publicação de todos os dados coletados e garantindo uma persistência local dos mesmos nos períodos em que a publicação ficar inviabilizada.

Neste sentido, toda publicação a ser feita, antes é persistida, sendo removido o dado do Banco de Dados quando do sucesso da publicação no Servidor de Contexto. A Figura 16 sumariza o fluxo de dados trocado entre os Servidores de Borda e Contexto.

Listing 4.5 – Formato da Mensagem de Publicação dos Dados Contextuais

```
{
  "gateway_uuid": "gateway_uuid",
  "device_uui": "device_uui",
  "value" "value_collect",
  "collect": "collect_date",
}
```

4.4 Gateway

O Gateway é responsável pelas funcionalidades de coleta das informações sensoreadas e de atuação, bem como abstrai os protocolos físicos de comunicações com os sensores e atuadores, repassando para os Servidores de Borda uma informação padronizada.

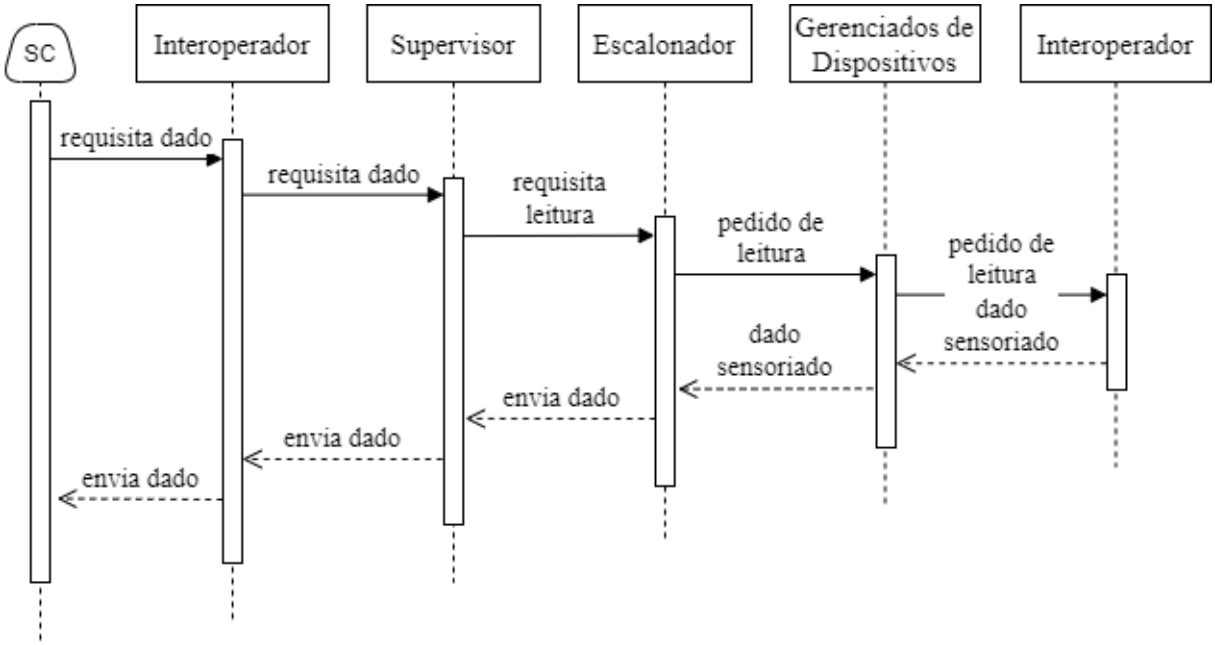

Figura 16 – Fluxo da Publicação de Informações Contextuais
Fonte: O Autor

A arquitetura concebida para o Gateway do EXEHDA-FC pode ser observada na Figura 17, e na continuidade são caracterizados os seus componentes.

4.4.1 Interoperador

O componente Interoperador do Gateway foi projetado considerando as mesmas premissas empregadas nos Servidores de Borda e Contexto.

Neste sentido, também é empregado como protocolo físico para as comunicações o MQTT, sobre o qual é realizada troca de informações empregando arquivos JSON.

4.4.2 Configurador

Este componente é responsável por tratar o arquivo de configuração do Gateway quando da sua inicialização. Este arquivo pode ser observado no *Listing* 4.6.

O primeiro campo, *type* é definido como *configuration*, especificando que este arquivo contém dados para a inicialização do Gateway.

O campo *device*, por sua vez, é composto pela lista dos sensores e/ou atuadores disponíveis no Gateway, registrando diversos argumentos utilizados pela arquitetura do EXEHDA-FC para o relacionamento dos dados sensoreados com informações sobre a infraestrutura computacional envolvida.

Por último, estão indicados os dados de identificação do Gateway e do Servidor de Borda envolvidos.

Listing 4.6 – Arquivo de Configuração do Gateway em JSON

```
1 {
```

```
  {
  "type": "type",
  "device": [{
    "status": "device_status",
    "manufacturer": "device_manufacturer",
    "uuid": "device_uuid",
    "name": "device_name",
    "pin": ["pin_conected"],
    "driver": "drive_name",
    "type": "type",
  },...{...}],
  "gateway": {
    "name": "gateway_name",
    "uuid": "gateway_uuid",
  },
  "date": "date_created",
  "edge": {
    "uuid": "edge_server_uuid",
  }
 }
}
```

4.4.3 Publicador

O componente Publicador tem a funcionalidade de enviar os dados obtidos dos sensores ao Servidor de Borda. O formato da mensagem pode ser observado no Listing 4.7.

Listing 4.7 – Formato da mensagem de publicação dos dados contextuais no Servidor de Borda

```
{
  "device_uuid": "device_uuid",
  "value": "value_collect",
  "collect_date": "collect_date"
}
```

4.4.4 Comunicador

Quando um Servidor de Borda requisita um dado contextual do ambiente, o Gateway carrega os drives relacionados aos sensores e realiza a leitura, após a coleta, os dados são enviados ao componente Publicador.

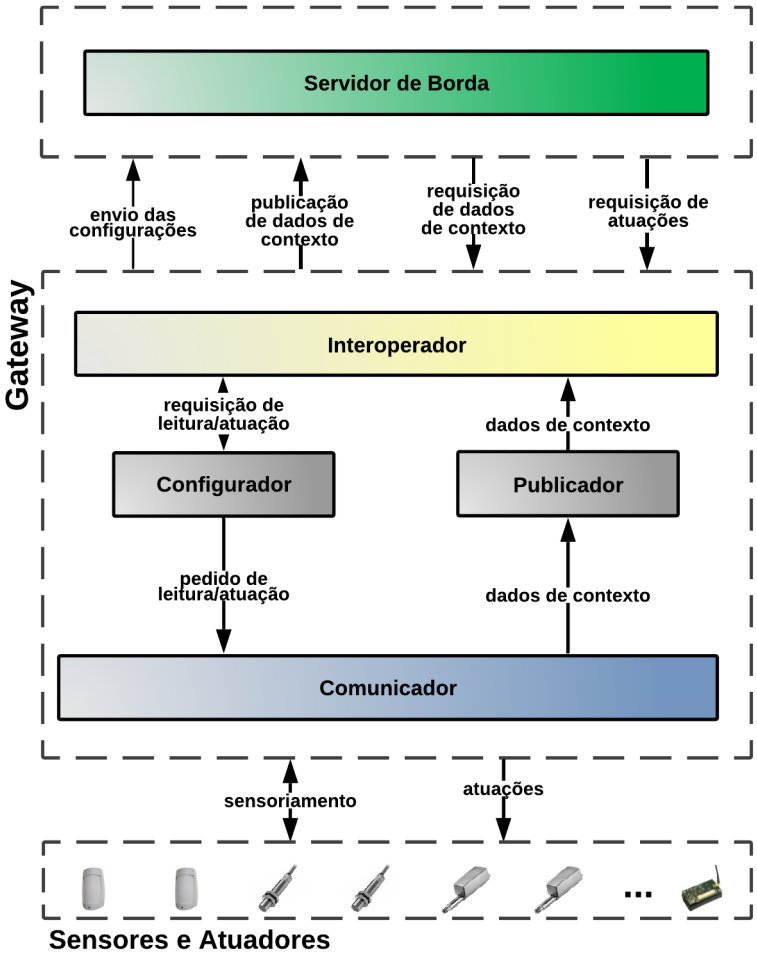

Figura 17 – Arquitetura do Gateway

Fonte: O Autor

4.5 Tecnologias de Software Empregadas

Nesta Seção são descritas as principais tecnologias utilizadas, considerando as premissas operacionais do EXEHDA-FC. As tecnologias são associadas aos componentes da arquitetura e apresentada as principais motivações para o seu emprego.

As tecnologias consideradas para emprego no EXEHDA-FC, foram selecionadas a partir dos seguintes critérios: (i) modernidade; (ii) sua adoção no cenário nacional e internacional; e (iii) serem open-source e por fim sua adequação ao cenário da IoT.

4.5.1 Ferramenta de Programação Node-RED

O Node-RED é uma ferramenta de Programação *Dataflow*, desenvolvida para "conectar" dispositivos distribuídos, suportando diversos protocolos e APIs. Originalmente foi desenvolvida pela equipe de Serviços de Tecnologia Emergentes da International Business Machines (IBM), e após por *JS Foundation* (NODE-RED, 2018). Foi proposto por J. Paul Morrison na década de 1970, com o objetivo que o usuário descreva o comportamento de uma aplicação por meio de caixas pretas, ou "nós".

Esta ferramenta destaca-se por fornecer uma interface de fácil manuseio, permitindo a construção e edição empregando um navegador Web. A aplicação é executada em *JavaScript* (JS) usando a estrutura Node.js, o qual propicia a construção dos dos fluxos de dados em tempo de execução assíncrono e orientado a eventos. Uma visão típica da interface do Node-RED, em um navegador web, pode ser observada na Figura 18.

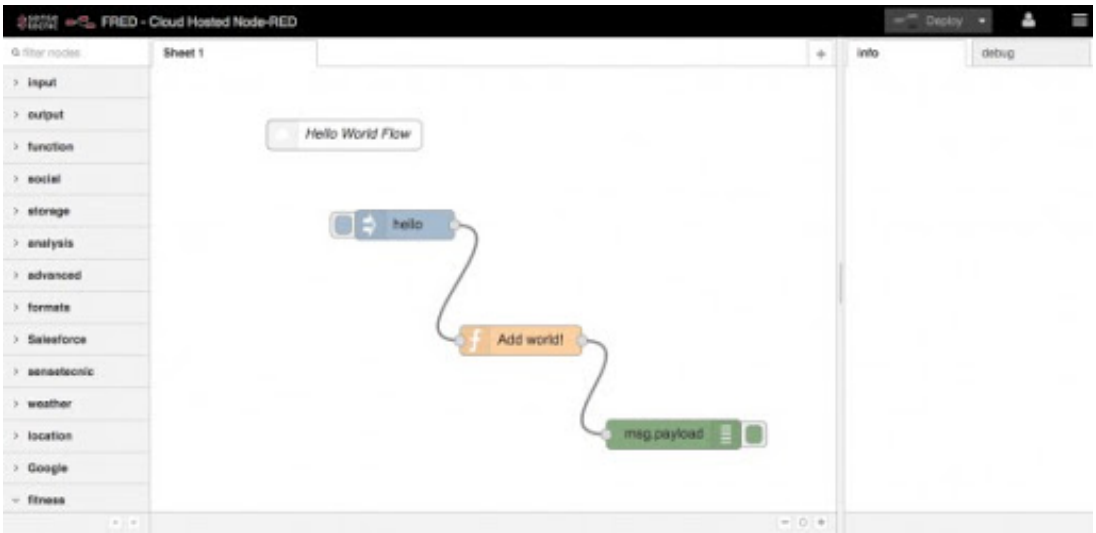

Figura 18 – Ambiente de Desenvolvimento do Node-RED.
Fonte: (NODE-RED, 2018)

Consiste em conjunto de nós conectados arcos orientados, como em outros sistemas de fluxo de dados visuais. A interface do usuário consiste em um editor de fluxo visual no qual os modelos de nós, representando diferentes nós de entrada, saída e

processamento de dados, podem ser inseridos na tela e conectados uns aos outros.

Foi desenvolvido inicialmente para a utilização em IoT, mas seu uso pode ser expandido para outras áreas devido a sua flexibilidade. No repositório de pacotes da ferramenta pode-se encontrar módulos com a capacidade de integração com o sistema de ônibus de Londres, atualização de coordenadas no mapa do Google em tempo real, relatórios de terremotos, entre outros (RAMME, 2015).

Existem diversos módulos nativos que suprem boa parte das necessidades do desenvolvimento das aplicações, sem a necessidade de criar novos blocos, entre eles: (i) leitura e escrita de mensagens MQTT, HTTP, websocket, TCP, UDP; (ii) envio e recebimento de e-mail e tweets; (iii) processamento de dados em formato CSV, JSON, HTML e XML; (iv) processamento de funções em JavaScript; (v) árvore de decisão conforme mensagem de entrada; e (vi) “análise de sentimento” baseada nas palavras de um determinado texto de entrada.

Cada nó é composto por uma entrada, e uma ou mais saídas. O dado é recebido, processado pela função determinada pelo usuário e após repassada a uma ou mais saídas e consequentemente aos próximos nós conectados. Existem nós que podem ter sua entrada proveniente de outros softwares que podem estar em outros servidores. O mesmo vale para os nós de saída, que podem enviar mensagens para equipamentos remotos, que possam estar de posse do usuário.

A Figura 19 apresenta um conjunto de nós disponibilizados no Node-RED, onde o usuário pode utiliza-los para desenvolver as aplicações, sendo os mesmos assíncronos e paralelos com orientação a eventos. Há diversas possibilidades para a criação destas aplicações com os nós disponíveis, como exemplos de funcionalidades podemos enumerar: conexão com Twitter, Facebook, Telegram, protocolos de rede (UDP, TCP, HTTP, MQTT, etc.), conexão local serial, conexão com banco de dados SQL e Não-SQL como MongoDB, mySQL, etc., painel para visualização de dados com diversos recursos (gráficos, tabelas, acionadores, etc.), envio de e-mails, conexão com Chatbots, processamento de áudio e vídeo, conexão com o sistema de arquivos, análise de dados com recursos simples e complexos, implementação de funções em Javascript, Python e outras linguagens, funcionalidades específicas para Raspberry Pi e Arduino, conversão de texto em áudio e vice-versa, ou seja, existe um potencial de utilização para vários recursos com fácil integração e conectividade em uma interface simples, rápida e acessível através de um navegador.

(NODE-RED, 2018)

O diferencial do Node-RED na criação de *frameworks* para uso na IoT é sua facilidade de manipulação dos fluxos dos dados, inclusive em muitos casos não existindo necessidade de modificação dos vários nós utilizados.

Figura 19 – Visão Parcial dos Nós do Node-RED.
Fonte: (NODE-RED, 2018)

4.5.2 Protocolo de Comunicação MQTT

O MQTT é um protocolo de rede leve usado para publicação e assinatura de mensagens entre dispositivos (YASSEIN et al., 2017). Este protocolo foi desenvolvido pela IBM em 1999 e seu propósito principal é atuar com dispositivos restritos em termos de poder computacional e memória, redes com baixa largura de banda e alta latência. Essas características caracterizam o MQTT como um protocolo excelente para o cenário de IoT (YASSEIN et al., 2017).

Diferentemente do modelo cliente-servidor, o protocolo MQTT é baseado no modelo *publish-subscribe*. A Figura 20 caracteriza o funcionamento do protocolo, onde três clientes estão conectados a um Broker. O Broker é responsável por receber os dados e orquestar a quais repassa aos dispositivos, e os clientes podem conectar-se de duas formas, *publish* e *subscribe*. O *publish* gera o conteúdo e envia ao broker, sendo responsável por transmitir estes dados aos clientes que realizaram o *subscribe*.

Quando um cliente publica dados, os mesmos são associados a tópicos específicos. Estes tópicos são identificados como uma hierarquia, utilizando uma

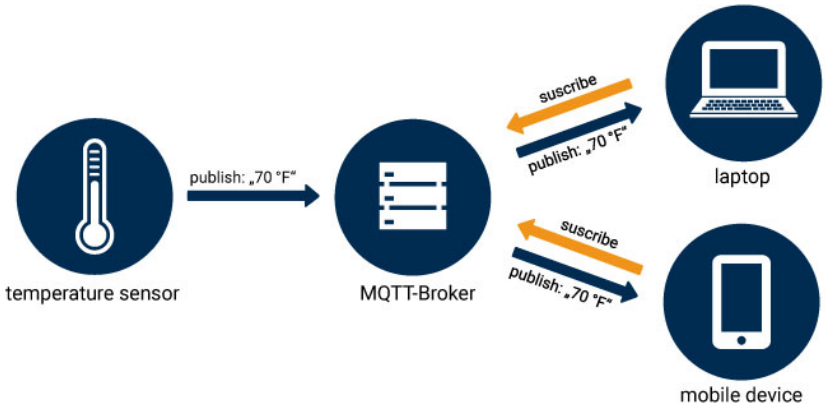

Figura 20 – Relação do Broker MQTT com seus Publicadores e Assinantes.
Fonte: (YASSEIN et al., 2017)

barra (/) como separador dos níveis. Por exemplo, vários clientes podem publicar suas informações de temperatura ou estado no seguinte tópico, devices/<DEVICE_ID>/sensor/<SENSOR_ID>/temperature.

Os clientes ao realizarem o *subscribe*, podem receber as mensagens respectivas as assinaturas de um tópico específico, ou até mesmo, todos as mensagens de um determinado nível inserindo curingas no tópico escolhido. Estes curingas podem ser de dois tipos:

- +: é utilizado para obter todas as informações da hierarquia específica. Neste caso, devices/<*DEVICE*_ID>/*sensor*/+/*temperature*, recebe todos os dados de temperatura de um determinado *device*.
- #: é utilizado como caractere final de uma assinatura, recebendo todos os dados do tópico, sem a necessidade de especificar todos o caminho. Neste caso, *devices*/#, são recebidas todas as informações de todos os *devices*.

No EXEHDA-FC, o MQTT é utilizado como protocolo físico de comunicação para as trocas de informações na arquitetura desenvolvida, contribuindo sobremaneira para a troca de dados entre equipamentos bastante heterogêneos.

4.5.3 Banco de Dados SQLite

O SQLite é um banco de dados *Open Source*, que implementa recursos de mecanismo de banco de dados SQL transacional, *zero-configuration*, *serverless* e *self-contained*. O SQLite foi originado de um projeto para Marinha dos EUA (YU, 2020).

Ao contrário de outros bancos de dados SQL, o SQLite não possui um processo de servidor separado, ou seja, lê e grava diretamente em arquivos de disco comuns. Um banco de dados SQL completo com várias tabelas, índices, gatilhos e visualizações está contido em um único arquivo de disco. O formato do arquivo de banco de dados é multiplataforma entre sistemas de 32 e 64 bits (SQLITE, 2020).

O banco de dados SQLite possui as seguintes características (SQLITE, 2020):

- Gerenciamento simples e estável: é composto por um arquivo único, facilitando o gerenciamento. A funcionalidade do SQLite pode ser gerenciada de duas maneiras: (i) em tempo de compilação; e (ii) instruções PRAGMA, configuração em tempo de execução.
- Escalabilidade e usabilidade: é executado no mesmo endereço de memória, ou seja, o driver pysqlite do SQLite da biblioteca padrão do Python ou o driver opcional aspw, fornece APIs que definem funções SQL personalizadas, funções de agregação e clássicas.
- Agilidade no acesso: por ser executado no mesmo endereço de memória, não possui protocolo de linha, serialização e não necessita de comunicação por meio de soquetes UNIX.

A base de código do SQLite é mantida por uma equipe internacional de desenvolvedores, que trabalham para expandir os recursos e aprimorar a confiabilidade e desempenho, ao mesmo tempo em que provê suporte para compatibilidade com versões anteriores e com a sintaxe SQL.

4.6 Considerações sobre o Capítulo

Este Capítulo discorreu sobre a concepção do EXEHDA-FC, apresentando uma discussão dos seus aspectos arquiteturais, relacionando o impacto dos mesmos nas funcionalidades previstas.

Também foram apresentas as tecnologias selecionadas para a prototipação do EXEHDA-FC. A descrição das tecnologias contemplou os aspectos com maior significado para concepção do EXEHDA-FC, particularmente aqueles atinentes aos aspectos relacionados a integração dos componentes arquiteturais, os quais além de distribuídos contemplam uma elevada heterogeneidade.

5 EXEHDA-FC: CENÁRIO DE USO

Nesta Seção está apresentado o cenário de uso desenvolvido com o objetivo de explorar a arquitetura proposta para o EXEHDA-FC. As funcionalidades tratadas no estudo de caso contemplam tarefas relacionadas a aquisição e processamento de dados contextuais distribuídos, bem como a gerência das atuações no ambiente provido pelo EXEHDA-FC.

Este cenário de uso foi direcionado a Viticultura de Precisão (VP), uma extensão do trabalho desenvolvido em (João et al., 2018). Tratando as possíveis contribuições com a utilização da *Fog Computing*. A VP constitui uma área de exploração agrícola que possui um grande potencial econômico e social em diversas regiões do Brasil e em particular do Rio Grande do Sul. A viticultura tem promovido investimentos significativos em pesquisa e tecnologias direcionadas a gerência dinâmica e autônoma dos seus parreirais e das áreas de produção agrícola associadas.

5.1 Infraestrutura Computacional Empregada

Nesta Seção estão caracterizados os principais componentes da arquitetura de software do EXEHDA-FC empregados no cenário de uso, apresentando o comportamento dos mesmos ante o problema atacado e os respectivos volumes de dados transferidos em decorrência da interoperabilidade praticada.

5.1.1 Prototipação do Gateway no EXEHDA-FC

No EXEHDA-FC, os Gateways constituem os dispositvos com maior escalabilidade e para tanto são construídos empregando dispositivos com boa relação custo/benefício.

Para a seleção de um embarcado para uso na concepção dos Gateways, foram considerados os seguintes aspectos: (i) baixo consumo energético; (ii) suporte a comunicação Wi-Fi; e (iii) suporte ao emprego de uma plataforma de software disseminada. Particularmente o critério de empregar em uma plataforma de software disseminada para sua gerência e programação, tem como motivação central supor-

tar o maior número possível de sensores e atuadores, bem como prover uma curva mais acelerada de aprendizado pelos novos integrantes do grupo de pesquisa. Após uma revisão de literatura a escolha recaiu sobre *System on Chip* (SoC) ESP32 (MAIER; SHARP; VAGAPOV, 2017).

A ESP32 é dotada de um microprocessador *dual-core*, possuindo comunicação WI-FI integrada, bluetooth 4.2, 36 pinos para comunicação com sensores, atuadores, dispositivos de armazenamento, entre outros periféricos.

Enquanto linguagem de programação, foi buscada uma opção cuja aprendizagem fosse facilitada e ao mesmo tempo já fosse utilizada pelo Grupo de Pesquisa (LUPS) em outras frentes de estudo e pesquisa. Neste sentido foi selecionada a linguagem MicroPython[1], a qual incorpora um conjunto bastante rico de bibliotecas bem como suporte a *multithreading*.

5.1.2 Prototipação do Servidor de Borda no EXEHDA-FC

O Servidor de Borda concebido para o EXEHDA-FC foi prototipado tendo como base o sistema embarcado Raspberry PI Modelo B+, empregando como software básico o sistema operacional Raspian[2]

A Raspberry Pi é um embarcado que tem como principais características o tamanho reduzido, baixo custo, baixo consumo energético, com comunidade envolvida bastante ativa e que baseia-se nos princípios de software livre (RASPBERRY, 2019).

Considerando estes aspectos, a utilização da Raspberry Pi como Servidor de Borda se mostra bastante alinhada ao cenário da IoT. Por outro lado, este embarcado possui boa capacidade de processamento, o que oferece suporte para que as funcionalidades do EXEHDA-FC possam ser implementadas. Dentre outras características, destacaríamos que a Raspberry Pi além de prover compatibilidade com diferentes protocolos de comunicação, oferece suporte à conexão física de dispositivos para sensoriamento e/ou atuação de diferentes naturezas.

As funcionalidades do Servidor de Borda foram implementadas na Raspberry Pi B+ utilizando a linguagem de programação Python associada a diversas tecnologias de software, dentre as quais destacamos as indicadas na Seção 4.5.

5.1.3 Tecnologia para Sensoriamento Padrão 1-Wire

Para atender as demandas de sensoriamento do EXEHDA-FC, foi prevista a utilização da tecnologia 1-Wire como protocolo físico de acesso ao hardware utilizado nos diferentes sensores e atuadores. O protocolo de comunicação 1-Wire, foi desenvolvido pela empresa *Dallas Semiconductor*, o qual baseia-se em uma rede de transmissão de dados de baixo custo que possibilita uma comunicação digital em

[1] http://micropython.org/
[2] https://www.raspberrypi.org/downloads/

barramento oferecendo suporte ao controle de erros.

Na proposta 1-Wire, o dispositivo que coordenada as interoperações e gerencia o endereçamento dos diferentes dispositivos envolvidos é denominado *master*, sendo os outros dispositivos da série 1-Wire, tais como sensores e atuadores, denominados de (*slaves*) (COMMONOPENRESEARCHEMULATOR, CORE).

A tecnologia 1-Wire destaca-se por ser um protocolo para troca de dados em rede de consumo energético baixo, baseado em dispositivos eletrônicos endereçáveis, apresentando uma boa robustez operacional, mesmo não exigindo cabeamento especializado para sua instalação.

A interoperação com os dispositivos de sensoriamento e/ou atuação é efetivada nos Gateways do EXEHDA-FC por meio de *drivers*. Este software é específico por tipo de dispositivo, sendo responsável por traduzir os dados transferidos pelo protocolo 1-Wire em uma informação numérica tratável e compatível com a grandeza física mensurada e/ou procedimento de atuação a ser realizado. Neste site[3] mantido pelo LUPS estão apresentados alguns dos trabalhos envolvendo a tecnologia 1-Wire que foram desenvolvidos pelo Grupo de Pesquisa.

5.1.4 Prototipação do Servidor de Contexto do EXEHDA-FC

A prototipação do Servidor de Contexto do EXEHDA-FC foi realizada em um nodo da infraestrutura de nuvem computacional da UFPEL, que dispõe de um processador Intel(R) Xeon(R) CPU E5-2660, operando na frequência de 2.20GHz e uma memória de 4 Gb de RAM.

As principais contribuições feitas por esta Dissertação de Mestrado ao Servidor de Contexto do middleware EXEHDA estão associadas as necessárias alterações no protocolo de comunicações para interação com o Servidor de Borda, bem como aquelas necessárias para o interfaceamento com o Node-RED enquanto estratégia para o processamento de informações contextuais.

Na implementação do Servidor de Contexto utilizado no EXEHDA-FC foram incorporadas diversas contribuições do grupo de pesquisa, dentre as quais destacaríamos as feitas mais recentemente em (TABIM, 2018) e (FILHO, 2019). Durante o desenvolvimento destes trabalhos, enquanto consequência da concomitância dos trabalhos em grupo no LUPS, as discussões referentes ao EXEHDA-FC foram consideradas pelas agendas de pesquisa de todos os envolvidos.

Do ponto de vista do desenvolvimento de software, as modificações no Servidor de Contexto, para suporte as funcionalidades do EXEHDA-FC foram concebidas utilizando a linguagem Python. Por sua vez, a aplicação Web para gerência e visualização dos dados armazenados no Servidor de Contexto o LUPS vem utilizando a linguagem PHP em sua versão 5.5, juntamente com o *framework* de desenvolvimento *CodeIg-*

[3] http://ubiq.inf.ufpel.edu.br/1-wire/

niter.

O CodeIgniter[4] é um *framework* PHP para desenvolvimento de aplicações Web que utiliza a arquitetura do padrão MVC (Model-View-Controller) para o desenvolvimento rápido e padronizado de suas aplicações. Seu funcionamento divide a aplicação em três tipos de componentes:

- Model: componente reservado para lógica e funções da aplicação;
- View: componente utilizado para representação de dados para o usuário;
- Controller: componente direcionado para a interoperação entre os dois componentes anteriores.

Este padrão utilizado pelo *framework* facilita a reusabilidade de código e a separação dos conceitos de programação empregados.

Após os necessários ajustes para que o banco de dados atendesse as expectativas das estratégias concebidas para o EXEHDA-FC, foram realizadas modificações na aplicação Web pela construção dos *Models*, os quais fazem o acesso ao banco de dados para captura, inserção, remoção e atualização de informações. Para cada tabela do banco de dados foi criado um *Model* o qual possui diversos métodos para requisições específicas, evitando assim que um dado seja tratado pelo banco de dados sem necessidade.

A estratégia utilizada consiste em após o Model já estar pronto, o *Controller* é construído a fim de requisitar dados para o Model através de seus métodos, manipulá-los e assim disponibilizá-los carregando a *View* para representação desses dados de modo conveniente para o usuário.

A comunicação entre *Controller* e *View* é feita através de requisições POST, tais requisições são feitas através do cabeçalho HTTP dispensando o uso de parâmetros nas URIs.

O *View* será a interface que fará a disposição dos dados para o usuário. Para esta aplicação foi implementado quatro tipos de Views, contemplando os dois módulos da aplicação, o de visualização dos dados contextuais e o de gerência de recursos de sensoriamento e/ou atuação.

- Visualização, responsável por disponibilizar uma interface para qualquer usuário pode visualizar contextos de interesse e seus respectivos valores coletados de sensores;
- Cadastro, responsável por criar um formulário de cadastro e edição de itens do banco de dados;

[4] https://codeigniter.com/

- Pesquisa, responsável por listar todos os dados de uma determinada tabela;
- Loby, responsável por disponibilizar formas rápidas de acesso às funções da aplicação.

A concepção da interface para aplicação web do Servidor de Contexto vai ao encontro da facilidade da utilização por diferentes tipos de usuários.

Visando a portabilidade e a responsividade da aplicação perante aos mais diversos dispositivos presentes no mercado, suas Views foram desenvolvidas utilizando um *framework* HTML, CSS e Javascript para desenvolvimento responsivo de aplicações WEB chamado *Bootstrap*[5].

Com o uso deste *framework* é possível desenvolver a mesma View da aplicação tanto para dispositivos portáteis quanto para dispositivos de mesa. Esta responsividade é feita através de classes HTML, as quais dizem que aquele elemento possui um determinado comportamento quando acessado por um dispositivo de determinado tamanho de tela.

5.1.5 CORE: Framework para Emulação do Gateway e do Servidor de Borda

Em função do distanciamento social imposto em 2020 pela pandemia de Covid-19, após aguardar alguns meses para a estabilização do quadro da crise sanitária e os decorrentes encaminhamentos das autoridades competentes, foi assumido que os esforços de prototipação a campo não seriam factíveis considerando as indefinições para superação deste momento. Neste sentido, a instituição parceira na pesquisa desenvolvida nesta Dissertação de Mestrado, a Embrapa Clima Temperado, teve suas atividades restritas a procedimentos considerados prioritários, tendo sido suspensas da agenda de trabalho atividades de estudo e pesquisa não enquadradas como essenciais.

Considerando isto, foi buscada uma tecnologia de emulação de sistemas distribuídos que facultasse o aproveitamento da infraestrutura de software do Servidor de Borda e Gateway concebidos para o EXEHDA-FC.

Por sua vez, o Servidor de Contexto em uso no EXEHDA-FC permaneceu em execução em equipamento da Nuvem Computacional da UFPEL, provida pela Pró-Reitoria de Gestão da Informação e Comunicação[6]. Esta capacidade interoperação com o equipamento externo a infraestrutura de emulação, constituiu um aspecto central para escolha da plataforma a ser utilizada.

Neste sentido, o ambiente computacional concebido para avaliação deste Cenário de Uso é baseado no *Common Open Research Emulator* (CORE) (COMMONOPENRESEARCHEMULATOR, CORE). Este framework *open-source* desenvolvido pela *Boeing*

[5] https://getbootstrap.com/
[6] https://wp.ufpel.edu.br/cti/

Research and Development Division (BRT)[7] visa a emulação de ambientes computacionais, permitindo assim testes em ambientes computacionais distribuídos compostos de vários nodos.

A infraestrutura computacional provida pelo CORE é emulada dinamicamente na medida que a execução se desenvolve. Com o CORE os equipamentos virtuais são criados tendo o Linux como sistema operacional de base. O CORE pode ser manipulado tanto via interface gráfica (GUI) como por Python Scripting. Geralmente a GUI é utilizada para especificar graficamente a interconexão dos nós, sendo os módulos em Python são utilizados para configurar e instanciar os procedimentos computacionais nos nodos de processamento, bem como para especificar parâmetros operacionais dos ativos de rede (hubs, switches, roteadores, etc.) empregados.

Com isto, o CORE permite a emulação das diferentes características inerentes a um ambiente computacional real. A Figura 21 exibe uma visão geral do ambiente de emulação provido pelo CORE, onde estão caracterizados nodos computacionais conectados por ativos usualmente empregados em uma infraestrutura de rede.

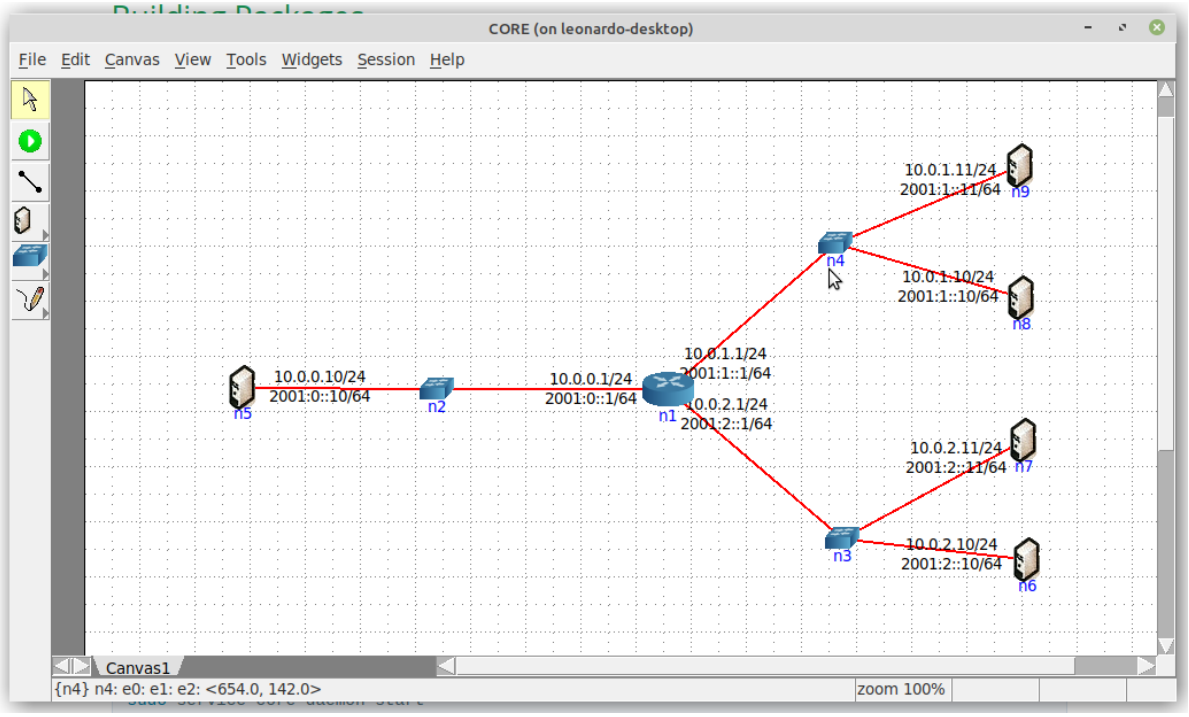

Figura 21 – Ambiente de Emulação do CORE
Fonte: O Autor

O hardware utilizado neste Cenário para instalação do CORE de Uso foi constituído por um processador Pentium G4560 de 3.5 GHz e dois núcleos, 16 GB de memória RAM DDR4 e HD 1TB 5400 RPM, com o Sistema Operacional Linux (Manjaro).

[7] http://www.boeing.com.au/products-services/research-technology.page

5.2 Área do Cenário de Uso: Viticultura de Precisão

O Cenário de Uso explorado para avaliação no EXEHDA-FC foi definido considerando demandas identificadas junto a empresa Empresa Brasileira de Pesquisa Agropecuária (Embrapa), particularmente a sua unidade Clima Temperado. A Embrapa Clima Temperado, vinculada ao Ministério da Agricultura, Pecuária e Abastecimento, está localizada em Pelotas, Rio Grande do Sul, Brasil. Esta unidade da Embrapa é comprometida com a pesquisa eco-regional e desenvolve atividades que buscam viabilizar soluções de desenvolvimento e inovação para a sustentabilidade da agricultura na região de clima temperado, em benefício direto tanto da comunidade produtora rural, como da sociedade em geral.

Mais especificamente, o tema deste Cenário de Uso é direcionado a Viticultura de Precisão, a qual pode ser entendida como a gestão da variabilidade temporal e espacial das áreas de videiras cultivadas. O objetivo central considerado é melhorar o rendimento econômico da atividade agrícola, seja pelo aumento da produtividade ou qualidade do produzido e/ou pela redução dos custos de produção, associado a redução do impacto ambiental.

A variabilidade temporal e espacial diz respeito às características da área agrícola que se referem às alterações dos fatores importantes às culturas, sejam físicos/químicos do solo e/ou climáticos, ao longo da área plantada, ao longo das diferentes etapas de desenvolvimento da planta (BRAGA; PINTO, 2009).

Nesse sentido, uma das principais técnicas utilizadas na Viticultura de Precisão é a tecnologia de taxa variável, a qual consiste em uma detalhada caracterização da variabilidade espacial da cultura e na gestão de fatores que afetam a produção.

Um aspecto que está associado a qualidade do vinho produzido diz respeito ao momento que a irrigação ocorre, onde o uso da água no vinhedo varia significativamente conforme o estágio de desenvolvimento da planta. No início da fase de crescimento, normalmente as plantas necessitam de uma baixa demanda de água devido à menor área foliar, quando o cultivo estiver plenamente desenvolvido este irá requerer uma alta demanda.

Outro aspecto a ser considerado em relação ao gerenciamento de sistemas para irrigação é a capacidade do solo para retenção de água. A fim de não prejudicar o desenvolvimento e a produção de frutos, deve-se evitar que a reserva hídrica do solo esgote. Portanto, as estratégias de gerenciamento da irrigação podem variar espacialmente ao longo da área cultivada (CONCEIÇÃO, 2008).

5.3 Aspectos considerados no Cenário de Uso

Para aquisição das informações contextuais é considerado o uso de Servidores de Borda e Gateways do EXEHDA-FC, os quais irão prover suporte a necessária distri-

buição dos diversos Sensores ao longo das zonas de manejo.

A Figura 22, representa um vinhedo típico da região sul do Brasil com suas respectivas zonas de manejo. A organização dada para este Cenário de Uso contempla a instanciação de sete Células de Execução no EXEHDA-FC.

Para suporte as interoperações neste tipo de ambiente físico, cada Célula de Execução contém uma estação rádio-base empregando o padrão Lora[8]. Os equipamentos que implementam o padrão Lora podem prover comunicação a distâncias elevadas, mesmo operando alimentados por energia solar. Isto tem promovido a adoção da tecnologia Lora em aplicações envolvendo instalações agrícolas.

Figura 22 – Zonas de Manejo em uma Área de Plantação de Videiras
Fonte: O Autor

Para determinar o momento correto de irrigar é necessário avaliar as condições físicas do ambiente. Neste Cenário de Uso, são consideradas as seguintes variáveis contextuais:

- Tensão da água no solo: esta informação é obtida através de sensores do tipo Tensiômetro, com saída de informação por nível de tensão;
- Temperatura e umidade do ar: estas informações são obtidas pelo uso de sensores com interfaceamento por rede 1-Wire.

Por sua vez, os comandos de atuação disparados pelas regras para processamento contextual podem ter as seguintes naturezas:

- Alertas Visuais;
- Alertas Sonoros;

[8] https://lora-alliance.org/

- Envio de mensagens (e-mail);
- Acionamento de transdutores elétricos para acionamento dos sistemas de irrigação.

Por meio deste Estudo de Caso pretende-se avaliar a arquitetura proposta quanto à capacidade de operação em *Fog Computing*, distribuindo o processamento das informações contextuais entre os Servidores de Borda e Contexto.

5.4 Avaliações Realizadas

As avaliações realizadas consideram as características do EXEHDA-FC dos sistemas baseados em *Fog Computing*, dentre estas a capacidade de realizar processamento local dos dados contextuais a nível de borda. Dessa forma, os dados contextuais são processados o mais próximo do local da coleta, minimizando os custos associados às transmissões de longa distância pela Internet até o Servidor de Contexto.

Para realização do estudo de caso foi considerada uma distribuição de Servidores de Borda, Gateways e Sensores considerando 7 regiões da área cultivada, onde cada uma é composta por:

- Zona de Manejo: cada Zona é gerenciada por um Servidor de Borda. Este servidor é responsável pela centralização dos dados contextuais coletados, bem como pela gerência das irrigações a serem realizadas de forma autônoma, interoperando com os outros SBs;
- Gateway: cada Zona de Manejo gerencia 6 Gateways dentro da sua área de cobertura. A comunicação do Gateway acontece de duas formas: (i) com o Servidor de Borda ocorre via Wi-Fi, e (ii) com os sensores por meio de conexão cabeada. O emprego de conexões cabeados com os sensores são determinantes para impor uma distribuição dos Gateways ao longo da Zona de Manejo;
- Sensoriamento: cada Gateway coordena a operação 6 sensores de cada tipo: tensão de solo (tensiômetros), umidade e temperatura.

Com a organização acima tem-se no estudo de caso 756 sensores, 42 Gateways, 7 Servidores de Borda sendo emulados através do CORE. A distribuição dos Gateways e Servidores de Borda ao longo do Vinhedo é feita conforme apresentado na Figura 22.

A avaliação do Cenário de Uso proposto considerou as duas situações a seguir:

- Operação sem Fog: a publicação dos dados contextuais dos sensores é transmitida ao Servidor de Contexto no momento da coleta. O Servidor de Contexto

recebendo os dados sensoriados das zonas de manejo quando um estado que exija intervenção é detectado, realiza o envio de um comando de atuação, de maneira remota, ao Servidor de Borda envolvido;

- Operação com Fog: neste caso é realizada uma operação de filtragem e agregação de dados contextuais, publicando somente a média dos valores coletados considerando um determinado período de tempo. Neste modo de operação os Servidores de Borda precisam interoperar e decidir entre si, a realização ou não das irrigações.

A tensão de água no solo de cada zona de manejo é monitorada e comparada com valor da tensão crítica da cultura que corresponde ao valor de tensão em que se deve proceder a irrigação.

A tensão crítica para a Uva Vinífera está na faixa de 15kPa a 50kPa, o valor mais baixo da tensão crítica deve ser usado em situações de elevada evapotranspiração e o valor mais alto em situação de baixa evapotranspiração. O valor da tensão da água do solo é obtido através do cálculo da média dos valores medidos por todos os sensores da Zona de Manejo.

Os dados utilizados na emulação do Cenário de Uso foram obtidos junto a EMBRAPA Clima Temperado, já tendo sido utilizados em outros trabalhos de estudo e pesquisa do autor, pode ser observada na Figura 23.

O processamento contextual para decidir a realização de irrigação em uma determinada Zona de Manejo tem por base a análise da tensão do solo em relação aos valores de umidade e temperatura ambiente, considerando o estado de Zonas de Manejo vizinhas.

Abaixo o processamento contextual realizado considerando a literatura da área, bem como a orientação técnica da Embrapa.

- **SE** o valor da tensão de água do solo do solo estiver abaixo do valor considerado crítico **ENTÃO** a irrigação é imediatamente acionada e são gerados alertas (luminoso, sonoro, mensagens de e-mail).

- **SE** o valor medido da tensão da água do solo estiver baixo, mas não crítico, **SE** a temperatura ambiente está alta **SE** a umidade relativa do ar estiver baixa **SE** alguma zona de manejo vizinha já tenha identificado valor crítico para tensão de água do solo em um intervalo de tempo especificado pelo usuário **ENTÃO** a irrigação é imediatamente acionada e alertas são gerados (luminoso, sonoro, mensagens de e-mail).

Este perfil operacional para processamento contextual explora as facilidades de interoperação do Servidor de Borda discutidas na Seção 4.3, as quais somente se tor-

Figura 23 – Visão Gráfica Parcial dos Níveis de Tensão de Água do Solo Utilizados
Fonte: O Autor

naram factíveis pelo middleware EXEHDA a partir das contribuições ao Subsistema de Reconhecimento de Contexto e Adaptação feitas por este trabalho.

Neste Cenário de Uso foram realizadas três avaliações: (i) Aquisições Regulares; (ii) Leituras em Intervalos Reduzidos de Tempo, e (iii) Escalabilidade dos Sensores Empregados. Para melhor caracterizar as avaliações realizadas, as mesmas foram dividades em: (a) Operação sem *Fog Computing*; e (b) Operação com *Fog Computing*, as quais estão caracterizadas nas próximas Sessões.

5.4.1 Operação sem *Fog Computing*

Nessa Seção estão caracterizados os testes realizados sem o uso de *Fog Computing*. Dessa forma, os dados contextuais produzidos pelos sensores são enviados imediatamente pelos Servidores de Borda ao Servidor de Contexto, o qual armazena os dados, bem como realiza todo processamento contextual, retornando os respectivos comandos de atuação que disparam as irrigações, quando for o caso.

Na primeira avaliação realizada, denominada de Aquisições Regulares, considerou-se a aquisição de dados dos sensores em intervalos de 5 minutos. Este intervalo foi considerado pela Embrapa como o mínimo razoável de se considerar.

A Tabela 3, apresenta os resultados obtidos considerando o período para totalização do volume de dados a ser transferido para o Servidor de Contexto. Os resultados

obtidos estão agrupados em quatro intervalos de tempo: (i) hora; (ii) dia; (iii) semana e (iv) mês.

Tabela 3 – Volume de Dados Transferido sem *Fog Computing* & Intervalo de Aquisição & Leituras a Cada 5 Minutos

Intervalo de Aquisição	Volume de dados
1 Hora	1,687 MBytes
1 Dia	40,49 MBytes
7 Dias	283,5 MBytes
30 dias	1.215 GBytes

Na segunda avaliação realizada, denominada Leituras em Intervalos Reduzidos de Tempo, explorou a modificação do período de aquisição dos dados sensoriados. Esta avaliação foi realizada, pois em alguns cenários é necessário preservar um controle mais rigoroso sobre o ambiente em monitoramento.

A Tabela 4, apresenta o total de dados transferidos ao longo de um mês com diferentes frequências de aquisição. As frequências de aquisição estipuladas foram de 1, 5 e 10 minutos.

Tabela 4 – Volume Transferido de Dados Sem *Fog Computing* & Diferentes Período de Aquisição dos Dados Sensoriados & Intervalo de um Mês

Período para Aquisição	Volume de dados
1 Minuto	6.074 GBytes
5 Minutos	1.215 GBytes
10 Minutos	607,5 MBytes

A terceira avaliação realizada, denominada Escalabilidade dos Sensores Empregados, foi direcionada a verificar o impacto da variação do número de sensores empregados em cada Zona de Manejo. Para tanto, foi variado o total de sensores ativos em cada Gateway.

A variação no volume de dados transmitidos no período de um mês considerando o número de sensores coletando dados está disponível na Tabela 5.

5.4.2 Operação com *Fog Computing*

Nessa Seção são apresentados os testes realizados considerando o processamento em regime de *Fog Computing*, no qual a decisão de irrigar ou não é realizada pelos Servidores de Borda explorando as funcionalidades de interoperação propostas por este trabalho.

Tabela 5 – Volume de Dados Transferido sem *Fog Computing* & Total de Sensores por Gateway & Intervalo de 1 Mês

Total de Sensores por Gateway	Volume de dados
3 Sensores	173,5 MBytes
6 Sensores	347 MBytes
9 Sensores	520,5 MBytes

Como neste caso, o Servidor de Contexto ficará responsável prioritariamente pelas funções de de armazenamento, oferta de visualização e envio de alertas para o usuário, o repasse das informações contextuais pode ser feita em períodos de tempo mais relaxados.

Buscando priorizar o acompanhamento dos dados coletados pelo usuário, foi definido um intervalo de 10 minutos para envio das informações sensoriadas ao Servidor de Contexto. Um intervalo tempo maior para envio, o qual pode ser configurado a critério do usuário no EXEHDA-FC, implicaria em custos mais reduzidos de banda passante para transferência das informações entre a infraestrutura de borda (*Fog Computing*) para o Servidor de Contexto (*Cloud Computing*).

As avaliações a seguir, traduzem as mesmas realizadas com ou sem o uso de *Fog Computing*, com o intuito de permitir uma comparação com relação o custo de rede imposto para troca de informações.

A Tabela 6, apresenta os resultados para a avaliação denominada de Aquisições Regulares, via de regra bastante utilizada, pois permite que o usuário estabeleça um período de avaliação das informações sensoriadas considerando seu entendimento sobre a perspectiva de evolução das informações contextuais ao longo do tempo.

Tabela 6 – Volume de Dados Transferido com *Fog Computing* & Intervalo de Aquisição & Leituras a Cada 5 Minutos

Intervalo de Aquisição	Volume de dados
1 Hora	0,8437 MBytes
1 Dia	20,24 MBytes
7 Dias	141,7 MBytes
30 Dias	607,5 MBytes

Na segunda avaliação realizada, Leituras em Intervalos Reduzidos, cujos resultados estão apresentados na Tabela 7, traduz uma situação em que se deseja avaliar as condições de contexto em intervalos pequenos, porém não se dispõe de infraestrutura de rede para transferir o volume de dados envolvido via Internet, para uma tomada de decisão remota.

Tabela 7 – Volume Transferido de Dados com *Fog Computing* & Diferentes Período de Aquisição dos Dados Sensoriados & Intervalo de um Mês

Período para Aquisição	Volume de dados
1 Minuto	607,5 MBytes
5 Minutos	607,5 MBytes
10 Minutos	607,5 MBytes

Por fim, terceira avaliação realizada, Escalabilidade dos Sensores Empregados, tem seus resultados apresentados na Tabela 8.

Esta avaliação está associada a uma necessidade de um maior número de informações para a tomada de decisões, o que uma operação em *Fog Computing* se beneficiaria das trocas acontecerem em uma rede local, cuja oferta de banda passante é bem maior.

Tabela 8 – Volume de Dados Transferido com *Fog Computing* & Total de Sensores por Gateway & Intervalo de 1 Mês

Número de sensores	Volume de dados
3 Sensor	86,78 MBytes
6 Sensores	173,56 MBytes
9 Sensores	260,34 MBytes

Observando os resultados das operações explorando ou não *Fog Computing* é evidenciada um potencial redução da banda passante necessária para ciência de contexto.

Este fato se reveste de um elevado significado, pois grande parte das áreas rurais dependem de conexões providas pela rede de celular para conexão à Internet, muitas vezes sujeitas à baixa largura de banda e volume de dados mensais limitados. Esses aspectos apontam para a necessidade de uso controlado do tráfego de dados através da Internet e o uso de estratégias que garantam a operação em momentos em que esse acesso não é possível.

5.5 Considerações Sobre o Capítulo

Neste capítulo é discutido o Cenário de Uso considerado para avaliação das funcionalidades do EXEHDA-FC. Também é caracterizado o esforço de prototipação realizado para os diferentes componentes arquiteturais, bem como a estratégia de emulação que se fez necessária. O cenário explora características típicas da IoT, como por exemplo: processamento contextual distribuído, conexões via Internet,

etc. O EXEHDA-FC mostrou-se capaz de lidar com dados contextuais de maneira distribuída, podendo manter sua operação mesmo com períodos de desconexão com a internet. Estes aspectos apontam para a possibilidade de utilizar o EXEHDA-FC na construção de aplicações distribuídas, considerando o perfil operacional da IoT.

6 CONSIDERAÇÕES FINAIS

Este capítulo é dedicado a apresentar as principais conclusões decorrentes do trabalho de concepção do EXEHDA-FC, apresenta as publicações realizadas e os principais trabalhos futuros identificados.

6.1 Principais Conclusões

A Internet das Coisas, como decorrência de sua escalabilidade crescente, vem ganhando destaque como uma abordagem para promover soluções computacionais distribuídas no mundo real, Neste sentido, inclusive é entendida como a tendência tecnológica atual para materializar as premissas da Computação Ubíqua, principalmente em relação aos aspectos infraestruturais para suporte a interoperação entre dispositivos de diferentes naturezas.

Um aspecto que se destaca para viabilizar o uso da IoT neste cenário de elevada escalabilidade, garantido a premissa da ubiquidade de minimizar o envolvimento do usuário em aspectos de gerência da infraestrutura computacional, é o emprego de estratégias autonômicas para controle computação, explorando para isto ciência de contexto e/ou situação.

Neste sentido, a revisão de literatura feita apontou a Ciência de Contexto enquanto um modelo de computação no qual o sistema computacional é capaz de verificar pelo emprego de regras as características do meio de seu interesse e quando necessário disparar ações sobre o mesmo. Por sua vez, a Ciência de Situação pode ser resumida como a interpretação de um conjunto de elementos contextuais considerando seu comportamento em um intervalo específico de tempo.

Considerando que a área de *Fog Computing* é uma área emergente no cenário internacional, a qual, inclusive, teve um crescimento significativo durante a realização deste trabalho de dissertação, fez-se necessário uma busca criteriosa para localizar trabalhos alinhados com as premissas do middleware EXEHDA de prover uma computação distribuída, multicelular e de perfil autônomo.

Uma das principais conclusões desta etapa de identificação e revisão dos Traba-

lhos Relacionados aponta que uma estratégia de *Fog Computing*, baseada em um tratamento distribuído de eventos, mostra-se promissora para promover a descentralização dos procedimentos de processamento contextual. A perspectiva é expandir, dentre outros aspectos a escalabilidade, das atuais soluções baseadas somente em *Cloud Computing*, as quais concentram o processamento de contexto e situação em servidores centralizados.

Neste sentido, tendo por base a sistematização das diferentes características dos Trabalhos Relacionados identificados, foi realizada a concepção da arquitetura do EXEHDA-FC, bem como definidas as suas funcionalidades.

A arquitetura proposta, permite a exploração das premissas de *Fog Computing* na coleta e processamento de dados contextuais. Sua operação acontece de maneira distribuída, empregando as bordas computacionais para realização de processamento contextual.

Como forma de atender aplicações de diferentes naturezas, cada uma com as suas diferentes demandas, o EXEHDA-FC faculta a instanciação de regras nos equipamento de borda, bem como mantém a solução já prevista para o middleware EXEHDA de processamento contextual centralizado a nível celular. Este processamento centralizado acontece em um Servidor de Contexto localizado em um equipamentos hospedeiro, via de regra, localizado em uma nuvem computacional cujo acesso acontece pela Internet.

O Cenário de Uso trabalhado explorou as funcionalidades de aquisição de dados contextuais e seu processamento, ambas de forma distribuída. Os resultados obtidos se mostraram oportunos para redução do volume de dados transmitidos entre as bordas computacionais (Servidores de Borda) e a infraestrutura de nuvem (Servidor de Contexto), promovendo a ciência de contexto empregando de forma colaborativa as bordas computacionais.

6.2 Publicações Realizadas

Nesta seção são apresentadas as publicações realizadas como primeiro autor ou coautor relacionadas aos tópicos de pesquisa do EXEHDA-FC:

- Applying the Internet of Things in Precision Viticulture: An Approach Exploring the EXEHDA Middleware; Latin American Computing Conference - CLEI; 2018 (Primeiro Autor)

- Uma Contribuição à Ciência de Contexto no Middleware EXEHDA Explorando Fog-Computing; Seminário Integrado de Software e Hardware - SEMISH; 2018 (Primeiro Autor)

- Continuous Monitoring Seed Testing Equipaments Using Internet of Things. Journal of Computers and Eletronics in Agriculture; 2019.

- Exploring Fog-Computing for Context Awareness in EXEHDA Middleware; Simpósio Sul de Microeletrônica - SIM; 2018

6.3 Trabalhos Futuros

A concepção da arquitetura do EXEHDA-FC tem por foco contribuir com o avanço do Subsistema de Reconhecimento de Contexto e Adaptação do Middelware EXEHDA. Considerando que o processamento de informações contextuais é um tópico central de pesquisa no LUPS, contemplado em diferentes trabalhos, foram identificadas algumas oportunidades para desdobramento do trabalho realizado, dentre as quais destacaríamos:

- Expandir as opções para integração do Node-RED com o módulo empregado para geração de regras. A expectativa é poder gerar regras complexas, que permitam uma melhor exploração de aspectos composicionais, facultando uma melhor exploração dos conectivos AND e OR;

- Considerando um cenário de uso intenso de *Fog Computing*, se mostra oportuno consolidar uma estratégia que que faculte a visualização de dados nos Servidores de Borda envolvidos a medida que são aquisitados. Uma abordagem neste sentido permitiria que determinadas classes de aplicações pudessem se beneficiar de um melhor acompanhamento do avanço das informações contextuais. Esta situação ganha significado uma vez que estas informações seriam enviadas de forma consolidadas para o Servidor de Contexto, no qual o EXEHDA já dispõem de diversas ferramentas de visualização textuais e/ou gráficas;

- Enquanto parte do consórcio de pesquisa com a Embrapa Clima Temperado, avaliar outras oportunidades para emprego do EXEHDA-FC, além do Cenário de Uso contemplado nesta Dissertação, que possam se beneficiar do uso combinado de *Fog e Cloud Computing* para o processamento contextual.

Somando-se a estes trabalhos futuros, artigos serão escritos direcionados a revistas e eventos da área, com o intuito de divulgar os resultados atingidos na etapa final desta Dissertação de Mestrado.

REFERÊNCIAS

AL-GARADI, M. A. et al. A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. , [S.l.], p.1–1, 2020.

ALEGRE, U.; AUGUSTO, J. C.; CLARK, T. Engineering context-aware systems and applications: A survey. **The Journal of Systems and Software**, [S.l.], v.117, p.55–83, 2016.

APSCHEDULER. Disponível em https://apscheduler.readthedocs.io/en/stable/. Acesso em março de 2020.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, [S.l.], v.54, n.15, p.2787–2805, 2010.

AYDOS, T. F. **Sistema de Monitoramento da Manufatura Baseado em RFID no Âmbito da Internet of Things**. 2016. Dissertação de Mestrado — Programa de Pós-Graduação em em Engenharia Mecânica - Universidade Federal de Santa.

BANDYOPADHYAY, S.; SENGUPTA, M.; MAITI, S.; DUTTA, S. Role of middleware for internet of things: A study. **International Journal of Computer Science & Engineering Survey (IJCSES)**, USA, v.2, n.3, p.94–105, 2011.

BASSI, A.; HORN, G. Internet of Things in 2020: A Roadmap for the Future. **European Commission: Information Society and Media**, [S.l.], 2008. disponível em: <http://www.smart-systems-integration.org/public/documents/publications/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v3.pdf>.

BIBRI, S. E. Context Modeling, Representation, and Reasoning: An Ontological and Hybrid Approach. In: **The Human Face of Ambient Intelligence. Atlantis Ambient and Pervasive Intelligence**. Paris: Atlantis Press, 2015. p.197–257.

BLACKSTOCK, M.; LEA, R. Toward a distributed data flow platform for the web of things (distributed node-red). In: INTERNATIONAL WORKSHOP ON WEB OF THINGS, 5., 2014. **Proceedings...** [S.l.: s.n.], 2014. p.34–39.

BRAGA, R.; PINTO, P. Alterações climáticas e agricultura. **Inovação e Tecnologia na Formação Agrícola**, [S.l.], v.12, n.2, p.34–56, 2009.

COMMONOPENRESEARCHEMULATOR(CORE). **Common Open Research Emulator (CORE)**. Disponível em: <https://www.nrl.navy.mil/itd/ncs/products/core>. Acesso em março de 2020.

CONCEIÇÃO, M. A. F. C. **A irrigação na produção de uvas para elaboração de vinhos finos**. [S.l.]: Embrapa Uva e Vinho, 2008.

DAVET, P. T. **EXEHDA-IoT**: Uma Contribuição à Arquitetura do Middleware EXEHDA Direcionada à Internet das Coisas. 2016. Dissertação de Mestrado — Programa de Pós-Graduação em Computação - Universidade Federal de Pelotas.

De Donno, M.; Tange, K.; Dragoni, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. **IEEE Access**, [S.l.], v.7, p.150936–150948, 2019.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, [S.l.], v.5, p.4–7, 2001.

DOBSON, G. B.; CARLEY, K. M. A Computational Model of Cyber Situational Awareness. In: INTERNATIONAL CONFERENCE ON SOCIAL COMPUTING, BEHAVIORAL-CULTURAL MODELING AND PREDICTION AND BEHAVIOR REPRESENTATION IN MODELING AND SIMULATION, 2018. **Anais...** [S.l.: s.n.], 2018. p.395–400.

El Kadiri, S. et al. Current trends on ICT technologies for enterprise information systems. **Computers in Industry**, [S.l.], v.79, p.14 – 33, 2016. Special Issue on Future Perspectives On Next Generation Enterprise Information Systems.

FILHO, H. K. **EXEHDA-HS**: Uma Abordagem Baseada em Redes Peer-to-Peer para Descoberta de Recursos na IoT. 2019. Dissertação de Mestrado — Programa de Pós-Graduação em Computação - CDTec - Universidade Federal de Pelotas.

GAZIS, V. et al. Components of Fog Computing in an Industrial Internet of Things Context. In: SENSING, COMMUNICATION, AND NETWORKING-WORKSHOPS, 2015. **Anais...** [S.l.: s.n.], 2015. p.1–6.

Giang, N. K.; Blackstock, M.; Lea, R.; Leung, V. C. M. Developing IoT applications in the Fog: A Distributed Dataflow approach. In: INTERNATIONAL CONFERENCE ON THE INTERNET OF THINGS (IOT), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.155–162.

Giang, N. K.; Blackstock, M.; Lea, R.; Leung, V. C. M. Developing IoT applications in the Fog: A Distributed Dataflow approach. In: INTERNATIONAL CONFERENCE ON THE INTERNET OF THINGS (IOT), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.155–162.

GIANG, N. K.; LEA, R.; BLACKSTOCK, M.; LEUNG, V. C. M. Fog at the Edge: Experiences Building an Edge Computing Platform. **2018 IEEE International Conference on Edge Computing (EDGE)**, [S.l.], p.9–16, 2018.

GROUP, O. M. **Business Process Model and Notation**. Disponível em: <http://www.bpmn.org/>. Acesso em junho de 2018.

GUILLEMIN, P.; FRIESS, P. Internet of things strategic research roadmap. **The Cluster of European Research Projects**, [S.l.], September 2009.

HU, P.; DHELIM, S.; NING, H.; QIU, T. Survey on fog computing: architecture, key technologies, applications and open issues. **Journal of Network and Computer Applications**, [S.l.], v.98, p.27 – 42, 2017.

João, L. et al. Applying the Internet of Things in Precision Viticulture: An Approach Exploring the EXEHDA Middleware. In: XLIV LATIN AMERICAN COMPUTER CONFERENCE (CLEI), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018. p.680–687.

LOPES, J. B. **Uma Arquitetura para Provimento de Ciência de Situação Direcionada às Aplicações Ubíquas na Infraestrutura da Internet das Coisas**. 2016. 123p. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

LOPES, J. et al. A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing. **j-jucs**, [S.l.], v.20, n.9, p.1327–1351, sep 2014.

LOPES, J. L. B. Uma arquitetura para provimento de ciência de situação direcionada às aplicações ubíquas na infraestrutura da internet das coisas. , [S.l.], 2016.

LOPES, J. L. et al. A middleware for context-aware adaptation in Ubicomp. In: BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, 18., 2012, New York, NY, USA. **Proceedings. . .** ACM, 2012. (WebMedia '12).

MACHADO, J.; MORENO, E.; RIBEIRO, A. A Review of Computing Fog and its Research Challenges. **Journal on Advances in Theoretical and Applied Informatics**, [S.l.], v.3, n.2, p.32–39, 2017.

MACHADO, R. et al. A Hybrid Architecture to Enrich Context Awareness through Data Correlation. In: ANNUAL ACM SYMPOSIUM ON APPLIED COMPUTING, 33., 2018, New York, NY, USA. **Proceedings. . .** Association for Computing Machinery, 2018. p.1451–1453. (SAC '18).

MAIER, A.; SHARP, A.; VAGAPOV, Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In: INTERNET

TECHNOLOGIES AND APPLICATIONS (ITA), 2017, 2017. **Anais...** [S.l.: s.n.], 2017. p.143–148.

MARTÍN, C. et al. Appdaptivity: An Internet of Things Device-Decoupled System for Portable Applications in Changing Contexts. In: SENSORS, 2018. **Anais...** [S.l.: s.n.], 2018.

MOTTA, R. C.; de Oliveira, K. M.; TRAVASSOS, G. H. A conceptual perspective on interoperability in context-aware software systems. **Information and Software Technology**, [S.l.], v.114, p.231 – 257, 2019.

NI, J.; ZHANG, K.; LIN, X.; SHEN, X. Securing fog computing for internet of things applications: Challenges and solutions. **IEEE Communications Surveys & Tutorials**, [S.l.], 2017.

NODE-RED. **Node-red**. Disponível em: <https://nodered.org/about>. Acesso em abril de 2018.

Peralta, G. et al. Fog computing based efficient IoT scheme for the Industry 4.0. In: IEEE INTERNATIONAL WORKSHOP OF ELECTRONICS, CONTROL, MEASUREMENT, SIGNALS AND THEIR APPLICATION TO MECHATRONICS (ECMSM), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1–6.

PERERA, C.; ZASLAVSKY, A. B.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context Aware Computing for The Internet of Things: A Survey. **CoRR**, [S.l.], v.abs/1305.0982, 2013.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context aware computing for the internet of things: A survey. **IEEE Communications Surveys and Tutorials**, [S.l.], v.16, n.1, p.414–454, jan 2014.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, Butterworth-Heinemann Newton, MA, USA, v.64, n.Supplement C, p.1 – 18, 2015.

PIRES, P. F. et al. Plataformas para a Internet das Coisas. **Livro Texto de Minicursos - SBRC 2015**, Vitória - ES, 2015.

QIU, T. et al. How can heterogeneous Internet of Things build our future: A survey. **IEEE Communications Surveys & Tutorials**, [S.l.], v.20, n.3, p.2011–2027, 2018.

RAMME, B. J. Controle Automático de Luminosidade de Ambientes de Alarme com Trava Eletrônica, Aplicados a Sistemas Residenciais, Utilizando Rede Zigbee e Arduino. **XLIII Congresso Brasileiro de Educação em Engenharia (COBENGE), Mauá, Brazil**, [S.l.], 2015.

RASPBERRY. **Raspberry Pi**. Disponível em: <http://www.raspberrypi.org>. Acesso em julho de 2019.

RAVINDRA, P. et al. ECHO: An Adaptive Orchestration Platform for Hybrid Dataflows across Cloud and Edge. In: INTERNATIONAL CONFERENCE ON SERVICE-ORIENTED COMPUTING, 2017. **Anais...** [S.l.: s.n.], 2017. p.395–410.

SARKAR, S. Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. **IET Networks**, [S.l.], v.5, p.23–29(6), March 2016.

Sezer, O. B.; Dogdu, E.; Ozbayoglu, A. M. Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey. **IEEE Internet of Things Journal**, [S.l.], v.5, n.1, p.1–27, 2018.

SOLDATOS, J.; SERRANO, M.; HAUSWIRTH, M. Convergence of utility computing with the internet-of-things. In: INNOVATIVE MOBILE AND INTERNET SERVICES IN UBIQUITOUS COMPUTING (IMIS), 2012 SIXTH INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p.874–879.

SOUZA, R. et al. An Architecture for IoT Management Targeted to Context Awareness of Ubiquitous Applications. **Journal of Universal Computer Science**, [S.l.], v.24, n.10, p.1452–1471, 2018.

SQLITE. **SQLite**. Disponível em: <https://www.sqlite.org/about>. Acesso em março de 2020.

TABIM, V. M. **EXEHDA-DT**: Uma Abordagem Dataflow para Ciencia de Contexto na Industrial Internet of Things. 2018. Dissertação de Mestrado — Mestrado em Engenharia Eletrônica e Computação - Universidade Católica de Pelotas.

TAN, L.; WANG, N. Future internet: The internet of things. In: ADVANCED COMPUTER THEORY AND ENGINEERING (ICACTE), 2010 3RD INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. v.5, p.V5–376.

TEMDEE, P.; PRASAD, R. **Context-Aware Communication and Computing**: Applications for Smart Environment. Switzerland: Springer International Publishing, 2018. n.1. (Springer Series in Wireless Technology).

VAQUERO, L. M.; RODERO-MERINO, L. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. **SIGCOMM Comput. Commun. Rev.**, New York, NY, USA, v.44, n.5, p.27–32, Oct. 2014.

VASILEV, A. et al. Mechanism for context-aware substitution of Smart-M3 agents based on dataflow network model. In: INTERNATIONAL CONGRESS ON ULTRA MODERN

TELECOMMUNICATIONS AND CONTROL SYSTEMS AND WORKSHOPS (ICUMT), 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.113–117.

KRUMM, J. (Ed.). **Ubiquitous Computing Fundamentals**. [S.l.]: Chapman & Hall/-CRC, 2010.

WEISER, M. The Computer for the 21st Century. **Scientific American**, [S.l.], v.265, n.3, p.66–75, January 1991.

WEISER, M. Some computer science issues in ubiquitous computing. **Communications of the ACM**, [S.l.], v.36, n.7, p.75–84, 1993.

YASSEIN, M. B.; SHATNAWI, M. Q.; ALJWARNEH, S.; AL-HATMI, R. Internet of Things: Survey and open issues of MQTT protocol. In: INTERNATIONAL CONFERENCE ON ENGINEERING & MIS (ICEMIS), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.1–6.

YI, S.; LI, C.; LI, Q. A survey of fog computing: concepts, applications and issues. In: WORKSHOP ON MOBILE BIG DATA, 2015., 2015. **Proceedings...** [S.l.: s.n.], 2015. p.37–42.

YOUSEFPOUR, A.; ISHIGAKI, G.; JUE, J. P. Fog Computing: Towards Minimizing Delay in the Internet of Things. In: IEEE INTERNATIONAL CONFERENCE ON EDGE COMPUTING (EDGE), 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.17–24.

YU, Y. Embedded Internet of Things Applications of SQLite Based on WinCE Mobile Terminal. , [S.l.], 06 2020.

ZIMMERLE, C.; GAMA, K. A Web-based Approach Using Reactive Programming for Complex Event Processing in Internet of Things Applications. In: ANNUAL ACM SYMPOSIUM ON APPLIED COMPUTING, 33., 2018, New York, NY, USA. **Proceedings...** ACM, 2018. p.2167–2174. (SAC '18).

ČOLAKOVIĆ, A.; HADŽIALIĆ, M. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. **Computer Networks**, [S.l.], v.144, p.17 – 39, 2018.