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

Dissertação

EXEHDA-DEP: Uma Arquitetura para o Processamento Distribuído de Eventos Complexos

Weslen Schiavon de Souza

Pelotas, 2020

Weslen Schiavon de Souza

EXEHDA-DEP: Uma Arquitetura para o Processamento Distribuído de Eventos Complexos

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.

Orientadora: Profª. Drª. Ana Marilza Pernas

Pelotas, 2020

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

S719e Souza, Weslen Schiavon de

EXEHDA-DEP : uma arquitetura para o processamento distribuído de eventos complexos / Weslen Schiavon de Souza ; Ana Marilza Pernas, orientadora. — Pelotas, 2020.

84 f.

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

1. Processamento de eventos complexos. 2. Processamento de eventos distribuído. 3. Internet das coisas. I. Pernas, Ana Marilza, orient. II. Título.

CDD : 005

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

Weslen Schiavon de Souza

EXEHDA-DEP: Uma Arquitetura para o Processamento Distribuído de Eventos Complexos

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: 25 de Março de 2020

Banca Examinadora:

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

Prof. Dr. Adenauer Correa Yamim
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Andre Rauber Du Bois
Doutor em Ciência da Computação pela Universidade de Heriot-Watt.

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

Dedico aos meus pais, os quais proporcionaram que tudo fosse possível.. . .

AGRADECIMENTOS

Agradeço primeiramente aqueles sem os quais eu jamais estaria presente, meus pais, Rosângela e Valdirnei, eles que sempre se esforçaram para me proporcionar as melhores oportunidades de ensino e sempre me incentivaram a seguir em frente.

A minha namorada Diulie pelas muitas revisões e correções de ortografia.

Aos meus amigos e professores que de alguma forma contribuíram para que este trabalho fosse possível, agradeço a todos.

O homem erudito e um descobridor de fatos que já existem, mas o homem sábio é um criador de valores que não existem e que ele faz existir.

— ALBERT EINSTEIN

RESUMO

SOUZA, Weslen Schiavon de. **EXEHDA-DEP: Uma Arquitetura para o Processamento Distribuído de Eventos Complexos**. Orientadora: Ana Marilza Pernas. 2020. 84 f. Dissertação (Mestrado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2020.

Com o advento de novos avanços tecnológicos, a Internet das Coisas (termo traduzido do inglês *Internet of Things* - IoT) tem se mostrado cada vez mais presente, onde a introdução da computação aos mais variados dispositivos, como geladeiras e cafeteiras, visa oferecer aos seus usuários melhorias e funcionalidades adicionais as já naturalmente esperadas por estes equipamentos. Porém, existem ainda alguns desafios inerentes à IoT, os quais precisam ser transpostos para que esta possa tornar-se uma realidade onipresente a todos os tipos de usuários, sejam estes especialistas da área ou não.

Alguns paradigmas e técnicas computacionais então sendo introduzidos na IoT visando conceber soluções para seus desafios. Dentre estas abordagens destacam-se as aplicadas ao processamento de eventos, que objetivam auxiliar a análise e a extração de informações de alto nível do considerável volume de dados gerado nestas redes de dispositivos, e ao desenvolvimento de *middlewares*, aplicados com a finalidade de abstrair a heterogeneidade inerente aos ambientes de IoT. Foram identificados trabalhos que abordam estratégias visando fornecer arquiteturas de processamento de eventos para a IoT, porém estes não levam em consideração a execução de suas arquiteturas em meios com largura de banda reduzida ou que possuam *links* de comunicação saturados. Ainda, por vezes apresentam pouco ou nenhum teste de validação da capacidade de escalabilidade e distribuição da arquitetura.

Este trabalho tem como objetivo apresentar uma solução, materializada em uma arquitetura, para o processamento distribuído de eventos na IoT, que seja escalável e apta a lidar com a heterogeneidade destes ambientes, aplicando ainda estratégias que visem favorecer a sua execução em meios com largura de banda reduzida ou que possuam *links* de comunicação saturados.

Os objetivos puderam ser atingidos por meio da concepção da EXEHDA-DEP, uma arquitetura para o processamento distribuído de eventos complexos, na qual os resultados demonstraram uma considerável capacidade de processamento de eventos auxiliada pela introdução de algoritmos de compactação e uma estratégia de compartilhamento de trabalho concorrente entre os nodos de processamento ao modelo de um produtor-consumidor.

Palavras-chave: Processamento de Eventos Complexos. Processamento de Eventos Distribuído. Internet das Coisas.

ABSTRACT

SOUZA, Weslen Schiavon de. **EXEHDA-DEP: An Architecture for Distributed Processing of Complex Events**. Advisor: Ana Marilza Pernas. 2020. 84 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2020.

With the advent of new technological advances, the Internet of Things (IoT) has shown itself to be increasingly present, where the introduction of computing to the most varied devices, such as refrigerators and coffee makers, aims to offer its users improvements and additional features to those already naturally expected by this equipment. However, there are still some challenges inherent to IoT, which need to be overcome so that it can become a ubiquitous reality for all types of users, whether these experts in the field or not.

Some paradigms and computational techniques then being introduced in the IoT in order to devise solutions for your challenges. Among these approaches, those applied to event processing stand out, which aim to assist the analysis and extraction of high-level information from the considerable volume of data generated in these device networks, and to the development of middleware, applied with the purpose of abstracting the inherent heterogeneity in IoT environments. Studies were identified that address strategies aimed at providing event processing architectures for the IoT, but these do not take into account the execution of their architectures in mediums with reduced bandwidth or that have saturated communication links. Yet, sometimes they present little or no validation test of the architecture's scalability and distribution capacity.

This work aims to present a solution, materialized in an architecture, for the distributed processing of events in the IoT, which is scalable and able to deal with the heterogeneity of these environments, also applying strategies that aim to favor its execution in media with width of reduced bandwidth or that have saturated communication links.

The objectives could be achieved through the design of EXEHDA-DEP, an architecture for the distributed processing of complex events, in which the results demonstrated a considerable event processing capacity aided by the introduction of compression algorithms and a work-sharing strategy competitor between the processing nodes to the model of a producer-consumer.

Keywords: Complex Event Processing. Distributed Event Processing. Internet of Things.

LISTA DE FIGURAS

1 Relação associativa entre os conceitos. 22

2 *Strings* de buscas usadas. 26
3 Percentual de publicações encontradas por base. 27
4 Número de publicações encontradas por base. 27
5 Quantidade de publicações de interesse por ano. 28
6 Fluxo de triagem dos artigos. 31
7 Comparativo entre os artigos selecionados. 39

8 Gerenciamento de um Ambiente Ubíquo pelo EXEHDA. 41
9 Arquitetura EXEHDA. 41
10 Arquitetura EXEHDA-SA. 43

11 Fluxo de comunicação padrão. 46
12 Comparativo de consumo de rede. 47
13 Fluxo de comunicação proposto. 48
14 Arquitetura simplificada Produtor Consumidor. 50
15 Cabeçalho do protocolo MQTT. 50
16 Fluxo de comunicação Apache Kafka. 52
17 Módulos a serem modificados no EXEHDA-SA. 55
18 Nodo de Pré-processamento. 56
19 Nodo Broker. 58
20 Nodo de processamento. 59
21 Visão geral da arquitetura. 60

22 Fluxo de execução do EXEHDA-DEP. 65
23 Escalabilidade Vertical nodo de pré-processamento. 66
24 Escalabilidade Vertical nodo de processamento. 67
25 Escalabilidade horizontal nodo de pré-processamento. 68
26 Escalabilidade horizontal nodo de processamento. 69
27 Estabilidade nodo pré-processamento. 70
28 Estabilidade nodo processamento. 71
29 Consumo de recursos pré-processamento. 72
30 Consumo de recursos processamento. 72
31 Consumo de rede com a distribuição. 74
32 Consumo de rede por algoritmos de compactação. 75
33 Fluxo de processamento com compactação. 76

LISTA DE ABREVIATURAS E SIGLAS

CEML	Complex Event Machine Learning
CEP	Complex Event Processing
CPU	Central Processing Unit
CSV	Comma-Separated Values
DAG	Directed Acyclic Graph
DUP	Duplicate Delivery
EPL	Event Processing Language
ESP	Event Stream Processing
EXEHDA	Execution Environment for Highly Distributed Applications
EXEHDA-HM	Execution Environment for Highly Distributed Applications - Hybrid Modeling
EXEHDA-SA	Execution Environment for Highly Distributed Applications - Situational Awareness
HD	Hard Disk
HDFS	Hadoop Distributed File System
IP	Internet Protocol
IoT	Internet of Things
JSON	Javascript Object Notation
LAs	Learning Agents
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
RAM	Random Access Memory
SQL	Structured Query Language
UbiComp	Ubiquitous Computing
UFPel	Universidade Federal de Pelotas

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Motivações 15
1.2 Objetivos e Contribuições 16
1.3 Organização do Trabalho 17

2 EMBASAMENTO TEÓRICO 18
2.1 Internet das Coisas 18
2.2 Processamento de Eventos 20
2.2.1 Processamento de Fluxo de Eventos 22
2.2.2 Processamento de Eventos Complexos 23

3 ESTADO DA ARTE 25
3.1 Mapeamento Sistemático da Literatura 25
3.1.1 Critérios de Inclusão e Exclusão 28
3.2 Trabalhos Relacionados 32
3.2.1 SAMURAI: A batch and streaming context architecture for large-scale intelligent applications and environments 32
3.2.2 A Distributed Complex Event Processing System Based on Publish/-Subscribe 33
3.2.3 CEML: Mixing and moving complex event processing and machine learning to the edge of the network for IoT applications 34
3.2.4 Semantic IoT Middleware-enabled Mobile Complex Event Processing for Integrated Pest Management 35
3.2.5 Parallel big data processing system for security monitoring in Internet of Things networks 36
3.3 Discussão dos Trabalhos Relacionados 37

4 ESCOPO DE DESENVOLVIMENTO 40
4.1 EXEHDA 40
4.2 EXEHDA-SA 42

5 EXEHDA-DEP: CONCEPÇÃO E TECNOLOGIAS 45
5.1 Modelo de Comunicação 45
5.2 Modelo de Processamento 47
5.3 Tecnologias Associadas 49
5.3.1 Protocolo MQTT 49
5.3.2 Apache Kafka 51
5.3.3 Apache Spark 52

5.3.4 Esper 53
5.4 Concepção da Arquitetura 53
5.4.1 Nodo de Pré-processamento 54
5.4.2 Nodo Broker 56
5.4.3 Nodo de Processamento 57
5.4.4 Visão geral da arquitetura 59

6 EXEHDA-DEP: AVALIAÇÕES E RESULTADOS 61
6.1 Cenário de Aplicação 61
6.1.1 Ambiente de Teste 62
6.2 Escalabilidade 65
6.2.1 Escalabilidade Vertical 65
6.2.2 Escalabilidade Horizontal 67
6.3 Estabilidade e Consumo de Recursos 69
6.4 Consumo de Rede 73

7 CONSIDERAÇÕES FINAIS 78
7.1 Contribuições 80
7.2 Trabalhos Futuros 80

REFERÊNCIAS 81

1 INTRODUÇÃO

A Computação Ubíqua (*Ubiquitous Computing* - UbiComp) se caracteriza pela integração da computação ao ambiente de forma onipresente e imperceptível aos seus usuários, de modo que estes possam interagir com a tecnologia de maneira tão elementar e transparente quanto possível, visando assim favorecer a interação simplificada e ocultando perante aos usuários, toda e qualquer complexidade do uso da tecnologia (KRUMM, 2016).

Um paradigma emergente que tem se mostrado uma materialização da UbiComp é a Internet das Coisas (*Internet of Things* - IoT). A IoT consiste da integração de dispositivos computacionais móveis e com conectividade, a objetos físicos comuns, como lâmpadas e cafeteiras, proporcionando a inclusão destes dispositivos a redes sem fio, onde os dados gerados durante o funcionamento destes equipamentos podem ser coletados e armazenados em nuvem, permitindo o uso de ferramentas para extrair dados semânticos destas informações e assim fornecer algum novo serviço ao usuário deste equipamento (HANES et al., 2017).

A IoT tem se popularizado e beneficiado principalmente pelo avanço de diversas áreas da tecnologia, como a dos sistemas embarcados, a microeletrônica, a comunicação e o sensoriamento. Tais avanços tecnológicos têm favorecido o barateamento e a elaboração de microcontroladores menores e com maior poder computacional, o que propiciou a concepção de bibliotecas dedicadas e semanticamente de mais alto nível aos mesmos, contribuindo para o desenvolvimento e a portabilidade de softwares que necessitem maior poder computacional (RUIZ-RUBE et al., 2019).

Tais avanços tecnológicos permitiram a integração destes dispositivos aos mais variados objetos, adicionando “inteligência” a estes, permitindo assim oferecer outros serviços aos seus usuários, facilitando seu uso, como por exemplo, uma cafeteira que se liga automaticamente minutos antes de seu usuário acordar, evitando que o mesmo tenha de ligar a cafeteira e esperar que seu café fique quente.

Algumas previsões mostram que há um crescimento constante no número de dispositivos conectados, estimando para 2020 mais de 50 bilhões de equipamentos ligados à internet. Tais perspectivas demostram que a IoT não é um futuro longínquo,

destacando a relevância dos estudos sobre este paradigma computacional (XAVIER, 2016).

Porém, existem ainda alguns desafios inerentes à IoT que vem dificultando seu avanço, dentre estes, pode-se citar: o tratamento da heterogeneidade das informações geradas pelos dispositivos, já que diferentes modelos destes, potencialmente dispondo de *Hardware* e recursos distintos, podem estar comunicando-se nestes ambientes, sem qualquer tipo de padronização (PARK; ABUZAINAB; SAAD, 2016); o processamento do considerável volume de eventos produzido por estas redes de dispositivos, onde devido a natureza distribuída destes ambientes, torna-se indispensável a análise dessas informações de forma descentralizada e escalável (KOTENKO; SAENKO; KUSHNEREVICH, 2017); a largura de banda necessária para trafegar o considerável volume de dados geradas nestes ambientes (CHEN; KUNZ, 2016), onde muitas estratégias de processamento de eventos abordam técnicas negligentes quanto ao consumo de rede.

Alguns paradigmas e técnicas computacionais vem sendo considerados na IoT visando conceber soluções para estes desafios. Dentre as abordagens comumente aplicadas a este meio, pode-se citar:

- **Processamento de Eventos** - paradigma computacional empregado com o objetivo de auxiliar na análise e extração de informações de alto nível do considerável volume de dados gerado nestas redes de dispositivos (KAMIENSKI et al., 2019). O conceito de evento adotado neste paradigma normalmente é caracterizado por uma tentativa de alteração de estado do sistema, a qual comumente inclui, a noção de tempo, localidade e detalhes pertinentes a ação que originou esta determinada ocorrência, sendo estas informações fundamentais no auxílio da compreensão das causas ou efeitos desencadeadores (HEINZ et al., 2019). Dentro deste paradigma surge ainda outros dois conceitos, o processamento de fluxo de eventos (*event stream processing* - ESP) e o processamento de eventos complexos (*complex event processing* - CEP), onde estes tem adquirido amplo destaque no desenvolvimento de soluções voltadas para a IoT (YANG, 2017).

- ***Middlewares* da IoT** - com a finalidade de abstrair a heterogeneidade destes ambientes, algumas soluções tem aplicado *middlewares* para atingir este determinado fim, facilitando por meio destes, por exemplo, o uso dos dados gerados pelos dispositivos destas redes (RAZZAQUE et al., 2015).

1.1 Motivações

Dentro das motivações tomadas para o desenvolvimento deste trabalho, pode-se destacar:

- A carência de soluções de processamento de eventos para IoT que possuam suporte a execução em ambientes com largura de banda reduzida ou que possuam *links* de comunicação saturados;
- A necessidade de arquiteturas de processamento de eventos aptas a executarem em meios altamente distribuídos;
- O desprovimento de arquiteturas de processamento de eventos com a capacidade de lidar com a heterogeneidade dos ambientes da IoT.

Estas motivações citadas por este trabalho foram identificadas com o auxilio da execução de um mapeamento sistemático, o qual é apresentado no capitulo 3, onde foi possível perceber por meio deste que alguns trabalhos como (SOTO et al., 2016) e (NOCERA et al., 2017) abordam estratégias que visam fornecer arquiteturas de processamento de eventos aptas a executarem nos ambientes heterogêneos da IoT, porém estes não levam em consideração a execução de suas arquiteturas em meios com largura de banda reduzida ou que possuam *links* de comunicação saturados. Ainda, grande parte dos demais estudos identificados pelo mapeamento sistemático apresentam pouco ou nenhum teste de validação da capacidade da arquitetura escalar e distribuir, ou ainda, soluções que possuam a capacidade de executar em um ambiente heterogêneos da IoT.

1.2 Objetivos e Contribuições

No âmbito das motivações citadas, o presente trabalho possui como objetivos principais:

- O desenvolvimento de uma arquitetura de processamento de eventos distribuída com escalabilidade voltada à IoT.
- Uma arquitetura apta a lidar com a heterogeneidade dos dados na IoT.
- A concepção de uma arquitetura de processamento distribuída, capacitada para executar em ambientes com largura de banda reduzida ou que possuam *links* de comunicação saturados pelo considerável volume de dados trafegados.

Já como objetivos específicos, este estudo visa aplicar estratégias que aprimorem a eficiência do consumo de largura de banda, com o objetivo de proporcionar a aplicabilidade desta arquitetura em meios que possuam alguma limitação de rede, seja esta uma restrição por *links* de comunicação saturados, onde perdas constantes de pacotes são recorrentes (FALL; STEVENS, 2011), ou em uma rede que possua limitações quanto a largura de banda reduzida.

Os objetivos apresentados foram atingidos com a concepção do EXEHDA-DEP(*(Execution Environment for Highly Distributed Applications - Distributed Event Processing)*) uma arquitetura de processamento de eventos complexos distribuída para a IoT, sendo esta apta a lidar com a heterogeneidade proveniente deste meios bem como de executar em ambientes altamente distribuídos e que possam conter alguma restrição de conexão, seja esta limitação gerada pela largura de banda reduzida ou meios que possuam *links* de comunicação saturados.

1.3 Organização do Trabalho

No capítulo 2 é apresentado o embasamento teórico necessário para uma melhor compreensão da presente dissertação. Já no capítulo 3 é apresentado ao leitor os trabalhos relacionados com esta dissertação, identificados por meio da execução de um mapeamento sistemático. O capítulo 4 contém o escopo de desenvolvimento da proposta concebida por este trabalho. No capítulo 5 é apresentado o EXEHDA-DEP uma arquitetura destinada ao processamento distribuído de eventos complexos. No capítulo 6 é demonstrado os resultados dos testes de avaliação executados no EXEHDA-DEP. Por fim, no capítulo 7 é abordada uma discussão sobre os resultados obtidos com a concepção do EXEHDA-DEP.

2 EMBASAMENTO TEÓRICO

Este capítulo tem como objetivo apresentar o embasamento teórico necessário a compreensão dos trabalhos apresentados no capítulo 3. Na seção 2.1 deste capítulo será introduzido o embasamento teórico sobre a Internet das coisas, destacando seus principais objetivos e desafios a serem superados. Por fim, Na seção 2.2 é abordado o paradigma computacional de processamento de eventos, onde será descrito outros dois sub-conceitos do mesmo: processamento de fluxo de eventos e o processamento de eventos complexos.

2.1 Internet das Coisas

A tecnologia computacional tem avançado consideravelmente nos últimos anos, onde dispositivos móveis de amplo poder computacional com capacidade de se comunicar em rede, estão se tornando cada vez mais comuns. A popularização do uso destes dispositivos "inteligentes" tem sido referenciada como Internet das Coisas, onde este novo paradigma computacional vem mudando a forma como as pessoas interagem com os objetos de seu cotidiano (XAVIER, 2016).

Deste modo, a IoT pode ser vista como a materialização da Computação Ubíqua, a qual tem por finalidade fornecer computação com conectividade constante e mobilidade, de forma transparente e integrada ao ambiente, visando assim anexar-se ao mundo físico, em um esforço para torná-la imperceptível aos usuários, de modo que estes não percebam que estão dando comandos a uma máquina, removendo do usuário toda e qualquer responsabilidade perante a complexidade de uso da tecnologia, necessitando apenas que estes façam uso da mesma (KRUMM, 2016).

A Internet das Coisas tem o potencial de transformar o modo como as pessoas interagem com o mundo ao seu redor, adicionando "inteligência" aos mais variados itens do dia a dia, visando assim adicionar a capacidade de fornecer novas funcionalidades a estes dispositivos ou mesmo aprimorar as já existentes (HANES et al., 2017). As áreas de aplicação para a IoT são das mais diversas possíveis: como o uso em grandes cidades, onde a implementação de dispositivos inteligentes é aderida com o

intuito de fornecer diferentes tipos de serviços a sua população, entre estes pode-se citar o fornecimento de informações sobre o tráfego e de eventos públicos; ainda, a introdução de dispositivos inteligentes na agricultura, onde comumente estes equipamentos executam um monitoramento preciso sobre as plantas e solo (GONÇALVES, 2017).

Porém, para que a IoT se torne efetivamente uma realidade presente aos mais diversos ambientes, servindo a todos os tipos de usuários e não apenas a especialistas da área de tecnologia, existem ainda alguns desafios a serem transpassados, dos quais pode-se citar:

- O tratamento da heterogeneidade das informações geradas pelos dispositivos, já que estes equipamentos podem apresentar hardwares e recursos totalmente distintos, o que tende a tornar as informações concebidas muito discrepantes entre si, tendo em vista que normalmente não há uma padronização no formato dos dados gerados (AGRAWAL; VIEIRA, 2013);

- O processamento do grande volume de eventos gerados por estas redes de dispositivos, já que estes ambientes poderão conter dezenas de milhares de dispositivos interconectados se comunicando constantemente e gerando dados de forma ininterrupta, onde estas informações comumente demandam ser processados e analisados (HANES et al., 2017);

- A largura de banda necessária para trafegar o grande volume de informações geradas neste ambiente, levando em consideração que estes dispositivos “inteligentes” podem estar localizados em áreas remotas, conectados em redes de baixa largura de banda, como por exemplo em zonas rurais (CHEN; KUNZ, 2016);

- A configuração desses dispositivos, a qual deve ser feita de forma simplificada, já que usuários finais sem conhecimentos técnicos precisam ser capazes de adicionar e remover dispositivos e recursos de uma rede sempre que estes desejarem (HANES et al., 2017);

- A segurança aplicada sobre estas redes de dispositivos, tendo em vista que soluções comuns de segurança normalmente não podem ser aplicadas aos dispositivos destes ambientes, devido à muitos destes equipamentos apresentarem hardwares simples e de baixo poder computacional, os quais frequentemente não tem a capacidade para executar técnicas de criptografias modernas (AGRAWAL; VIEIRA, 2013).

Com o intuído de se desenvolver técnicas que visem solucionar tais desafios, alguns estudos tem aplicado estratégias no auxilio do desenvolvimento destas soluções, dentre as quais comumente abordadas por estes trabalhos, pode-se destacar:

- **Processamento de Eventos** - este paradigma computacional vem sendo aplicado na IoT com o objetivo de favorecer a análise e extração de informações de alto nível do considerável volume de dados gerados nestes ambientes, os quais são vistos e modelados neste paradigma como eventos (KAMIENSKI et al., 2019), sendo este conceito de evento, caracterizado por uma ação de alteração de estado do sistema, a qual normalmente inclui: noção de tempo, localidade e detalhes pertinentes a esta ação que deu origem ao evento de interesse (HEINZ et al., 2019). Estas informações agregada pelos eventos auxiliam na análise dos dados, sendo assim oportuno sua aplicação em soluções voltadas para a IoT (HANES et al., 2017).

- ***Middlewares* da IoT** - os *middlewares* são aplicações que atuam como uma camada mediadora entre um meio e os aplicativos nele executados. Esta ferramente normalmente opera traduzindo as informações deste meio para as demais aplicações, favorecendo o uso simplificado dos serviços e recursos fornecidos por este (RAZZAQUE et al., 2015). Estas características tornam oportuno a agregação de *middlewares* na IoT, os quais podem facilitar o uso dos dados gerados nestes ambientes, por aplicações de terceiros, auxiliando assim na abstração da heterogeneidade destes ambientes (HANES et al., 2017).

Estes desafios citados se tornam ainda mais relevantes quando analisado em conjunto com dados de previsões, as quais demostram que há um crescimento constante no número de dispositivos conectados, projetando para 2020 mais de 50 bilhões de equipamentos ligados à internet. Tais dados enfatizam que a IoT não e mais o futuro, mas sim o presente, tomando de suma importância o desenvolvimento de soluções que estejam aptas a lidar de forma eficiente com tais desafios citados (XAVIER, 2016).

2.2 Processamento de Eventos

Para que se possa elucidar sobre o paradigma computacional de Processamento de Eventos há necessidade de se abordar primeiramente o conceito de evento, já que o mesmo está contido pelo processamento de eventos. Assim, a definição de evento adotada por este documento consiste na determinação de como a ocorrência de uma ação específica dentro de um ambiente ocorre, onde esta determinada ação geralmente envolve uma tentativa de alteração de estado do sistema. Esta mudança normalmente inclui, a noção de tempo, localidade e detalhes pertinentes ao evento que desencadeou esta determinada ação, sendo estas informações fundamentais no auxílio da compreensão das causas ou efeitos desencadeadores (FITZGERALD et al., 2010).

A um evento pode-se atribuir campos adicionais, que auxiliem na descrição de

suas propriedades. Um evento na IoT pode incluir quatro atributos: eventID, eventName, eventTime e recordTime. EventID e eventName são normalmente definidos como registros básicos representando o código identificador do evento e seu nome de representação. EventTime e recordTime expressam o conceito de tempo no evento, descrevendo sua hora de ocorrência e de captura respectivamente (MINBO; ZHU; GUANGYU, 2013).

Tais eventos podem ser empregados em diversos sistemas para se atingir determinados fins, como por exemplo, o uso em ferramentas de monitoramento onde estes eventos são utilizados para representar mudanças de situações (CRUZ et al., 2016). Estes sistemas monitorados podem ser representados por conjuntos de sensores, onde por exemplo, em aplicações na agricultura de precisão são usados para o monitoramento da umidade e acidez do solo, de forma que tais valores emitidos por estes sensores podem ser vistos como mudanças de estado do ambiente, como uma mudança brusca na acidez do solo ou em sua umidade, as quais podem ser representadas como uma mudança de situação de interesse (KAMIENSKI et al., 2019).

Assim com o conceito de evento bem definido e exemplificado, pode-se então citar que o processamento de eventos é um paradigma computacional onde eventos são analisados, com o objetivo de extrair informações relevantes e de alto nível destes dados. Existem diversas áreas com aplicabilidade a serem exploradas pelo processamento de eventos, dentre estas pode-se citar os setores da saúde com o monitoramento das condições de saúde dos pacientes onde os diversos eventos precisam ser processados e analisados (PÉREZ-VEREDA et al., 2018); o setor da agricultura de precisão, o qual emprega diversos sensores para o monitoramento de plantas, gerando grandes fluxos de eventos que necessitam ser processados (KAMIENSKI et al., 2019); o setor de energia elétrica, o qual fazendo uso de eventos para o monitoramento do consumo excessivo desta, visando atingir uma melhor eficiência energética (HERDRICH et al., 2018).

O processamento de eventos tem sido empregado em diversos ambientes, com o objetivo de auxiliar na resolução de problemas distintos presentes nos mesmos. Normalmente existem algumas similaridades de requisitos exigidos nestes ambientes, as quais tornam o uso deste paradigma computacional, uma solução em potencial para os desafios presentes neste ambiente. Dentre as estas similaridades comumente compartilhadas entre estes ambientes distintos, pode-se citar a necessidade de processar em tempo de execução um volume expressivo de dados (DAYARATHNA; PERERA, 2018).

A IoT tem se destacado por possuir uma alta aplicabilidade ao processamento de eventos, onde este é aplicado no âmbito de solucionar problemas de tomadas de decisão a partir da análise de grandes volumes de eventos gerados nestes ambientes. Diversas ferramentas foram desenvolvidas para o processamento de eventos, visando

auxiliar na análise do extenso volume de dados, dentre estas pode-se citar o Apache Storm[1], Apache Spark[2] e Apache Flink[3].

O processamento de eventos pode ser subdividido em dois principais conceitos: o processamento de fluxo de eventos, que se caracteriza por ter a capacidade de executar operações contínuas como filtros, agregações, classificações e junções, sob fluxos contínuos de dados; e o processamento de eventos complexos o qual faz uso de padrões pré definidos, aplicando-os sobre dois ou mais eventos simples, visando realizar assim a detecção de eventos compostos por meio da informação adicional de alto nível obtida da associação destes eventos simples (DAYARATHNA; PERERA, 2018).

Na Figura 1 é ilustrada a relação associativa entre estes três paradigmas, onde o processamento de eventos pode ser visto como um conceito mais genérico e abrangente o qual engloba o ESP que por sua vez engloba o CEP.

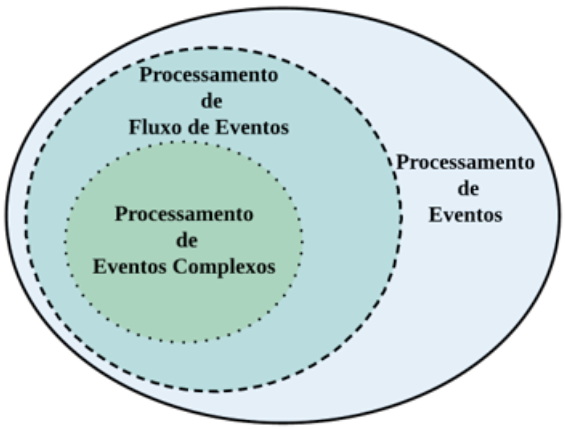

Figura 1 – Relação associativa entre os conceitos.

2.2.1 Processamento de Fluxo de Eventos

Para uma melhor e simples compreensão do significado de processamento de fluxo de eventos, pode-se separar este conceito em outros três sub-conceitos:

1. **Evento -** pode-se definir evento, neste contexto, como qualquer ação que aconteça com um tempo claramente definido, onde o mesmo pode ser mensurado.

2. **Fluxo -** neste contexto é definido como uma sequência contínua de eventos, sendo esta corrente de eventos claramente ordenada no tempo.

[1] https://storm.apache.org/
[2] https://spark.apache.org/
[3] https://flink.apache.org/

3. **Processamento** - é a ação final de executar a análise sobre o conjunto de informações capturadas.

Desta forma, com a combinação destes três sub-conceitos pode-se dizer que o processamento de fluxo de eventos nada mais é que o processo de analisar continuamente uma sequência constante de dados ordenados pelo tempo (DAYARATHNA; PERERA, 2018), o qual trata da identificação de padrões ou de relacionamentos significativos entre os fluxos de dados analisados, a fim de detectar determinados padrões como a correlação de eventos, causalidade ou tempo.

Características normalmente presentes em sistemas que possuem aplicabilidade deste paradigma computacional incluem a necessidade de analisar grandes fluxos de dados, correlacionando estas informações, aplicando filtros em tempo de execução e dando uma resposta de forma imediata (APPEL et al., 2013).

2.2.2 Processamento de Eventos Complexos

O processamento de eventos complexos é um paradigma computacional, o qual normalmente é aplicado para executar o processamento e a análise de conjuntos de informações em sistemas baseados em eventos, visando assim analisar a interação destes eventos entre si. Sistemas que empregam este paradigma normalmente apresentam as seguintes características: a necessidade de se verificar e informar a ocorrência de um evento composto, isto é, a necessidade de identificar a ocorrência de duas ações especificas A e B sob determinadas condições e/ou em um intervalo especifico de tempo. Assim, o sistema deverá ser capaz de notificar a ocorrência destes eventos sob tais condições, de modo que esta associação de dois ou mais eventos sob condições especificas pode ser visto como um novo evento C, o qual é semanticamente distinto se comparado com os eventos A e B que o originaram.

Assim, pode-se definir mais especificamente o processamento de eventos complexos como o ato de analisar dois ou mais eventos simples, visando assim inferir a partir da associação destes, um novo evento semanticamente distinto dos anteriores. Deste modo, pode-se dizer que a partir de uma análise combinatória de eventos simples, o CEP é capaz de gerar um novo conjunto de informações, semanticamente de mais alto nível que as informações analisadas (DAYARATHNA; PERERA, 2018).

Se pode exemplificar um cenário de atuação do processamento de eventos complexos como o *data center* de uma empresa qualquer, onde sensores monitoram o uso do disco rígido do sistema e o uso de rede, quando repentinamente o consumo de disco e de rede aumentam consideravelmente, sendo estes definidos pelo administrador como eventos de interesse, após a análise associativa destes dados de rede e disco, o sistema pode "decidir" disparar um novo evento: "possível ataque hacker" onde a partir do mesmo, os administradores podem tomar alguma decisão com o intuito de mitigar este problema , como por exemplo, desconectar da rede o *data center*

(CRUZ et al., 2016).

Esta capacidade do CEP de extrair informações de alto nível por meio da análise de um conjunto de eventos, torna favorável a sua implementação em meios que necessitem processar e analisar grandes volumes de dados onde a partir da identificação de eventos específicos, determinadas ações devem ser tomadas. Características estas que o tornam oportuno de ser agregado a soluções voltadas para a IoT (HANES et al., 2017).

3 ESTADO DA ARTE

Neste capítulo apresenta-se o estado da arte das pesquisas que tem como o enfoque principal o Processamento de Eventos e a Internet das Coisas. Na seção 3.1 é apresentada a estratégia seguida para a execução do mapeamento sistemático, bem como todos os passos executados, os quais levaram a escolha dos trabalhos de interesse definidos como base para a elaboração deste texto. A seção 3.2 disserta sobre os estudos selecionados como mais relevantes ao desenvolvimento deste trabalho. Por fim, na seção 3.3 é introduzida uma breve discussão sobre as soluções abordadas nos trabalhos de interesse definidos.

3.1 Mapeamento Sistemático da Literatura

O mapeamento sistemático abordado neste capítulo é baseado na metodologia proposta por Petersen et al. (2015), onde seguindo a série de passos propostos, torna o estudo realizado passível de ser replicado por outros pesquisadores (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). A partir desta metodologia, pode-se citar cinco etapas das quais foram seguidas por este mapeamento:

1. Definição dos tópicos de interesse;
2. Execução da pesquisa com os tópicos de interesse para a identificação de estudos primários realizados;
3. Triagem inicial, empregando critérios de inclusão e exclusão considerando o resumo dos artigos;
4. Triagem final, considerando as seções de introdução, concepção do projeto e conclusão;
5. Extração dos dados e mapeamento.

Para a consulta dos trabalhos relacionados, primeiramente foram definidos tópicos de interesse a serem pesquisados, dos quais extraiu-se um conjunto de palavras como

candidatas a palavras chave, com o objetivo de aplicar estas em uma *string* de busca. Esta *string* agregou as seguintes palavras: *internet of things*, *distributed* e *complex event processing*.

A partir da definição das palavras chave, foi possível elaborar a *string* de busca usada para executar as consultas sobre as bases da: ACM Digital Library, IEEE Explore, ScienceDirect, Springer, Web of Science e Scopus; e assim obter-se os trabalhos relacionados com o tema de pesquisa de interesse. As strings de consulta foram aplicadas no início do segundo semestre de 2018, as quais podem ser vistas na Figura 2 incluindo as respectivas bases na qual foram executadas.

Base de Dados	**String de Busca**
ACM Digital Library	recordAbstract:(distributed AND ("internet of things" OR iot) AND ("event stream processing" OR "event processing" OR "complex event processing"))
Demais Bases	distributed AND ("internet of things" OR iot) AND ("event stream processing" OR "event processing" OR "complex event processing")

Figura 2 – *Strings* de buscas usadas.

Após a execução desta consulta preliminar, que se entende como a etapa de levantamento dos estudos primários relevantes, foram identificados 647 trabalhos de interesse onde este valor compreende-se da soma dos resultados obtidos em todas as bases de consulta.

Todas as buscas foram realizadas sobre os metadados dos artigos (título, resumo e palavras chave), porém, na data da execução deste trabalho, a ferramenta de busca disponibilizada pela base de dados Springer não oferecia suporte a este tipo de consulta, aplicando a *string* de busca sobre todas as seções dos artigos(Introdução, resultados, referencias...). Assim, este problema foi contornado da seguinte forma: primeiramente foi feita a exportação do resultado preliminar da busca para o formato CSV (o único suportado) resultando em 472 artigos. Após isto, fez-se uso da ferramenta CSV2Bib[1] para converter o arquivo CSV para .bib com o intuito de importar o resultado, para a ferramenta Zotero[2], o que permitiu a execução da *string* de busca sobre os metadados dos 472 artigos encontrados preliminarmente pela Springer, resultando em 6 documentos de interesse. A Figura 3 apresenta o percentual de publicações que cada uma das bases contribuiu para o montante final, já a Figura 4 apresenta um gráfico de barras contendo o número de artigos encontrados pela *string* de busca em cada uma das bases.

O gráfico 5 apresenta o número total de publicações de interesse por ano encontra-

[1] https://github.com/jacksonpradolima/csv2bib
[2] https://www.zotero.org/

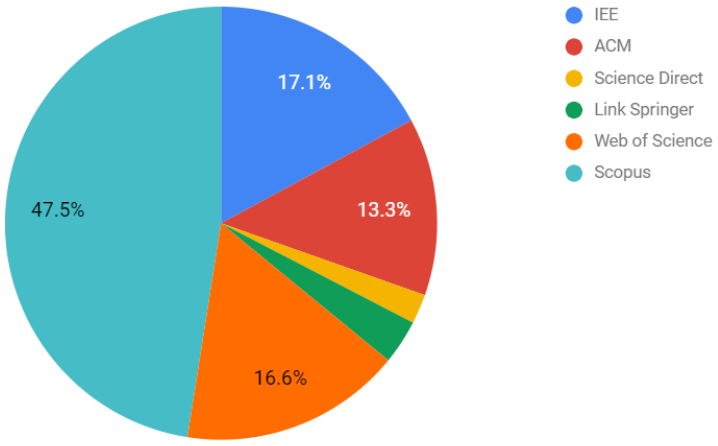

Figura 3 – Percentual de publicações encontradas por base.

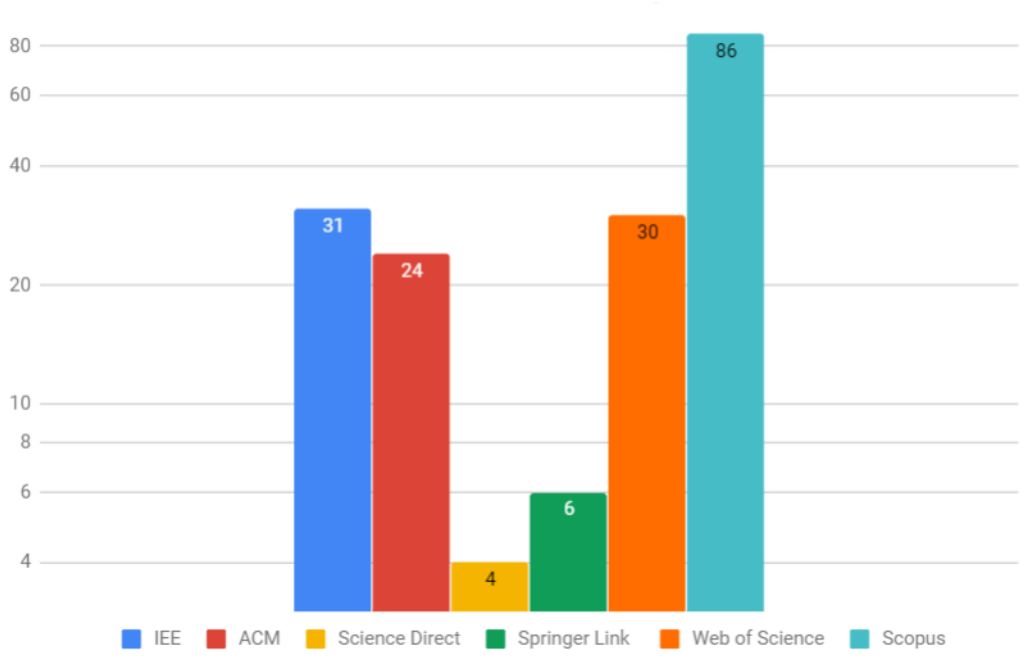

Figura 4 – Número de publicações encontradas por base.

das em cada uma das bases, onde para a representação do gráfico foram removidos todos os trabalhos duplicados. O eixo X apresenta o ano no qual os artigos foram publicados e o eixo Y apresenta o número total de publicações em relação ao ano. Pode-se perceber pela Figura 5 que a partir do ano de 2014 há um considerável aumento no número de pesquisas cientificas, e ainda um grande pico no ano de 2017, demonstrando assim pontos de interesse neste período.

Como as buscas foram realizadas no início do segundo semestre de 2018, o número de trabalhos encontradas foi inferior ao de 2017, porém considerando que o número de publicações se mantivesse crescendo constantemente durante o restante do ano, o número de artigos neste ano superaria facilmente o de 2017, destacando a relevância da área de pesquisa abordada. Os trabalhos relacionados de 2019 que

aparecem citados, são estudos que seriam publicados em revistas em sua edição seguinte no ano de 2019.

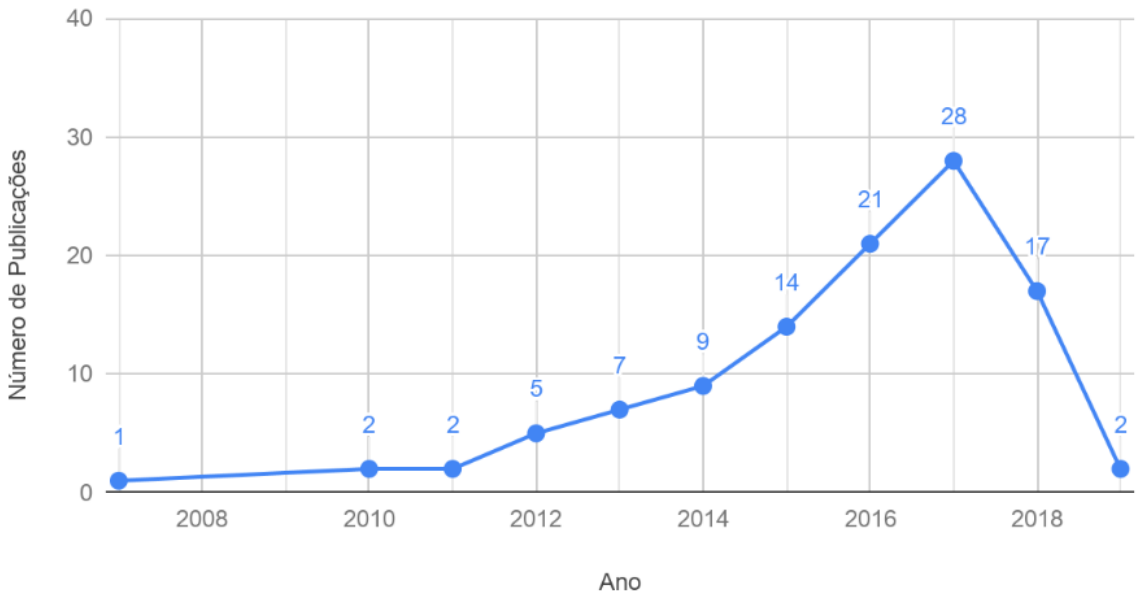

Figura 5 – Quantidade de publicações de interesse por ano.

3.1.1 Critérios de Inclusão e Exclusão

Após a seleção inicial realizada sobre as bases de dados, executou-se a triagem inicial sobre o resumo dos artigos, aplicando os seguintes critérios de inclusão e exclusão conforme a ordem apresentada a seguir:

- (E) Foi publicado antes de 2014;
- (E) Não é um artigo *full paper*;
- (E) Não estar em Inglês ou Português;
- (E) Indisponibilidade de acesso ao artigo completo;
- (E) Artigos que não apresentam avaliação da proposta;
- (I) Explora cenários de processamento de eventos em segurança da informação;
- (I) Explora conceitos de Computação Ubíqua;
- (E) O artigo não possui nenhum dos critérios de inclusão.

Para auxiliar na aplicação dos critérios de inclusão e exclusão fez-se a importação dos resultados preliminares das buscas na ferramenta Start[3]. Para isso usou-se os

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

arquivos .bib exportados pelas ferramentas das bases de busca, com exceção apenas da Spriger, onde usou-se o arquivo .bib exportado pelo Zootero, o qual foi gerado após a execução da consulta sobre os metadados, aplicada sobre o resultado preliminar da base.

Os critérios de exclusão e inclusão foram aplicados seguindo as seguintes ordens e etapas:

- **Remoção de trabalhos duplicados -** Alguns dos trabalhos retornados pela *string* de busca estavam indexados em ambas as bases de consulta, tornando necessária a execução de uma etapa de remoção dos mesmos, resultando em 74 trabalhos duplicados removidos.

- **Filtro por data -** O intervalo de interesse para a aplicação do filtro foi adotado com base no número de publicações por ano. Após o levantamento dos trabalhos de interesse, identificou-se 2014 como sendo o ano no qual o número de publicações aumenta consideravelmente, continuando a ascender até o pico máximo, no ano de 2017, como pode ser visto na Figura 5. Desta forma optou-se por eliminar todas as publicações que fossem anteriores ao ano de 2014, retirando um total de 26 artigos.

- **Artigos full paper -** Com o intuito de remover artigos que apresentassem apenas resumos superficiais, sem qualquer tipo de detalhamento sobre os trabalhos, ou que não tivessem cunho científico, foram removido os artigos que não se caracterizassem como *Full Paper* (livro ou capítulo de livro, introdução de anais, entre outros). No total foram excluídos 9 trabalhos.

- **Filtro por idioma -** Como as pesquisas foram realizadas sobre várias bases de dados com escopo global, as quais podem indexar trabalhos em diferentes línguas, optou-se por aplicar um filtro por idioma, visando remover qualquer trabalho que não esteja em Português ou Inglês (idiomas de total domínio do autor), removendo desta forma 1 artigo.

- **Indisponibilidade do artigo completo -** Dado que alguns dos estudos de interesse selecionados apresentaram apenas seus resumos e introdução disponíveis, não oferecendo de maneira simplificada a opção de obter-se o trabalho completo, optou-se por remover estes da pesquisa, sendo então 3 trabalhos excluídos.

- **Avaliação da proposta -** Foram removidos todos os artigos que não executaram algum tipo de teste ou estudo de caso das soluções propostas por seus trabalhos, excluindo-se assim 17 artigos.

- **Explora conceitos de segurança -** Este critério foi incluído devido a familiaridade do grupo com a analise e processamento de eventos da segurança da informação proveniente de estudos anteriores (ALMEIDA et al., 2019). Para a identificação destes eventos, é normalmente necessário de se executar a análise de um conjunto considerável de dados de rede, requisito este presente na IoT, onde há necessidade de se executar o processamento das informações geradas constantemente pelos dispositivos. Assim, selecionou-se os trabalhos que abordaram como estudo de caso a segurança da informação, esta aplicada a Computação Ubíqua ou que explorasse algum conceito desta. Com este critério de inclusão adicionou-se 4 trabalhos.

- **Explora conceitos de Computação Ubíqua -** Trabalhos que explorassem ou tivessem como foco de suas propostas a Computação Ubíqua foram selecionados incluindo assim 20 novos trabalhos.

- **Sem nenhum critério de inclusão -** Todos os trabalhos que não se enquadraram em nenhum dos critérios de inclusão foram removidos, excluindo desta forma 28 trabalhos da pesquisa.

Após execução da triagem inicial dos trabalhos, aplicando-se os critérios de inclusão e exclusão citados, sobre o resumo dos artigos, selecionou-se 24 documentos de interesse. O fluxo da aplicação destes critérios de exclusão pode ser visto na Figura 6, assim como o número total de trabalhos removidos por cada um dos critérios aplicados.

Após a triagem inicial dos trabalhos, executou-se a 4º etapa do mapeamento, que consiste da triagem final dos artigos de interesse, sendo realizada através da análise das seções de introdução, concepção e conclusão dos estudos. Os critérios considerados durante a execução da 4º etapa foram se os trabalhos tinham como estudo de caso a segurança da informação ou ainda se estes introduziam o uso de conceitos em computação ubíqua.

Na execução da etapa de triagem final dos estudos de interesse buscou-se selecionar os trabalhos que mais se assemelham com os temas de pesquisa abordados inicialmente na *string* de busca, incluindo aqueles que tivessem como tema solucionar desafios semelhantes aos abordados pelo grupo de pesquisa em que o autor deste trabalho esta inserido, isto é, desafios de distribuição e escalabilidade no processamento de eventos da IoT. Assim, com a execução da triagem final selecionou-se 5 dos 24 artigos identificados, estes sendo destinados para á análise completa de seu conteúdo e da extração das informações destes, os quais foram tomados como base para a elaboração deste trabalho.

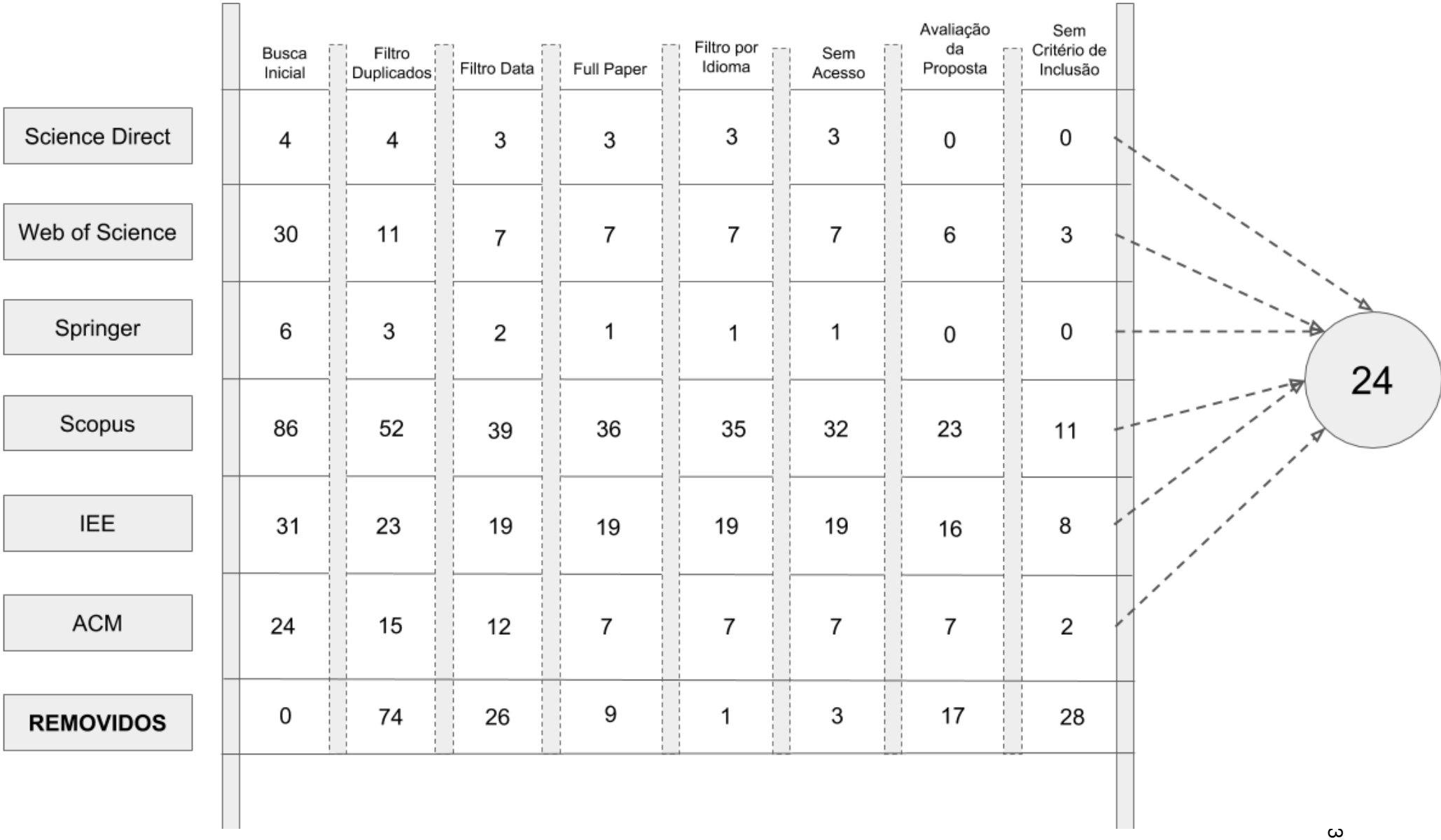

Figura 6 – Fluxo de triagem dos artigos.

3.2 Trabalhos Relacionados

Com a execução do mapeamento sistemático da literatura, foram identificados cinco estudos de interesse, os quais são tomados como base para o desenvolvimento desta pesquisa. Nas subseções a seguir, será dissertado sobre os aspectos mais pertinentes de cada uma das soluções propostas por estes trabalhos.

3.2.1 SAMURAI: A batch and streaming context architecture for large-scale intelligent applications and environments

Em (PREUVENEERS; BERBERS; JOOSEN, 2016) os autores apresentam o SAMURAI uma arquitetura de contexto em lote, *streaming* distribuído e multilocatária. Esta é baseada em processamento de eventos complexos, aprendizado de máquina e enriquecimento de contexto semântico, visando assim oferecer uma solução com escalabilidade horizontal para a IoT.

O sistema proposto foi desenvolvido seguindo os conceitos da arquitetura Lambda, esta dita pelos autores ser capaz de proporcionar o processamento de volumes de dados consideráveis, de maneira eficiente, escalável e tolerante a falhas, mantendo ainda uma interação responsiva com o usuário. A arquitetura Lambda projetada pelos autores é dividida em três camadas:

- **Camada de Lote -** os dados são recebidos por meio de atualizações periódicas, estas resultantes da execução de operações de redução executadas sobre o conjunto de dados principal, por exemplo, um armazenamento imutável no HDFS[4](*Hadoop Distributed File System*). O processamento dos dados nesta camada é de alta latência levando até horas para serem concluídos.

- **Camada de Velocidade -** processa atualizações incrementais dos dados, sendo estas de baixa latência, ocorrendo na ordem de segundos. Esta camada gerencia apenas os dados novos recebidos, produzindo visualizações em tempo real que compensam as atualizações de alta latência da Camada Provedora.

- **Camada provedora -** responsável por expor a visualização dos dados pré-computadas para atender a consultas ad-hoc com baixa latência.

Para o desenvolvimento da arquitetura proposta, os autores fizeram uso da ferramenta Apache Kafka[5], sendo esta usada como mecanismo para o provimento da comunicação, efetuando a distribuição dos dados em um padrão de publicação e assinatura. Para executar o processamento dos dados na camada de lote, fez-se uso do

[4]https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[5]https://kafka.apache.org/

Apache Spark[6], o qual é encarregado de executar algoritmos distribuídos que necessitam processar grandes volumes de dados de maneira escalável.

Já para o desenvolvimento da Camada de Velocidade, os autores incorporaram a opção de selecionar o uso individual do Apache Spark ou do Apache Storm[7] em conjunto com Esper[8] na execução do processamento de eventos complexos, esta integração é permitida pela tipagem das tuplas do Storm corresponderem ao formato dos eventos do Esper.

O SAMURAI também incorpora a biblioteca de aprendizado de máquina Weka[9] visando assim permitir a detecção de eventos por meio do uso de técnicas de IA. A arquitetura desenvolvida expõe os principais recursos do Weka por meio de uma API RESTful, permitindo que cada aplicativo registre um ou mais modelos, onde cada modelo possui um conjunto de atributos e um classificador específico. Como alternativa ao Weka, o sistema também permite o uso da biblioteca de aprendizado de máquina MLlib[10] do Spark, que apesar de oferecer menos funcionalidades, apresenta um desempenho superior quando aplicada sobre grandes fluxos de dados.

3.2.2 A Distributed Complex Event Processing System Based on Publish/Subscribe

No trabalho de (WANG; SHANG, 2019) é apresentada uma proposta de arquitetura de processamento de eventos distribuído com abordagem de publicação e assinatura. O processamento de eventos complexos neste trabalho foi modelado como um DAG (*Directed Acyclic Graph*), onde cada nó representa um mecanismo CEP e suas bordas determinam o fluxo que os dados percorrem. Diferentes nós CEP executam operações de processamento distintas, onde nós CEP de baixo nível enviam eventos para nós CEP de alto nível como um produtor e consumidor.

O fluxo de comunicação desta arquitetura foi projetado com a ferramenta Apache Kafka, onde cada nodo de processamento consome de um tópico e publica os dados já processados em um tópico seguinte, que por sua vez será consumido por algum outro nodo de processamento. Um exemplo funcional desta arquitetura seria onde um nodo1 consome os dados brutos recebidos em um tópico nomeado base-topic, este nodo1 produzirá eventos para um segundo tópico nomeado node1-topic, que por sua vez será subscrito pelos nodo2 e nodo3 que publicaram os dados no topico node2-topic e o node3-topic respectivamente. Por fim, um nodo4 irá consumir os dados do node2-topic e node3-topic e publicará seu resultado no tópico nomeado final-topic.

A arquitetura do nodo de processamento de eventos complexos possui quatro mó-

[6] https://spark.apache.org/
[7] http://storm.apache.org/
[8] http://www.espertech.com/esper/
[9] https://www.cs.waikato.ac.nz/ml/weka/
[10] https://spark.apache.org/mllib/

dulos, onde primeiramente o módulo Input Adapter busca os dados de um tópico Kafka e repassa estes aos módulos Event Statement e o Event Processor os quais são responsáveis por descrever a lógica de processamento dos eventos, incluindo o tipo de evento a ser consumido e as regras de agregação dos eventos. Para a execução do processamento dos dados, o Esper foi implementado neste módulo, sendo usado a linguagem EPL(*Event Processing Language*) suportada por este para executar a descrição das regras de processamento. Por fim, o componente Output Adapter publica os dados já processados em um tópico Kafka previamente definido.

3.2.3 CEML: Mixing and moving complex event processing and machine learning to the edge of the network for IoT applications

Em (SOTO et al., 2016) é apresentado o CEML (Complex Event Machine Learning) um *framework* capaz de fornecer técnicas de aprendizado de máquina para IoT, trabalhando com eventos de forma distribuída e escalável. Os autores citam ainda que esta solução pode ser implementada em nuvem, sendo assim capaz de trabalhar sobre grandes fluxos de dados.

A arquitetura do sistema foi projetada em um padrão de publicação e assinatura, onde os clientes (dispositivos, aplicativos ou serviços) se comunicam publicando mensagens assíncronas nas quais os LAs (*Learning Agents*) estão inscritos. Estes modelos de arquitetura permitem que o sistema distribua os dados em nós de processamento externos ou internos, dependendo das necessidades do ambiente.

Cada tarefa a ser executada pela arquitetura é transformada em instruções, das quais serão implantadas em um ou mais LA. Quando estas instruções forem designadas a mais de um LA, as subscrições dos tópicos serão entre os próprios LAs, distribuindo assim o processamento das tarefas em diferentes serviços. Caso a distribuição das instruções seja designada a apenas um LA, a subscrição será apenas interna ao sistema.

Já heterogeneidade dos dados, é abordada da seguinte forma pela arquitetura proposta: primeiramente os diferentes fluxos de informações são divididos em tópicos distintos; em seguida, o tratamento dos dados necessário(transformações, junções...) é executado por regras previamente definidas; por fim, os eventos são agrupadas em fluxos internos ou publicadas externamente em tópicos correspondentes. Estes fluxos são gerenciados pelo módulo de manipuladores.

O modelo de comunicação do sistema foi implantado com o uso do Mosquito[11] MQTT Broker e o cliente Java Paho[12], os quais gerenciam toda a comunicação do envio e recebimento dos dados.

Dentre os outros módulos do sistema implementado pelos autores, pode-se ainda

[11] https://mosquitto.org/

[12] https://www.eclipse.org/paho/clients/java/

destacar:

- **Feeders -** componentes encarregados de gerenciar os tipos de cargas úteis de dados recebidos, os quais podem ser classificados como: dados/eventos, instruções e solicitação de aprendizado.
- **Learning Agents -** módulo responsável por implementar toda lógica de processamento de eventos complexos, este é executado com o auxílio da ferramenta de processamento de dados Esper.

Dentro de um LA, os dados são ainda categorizados logicamente em três tipos de fluxos distintos: fluxos de aprendizado, sendo este a entrada dos Modelos, os quais são usados para a validação do sistema; fluxos de implantação, são os fluxos a serem usados quando o Modelo já estiver validado; fluxos auxiliares, são usados como nós de processamento interno pelos fluxos de aprendizado ou de implantação, sua visibilidade é restrita ao próprio mecanismo CEP.

O modelo atual desta arquitetura desenvolvida pelos autores, tem a capacidade de disponibilizar uma interface para problemas de classificação, usando para isto, várias implementações do *framework* de inteligência artificial Weka.

3.2.4 Semantic IoT Middleware-enabled Mobile Complex Event Processing for Integrated Pest Management

No trabalho de (NOCERA et al., 2017) é apresentada uma infraestrutura inteligente, capaz de processar dados de fontes heterogêneas executando parcialmente o processamento de eventos complexos em dispositivos móveis de forma distribuída com o intermédio de um *middleware* da IoT.

A infra-estrutura projetada pelos autores consiste basicamente de duas partes:

- **Servidor Back-End -** neste servidor um arquivo JSON(*Javascript Object Notation*) contém a definição das regras sobre os dados da ontologia extraídos, bem como as ações a serem executadas caso alguma regra for acionada. Em caso de ativação de uma das regras o módulo Observer notifica o NodeJS[13] que encaminha os dados da regra ativada, juntamente com as informações sobre sensores e variáveis ao Redis[14] para que possam ser publicados. Os consumidores assinam o Redis Message Broker e são notificados da mensagem, a qual é encaminhada com os dados do sensor, variável e regra(extraída do arquivo de configuração) para o módulo de Mecanismo de Regras o qual executa tarefas pré-configuradas.
- **Dispositivo Móvel -** primeiramente o módulo Sensor Adapter desta arquitetura se conecta aos sensores, coletando os dados e convertendo-os para um formato

[13]https://nodejs.org/
[14]https://redis.io/

de evento a ser usado pelos demais módulos deste sistema. Todas as informações já formatados são enviados para o componente Event Stream Management, que despacha o fluxos de eventos para o CEP Engine responsável por processar as regras CEP com o uso do Esper, caso estas sejam ativadas, a informação do evento é repassada ao módulo Action Handler que executa alguma ação predefinida no dispositivo móvel(notificação, alertas...) e por fim envia os dados ao Redis para que este possa buscar na ontologia informações baseado no evento ocorrido.

Todos os dados usados pelo sistema são provenientes de leituras de sensores de ar, temperatura, umidade e emissão de poeira. Estes sensores são controlados por um Arduíno o qual repassa as informações a uma RaspberryPi que fica responsável por enviar e receber estes dados para o Servidor Back-End local, que por sua vez faz a transição destes para o DeviceHive[15] um *middleware* da IoT que faz a padronização dos dados a serem usados pela arquitetura.

3.2.5 Parallel big data processing system for security monitoring in Internet of Things networks

Em (KOTENKO; SAENKO; KUSHNEREVICH, 2017) é proposto um sistema para processamento paralelo de dados de segurança destinado à implementação em ambientes IoT. O sistema implementado aplica estratégias CEP, sendo capaz de pré-processar dados em tempo real, executando normalização, filtragem, agregação e correlação de dados processados em tempo de execução.

A arquitetura projetada pelos autores inclui cinco componentes:

- **Coletor de Dados -** este é responsável por organizar os dados distribuídos recebidos e armazena-los no componente Armazenamento de Dados. Além disso, este módulo contém um gerador de fluxo de dados de teste, esse usado para avaliar a eficácia do sistema;

- **Armazenamento de Dados -** os dados recebidos por este componente são armazenados em HDFS sendo esta uma maneira alternativa para executar o armazenamento de dados na IoT que sistemas tradicionais de gerenciamento de banco de dados SQL. Após os eventos de segurança serem armazenados, estes são empacotados em fluxos de dados e enviados ao componente de Agregação de Dados com os seguintes campos: endereço IP(*Internet Protocol*) de origem, porta de origem, IP de destino e porta de destino;

- **Agregação de Dados -** executa o processamento do fluxo de dados recebido, usando para isto o Hadoop ou o Apache Spark. Para que a identificação dos

[15]https://devicehive.com/

eventos de segurança possa ser executada posteriormente, este módulo calcula medidas estatísticas dos dados (mínimo, máximo, média, moda, quantis ...). Por fim os resultados são registrados no HDFS e transferidos ao componente Normalização e Análise de Dados;

- **Normalização e Análise de Dados -** inicialmente mente este componente executa a conversão dos dados para o formato CSV(*Comma-Separated Values*), em seguida é realizada a análise dos dados recebidos, consistindo na identificação de incidentes de segurança por meio do uso de regras CEP predefinidas para correlacionar os eventos. Por fim os resultados deste módulo são armazenados no HDFS;

- **Visualização de Dados -** permite a visualização dos eventos de segurança detectados em um gráfico previamente selecionado pelo administrador.

Por fim, os autores executaram a validação da proposta, por meio de uma análise de desempenho comparativa, com o uso das ferramentas Hadoop e Apache Spark para o processamento de eventos complexos da arquitetura proposta. Os resultados obtidos pelos autores mostram que o Hadoop se comporta de forma satisfatória em sistemas com recursos computacionais limitados. Já quando implementado com o Apache Spark, o sistema aumenta seu desempenho em cerca de dez vezes, caso este tenha uma quantidade suficiente de memória RAM(*Random Access Memory*) disponível.

3.3 Discussão dos Trabalhos Relacionados

Após a análise dos trabalhos relacionados executada durante a última etapa do mapeamento sistemático, percebeu-se que grande parte das pesquisas aplica estratégias similares, baseadas em soluções ad-hoc, para a comunicação visando distribuição e processamento dos eventos. No caso do processamento de eventos, todos os trabalhos empregaram estratégias de CEP, aplicando em geral Esper, em conjunto ou não com outras ferramentas, visando assim permitir o tratamento em fluxo dos dados.

Porém, apesar de todas os trabalhos terem como área de foco a IoT, nenhuma das soluções apresentadas se preocupava com o consumo de rede necessário para a execução da distribuição dos eventos, onde muitos ainda aplicavam estratégias negligentes quanto a distribuição dos dados, as quais podem vir a gerar um consumo de rede diversas vezes maior do que o real necessário.

Identificou-se também que apenas em (NOCERA et al., 2017) e (SOTO et al., 2016) é apresentado soluções que levassem em consideração a heterogeneidade da IoT, onde os autores optaram por acoplar sua arquitetura proposta a um *middleware* ou

implementar regras de filtragem e transformação respectivamente, com o objetivo de contornar este desafio.

Percebeu-se ainda, que apesar de todos os trabalhos proporem uma arquitetura distribuída de processamento para a IoT, apenas (PREUVENEERS; BERBERS; JOOSEN, 2016) preocupou-se em executar testes de validação quanto a capacidade da arquitetura distribuir e processar dados, porem este não apresentou nenhum teste de consumo de recursos computacionais, algo de suma importância para a verificação da aplicabilidade da proposta em *hardware* com poder computacional reduzido, estes comumente presentes na IoT.

Dadas as informações levantadas a partir da análise dos trabalhos identificados no mapeamento sistemático, identificou-se a carência de arquiteturas aptas a lidarem com a heterogeneidade da IoT as quais sejam competentes em executar o processamento de um volume considerável de eventos de forma distribuída não negligenciando o consumo de banda gerado por este tráfego de informação. Assim no âmbito das motivações citadas, este trabalho tem como objetivo desenvolver uma arquitetura de processamento distribuído de eventos para a IoT com escalabilidade e que esteja apta a lidar com a heterogeneidade destes ambientes, aplicando ainda estratégias que visem aprimorar a eficiência do consumo de largura de banda, com o objetivo de proporcionara aplicabilidade desta arquitetura em meios que apresentem alguma limitação de rede, seja esta uma restrição por *links* de comunicação saturados, onde perdas constantes de pacotes são recorrentes (FALL; STEVENS, 2011), ou por uma rede que possua limitações quanto à largura de banda reduzida.

A Figura 7 apresenta uma análise comparativa entre os trabalhos selecionados pelo mapeamento sistemático e o EXEHDA-DEP, a nova arquitetura proposta por este trabalho, onde os campos selecionados para a comparação foram:

- **Comunicação:** exibe qual foi o padrão arquitetural adotado para executar a comunicação entre os nodos de processamento dos eventos.
- **Estratégia de EP:** apresenta qual estratégia de processamento de eventos o artigo adota em sua proposta. Esta coluna pode apresentar dois valores CEP ou ESP.
- **Ferramentas de EP:** nesta coluna é apresentado as ferramentas adotadas para executar o processamento de eventos em cada uma das propostas analisadas.
- **Tratamento de heterogeneidade:** exibe qual estratégia adotada no trabalho para o tratamento da heterogeneidade da IoT. Esta coluna pode exibir os seguintes valores: o nome do *middleware* usado para executar esta tarefa; “X” para caso a solução apresentada implemente uma estratégia própria para o tratamento da heterogeneidade; “-” caso o trabalho não aborde este quesito.

- **Segurança:** nesta coluna é identificado se o trabalho tem como estudo de caso a aplicação em segurança da informação ou ainda se a solução permite o uso de alguma técnica de segurança. Se alguma destas condições for satisfeita, o valor da coluna será “X”, caso contrário será “-”.

- **Validação da Operação Distribuída:** identifica se a solução proposta apresenta testes de validação da capacidade da arquitetura distribuir o processamento dos eventos, identificando os ganhos na taxa de processamento com o aumento dos nodos de trabalho. Esta coluna apresenta o valor “X” quando a solução proposta apresentar testes satisfatórios para a validação da distribuição, caso contrário o valor será “-”.

- **Avaliação do Consumo de Rede:** exibe se a arquitetura proposta no trabalho é capaz de fornecer um consumo de rede estável, isto é, se a distribuição do processamento não gera uma enorme carga sobre o tráfego de rede. O valor “X” nesta coluna representa que a solução é capaz de fornecer a distribuição do processamento sem impactar muito no consumo de rede. Caso isto não ocorra, o valor da coluna será “-”.

	Comunicação	Estratégia de EP	Ferramentas de EP	Tratamento da Heterogeneidade	Segurança	Validação da Operação Distribuída	Avaliação do Consumo de Rede
PREUVENEERS, 2016	*Publish/ Subscribe*	CEP	Spark, Strom, Esper	-	X	X	-
WANG, 2019	*Publish/ Subscribe*	CEP	Esper	-	-	-	-
SOTO, 2016	*Publish/ Subscribe*	CEP	Esper	X	-	-	-
NOCERA, 2017	*Publish/ Subscribe*	CEP	Esper	DeviceHive	-	-	-
KOTENKO, 2017	*Publish/ Subscribe*	CEP	Spark, Hadoop	-	X	-	-
EXEHDA-DEP	*Publish/ Subscribe*	CEP	Esper,Spark	EXEHDA	X	X	X

Figura 7 – Comparativo entre os artigos selecionados.

4 ESCOPO DE DESENVOLVIMENTO

Neste capítulo será dissertado sobre o EXEHDA, um *middleware* voltado para a IoT e uma de suas implementações, o modelo arquitetural EXEHDA-SA. Ambas as tecnologias citadas são tomadas como base para o desenvolvimento e proposta do EXEHDA-DEP.

A seção 4.1 descreve as principais funcionalidades do *middleware* EXEHDA e de sua arquitetura modular bem como dos cenários de aplicação para esta ferramenta. Já a seção 4.2 disserta sobre o modelo arquitetural EXEHDA-SA, onde são apresentados as funcionalidades atribuídas a cada um dos módulos desta arquitetura e seus principais objetivos.

4.1 EXEHDA

Um *middleware* pode ser considerado como uma camada mediadora entre duas aplicações, a qual gera um canal lógico de comunicação entre estas, possibilitando que o *middleware* atue traduzindo a comunicação estabelecida, abstraindo qualquer tipo de incompatibilidade ou complexidade existente entre estes dois meios. Desta forma, pode-se definir *middleware* como uma camada de *software* responsável por abstrair a heterogeneidade de um meio, facilitando o uso dos recursos disponibilizados neste meio a aplicações de terceiros (NGU et al., 2016).

Existem diferentes tipos de *middlewares* disponíveis, exemplos comuns são: os de banco de dados, os quais abstraem a complexidade da manipulação das informações aos seus clientes; os voltados para a IoT, que normalmente atuam abstraindo a heterogeneidade destes ambientes, dentre estes *middlewares* pode-se citar o EXEHDA (*Execution Environment for Highly Distributed Applications*) sendo este direcionado principalmente às aplicações distribuídas, móveis e cientes de contexto, o qual atua fornecendo serviços a estas aplicações (NGU et al., 2016).

O EXEHDA opera criando e gerenciando ambientes ubíquos formados por células de execução distribuídas e promovendo a computação sobre este ambiente heterogêneo (LOPES et al., 2014). Na Figura 8 é ilustrado um ambiente Ubíquo, onde diversos

dispositivos interagem, sendo estes gerenciados pelo EXEHDA, agregados de forma lógica em células de execução (MACHADO et al., 2017).

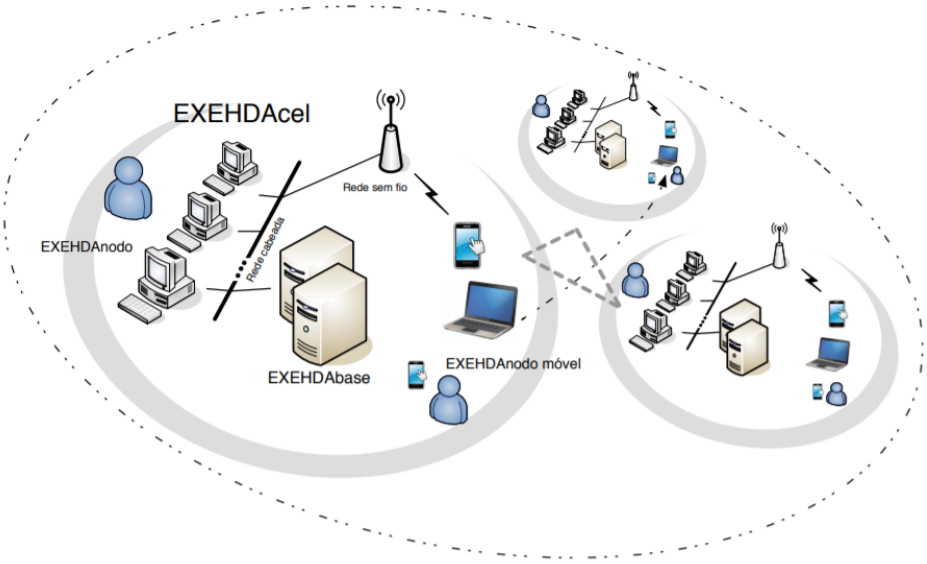

Figura 8 – Gerenciamento de um Ambiente Ubíquo pelo EXEHDA.
Fonte: (ALMEIDA et al., 2019).

A arquitetura do EXEHDA consiste da composição de diversos módulos, os quais são responsáveis por prover algum tipo de serviço. Dentre estes módulos destaca-se o Subsistema de Reconhecimento de Contexto, o qual é responsável pela síntese e correlação dos eventos capturados no ambiente (MACHADO et al., 2017). A Figura 9 apresenta uma visão geral da arquitetura EXEHDA.

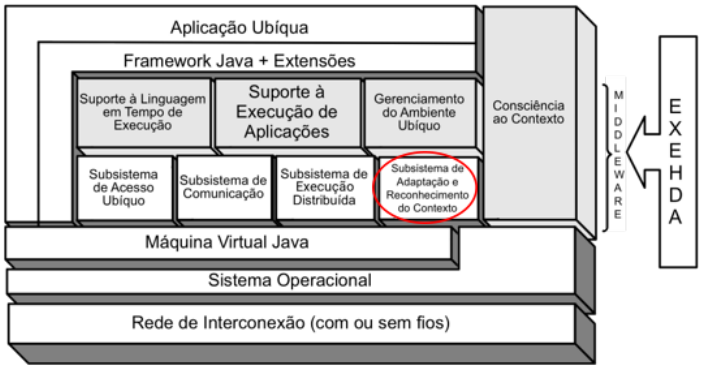

Figura 9 – Arquitetura EXEHDA.
Fonte: (YAMIN et al., 2005).

Existem diversos projetos concebidos sobre o EXEHDA: EXEHDA-HM(*Execution Environment for Highly Distributed Applications - Hybrid Modeling*) (MACHADO et al., 2017), EXEHDA-SA(*Execution Environment for Highly Distributed Applications - Situational Awareness*) (ALMEIDA et al., 2019), entre outros. Contudo, apesar do EXEHDA

ser um *middleware* direcionado a ambientes distribuídos, suas implementações apresentam a compreensão dos eventos em escopo local, o que pode vir a gerar um gargalo de processamento em ambientes IoT altamente distribuídos.

4.2 EXEHDA-SA

O EXEHDA-SA consiste de um modelo de arquitetura distribuída de alto nível, o qual foi projetado com base nas formalizações definidas pelo *middleware* EXEHDA. Este modelo arquitetural apresenta recursos de reação dinâmica e personalizável para interagir com a infraestrutura subjacente dos ambientes da IoT. Outra funcionalidade disponibilizada por este modelo é a capacidade de coletar dados em um ambiente heterogêneo como o da IoT, padronizando e abstraindo estas informações coletadas de modo a facilitar o uso destes dados (ALMEIDA et al., 2019).

A concepção do EXEHDA-SA é baseada em um conjunto de premissas as quais visam favorecer a este o suporte das demandas da IoT de escalabilidade, flexibilidade, autonomia e heterogeneidade, dentre estas premissas abordadas, se pode destacar: a extensibilidade modular para protocolos de coleta e comunicação de eventos; módulos extensíveis de processamento de eventos, os quais podem ser aplicados em estratégias híbridas, tais como aprendizado de máquina e estrategias baseadas em regras; recurso de reação dinâmica, personalizável e conectável os quais permitem a interação com infraestruturas subjacente dos ambientes de IoT (ALMEIDA et al., 2019).

Assim como no EXEHDA o EXEHDA-SA segue o mesmo mapeamento da infraestrutura física de três abstrações básicas, as quais são tomadas como base para a composição do ambiente onipresente:

- **EXEHDAnode**: consiste dos dispositivos de processamento, os quais são encarregados da execução da aplicação. Uma subcategoria pertencente a esta categoria é o EXEHDAnode mobile, o qual de diferencia de um EXEHDAnode pela alta portabilidade onde este normalmente possui uma capacidade mais restrita sendo normalmente detentor de uma interface de comunicação sem fio, onde nesse caso, é integrado a uma célula onde seu ponto de acesso é subordinado;
- **EXEHDAbase**: é o meio de comunicação entre os EXEHDAnodes, o qual é encarregado por prover todos os serviços essenciais do ambiente, onde estes serviços fornecidos podem ser distribuídos entre vários dispositivos de processamento, com o intuito de se obter um ambiente com escalabilidade;
- **EXEHDAcel**: encarregado por delimitar a área de operação do EXEHDAbase, as definições principais tomadas para a especificação do escopo de uma célula consiste: escopo institucional, proximidade geográfica e custos de comunicação.

Seguindo esta estrutura de projeto modular do EXEHDA, o EXEHDA-SA adquire flexibilidade, onde usuários podem adicionar ou remover módulos dependendo de suas necessidades. A Figura 10 apresenta uma visão geral da arquitetura EXEHDA-SA.

Dentre os módulos presentes na arquitetura EXEHDA-SA pode-se destacar:

- Percepção: responsável por identificar fontes relevantes de eventos no dispositivo em que uma instância do Collector está operacional;
- Pré-processamento: projetado para realizar a separação do evento em campos executando a normalização dos eventos coletados;
- Compreensão: consiste na correlação de eventos com base nos campos dos eventos obtidos no módulo de Pré-processamento;
- Projeção: evita reincidências de situações indesejadas previamente identificadas durante a etapa de compreensão;
- Interface Web: uma interface administrativa onde um administrador de rede pode analisar os eventos complexos identificados e definir novas regras e configurações;
- Repositório: onde são armazenadas todas as configurações de regras definidas pelo administrador, incluindo também dados de eventos identificados.

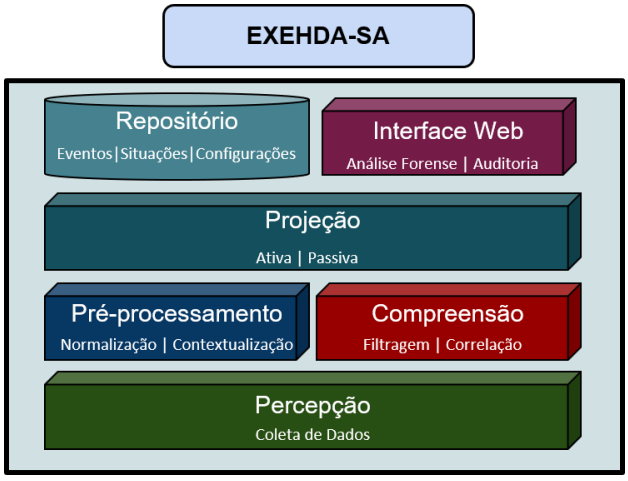

Figura 10 – Arquitetura EXEHDA-SA.

Nas implementações disponíveis do EXEHDA-SA as informações coletadas são comumente disponibilizadas no formato JSON, contendo diferentes campos padronizados e gerados pelo EXEHDA-SA, os quais podem ser modificados conforme a necessidade do usuário. Dentre os campos normalmente disponibilizados em implementações do EXEHDA-SA pode-se citar: TAG, o qual identifica um fluxo de dados

relacionados, como por exemplo, o fluxo de informações vinda de roteadores; risco, campo o qual identifica o nível de risco de segurança ao qual o dado remete; src_ip, endereço IP da origem do dado (ALMEIDA et al., 2019).

5 EXEHDA-DEP: CONCEPÇÃO E TECNOLOGIAS

Neste capitulo será apresentada a metodologia abordada para solucionar os desafios levantados durante o mapeamento sistemático da literatura.

Na seção 5.1 é apresentado o modelo de comunicação projetado, o qual se baseia no padrão publicação e assinatura, o mesmo aplicado pelos trabalhos relacionados citados na seção 3.2, porém neste novo modelo projetado, este padrão é reinventado de forma a aplicar estrategias que visem permitir a execução do EXEHDA-DEP em ambientes com limitações de comunicação na rede, sejam estas ocasionadas por *links* de comunicação saturados ou por conexões de baixa velocidade.

Na seção 5.2 é apresentado o modelo desenvolvido de uma arquitetura de processamento de eventos complexos, a qual possui a capacidade de processar eventos de forma distribuída com escalabilidade.

A seção 5.3 apresenta as principais ferramentas e tecnologias empregadas para o desenvolvimento da arquitetura proposta por este trabalho. Por fim, na seção 5.4 é apresentada a nova arquitetura de processamento distribuído proposta.

5.1 Modelo de Comunicação

A estratégia de comunicação citada nesta seção consiste na forma como os dados a serem processados são distribuídos entre os nodos de processamento. A grande maioria dos trabalhos relacionados identificados na seção 3.2 se baseava em um modelo de publicação e assinatura, onde os dados a serem processados são publicados em um tópico central, o qual é assinado por vários nodos de processamento que recebem estes mesmos dados e executam trabalhos de processamento distintos entre si. Esta estratégia para a distribuição dos dados a serem processados pode ser vista na Figura 11.

A estratégia citada pode ser identificada como de fácil implementação, garantindo uma alta escalabilidade e permitindo que um novo nodo de processamento seja adicionado simplesmente com a subscrição do tópico central. Porém estratégias com esta abordagem apresentam como principal desvantagem a sobrecarga gerada sobre os

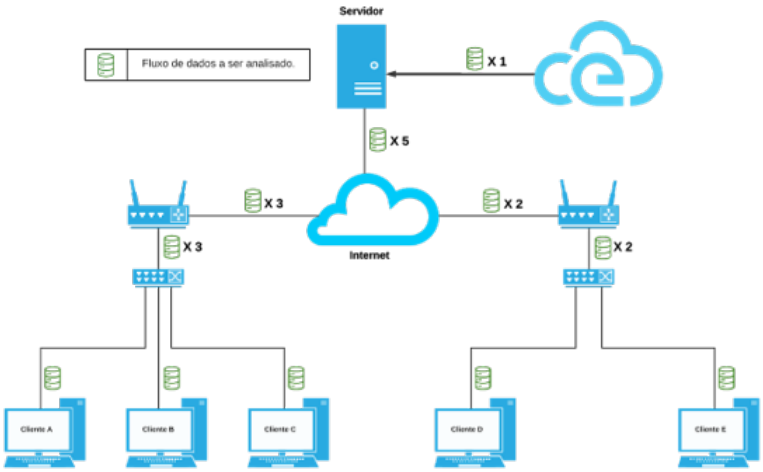

Figura 11 – Fluxo de comunicação padrão.

links de comunicação, já que todos os nodos de processamento subscritos no tópico central rebem os mesmos dados a serem processados, com o diferencial de estes executarem tarefas distintas sob estas informações.

Deste modo, pode-se verificar que a estratégia citada adiciona uma sobrecarga na rede de acordo com o número de nodos de processamento, quanto maior a quantidade de nodos processando dados, maior será a largura de *link* necessária para trafegar as informações a serem distribuídas, onde pode-se expressar esta sobrecarga adicionada ao *link* de comunicação por meio do produto do volume total dos dados a ser processado, pela quantidade de nodos de processamento existentes. A equação 1 demonstra uma fórmula para o cálculo do tráfego total de rede exigido pelo modelo de comunicação citado.

$$Trafego = Dados \times Nodos \tag{1}$$

Outra desvantagem derivada do recebimento total do conjunto de dados pelos nodos de processamento, consiste que estes potencialmente podem estar recebendo informações que não se aplicam a suas tarefas de processamento, as quais são simplesmente ignoradas. Este processo de ignorar estas informações, pode ser extremamente custoso quando é considerado grandes volumes de dados a serem analisados (HU; HONG; CHEN, 2017). Assim pode-se considerar que nodos de processamento tem parte do seu tempo de trabalho ocupado pela tarefa de ignorar dados inúteis para suas tarefas, gerando considerável perda de recursos computacionais nestes nodos (tempo de processamento, memória RAM, consumo energético).

A estratégia de comunicação desenvolvida por este trabalho consiste em uma modificação do padrão publicação e assinatura adotado pelos trabalhos relacionados citados na seção 3.2, visando com estas modificações solucionar os problemas citados. A estratégia consiste em adotar um padrão de classificação para o fluxo de dados a ser

analisado antes destes serem publicados no tópico que os nodos subscrevem. Esta classificação consiste em subdividir logicamente o fluxo principal de dados em outros fluxos menores, por meio do acréscimo de uma chave identificadora a estes, de modo a permitir o direcionamento destes dados para cada um dos nodos, baseando-se na distribuição destas chaves identificadoras. A Figura 12 apresenta uma comparação em escopo teórico do consumo de rede exercido pelo método atual aplicado nos trabalhos relacionados identificados com o novo método proposto, esta mesma comparação é executada também em um escopo prático, onde esta figura é reapresentada com os dados práticos obtidos na seção 6.4.

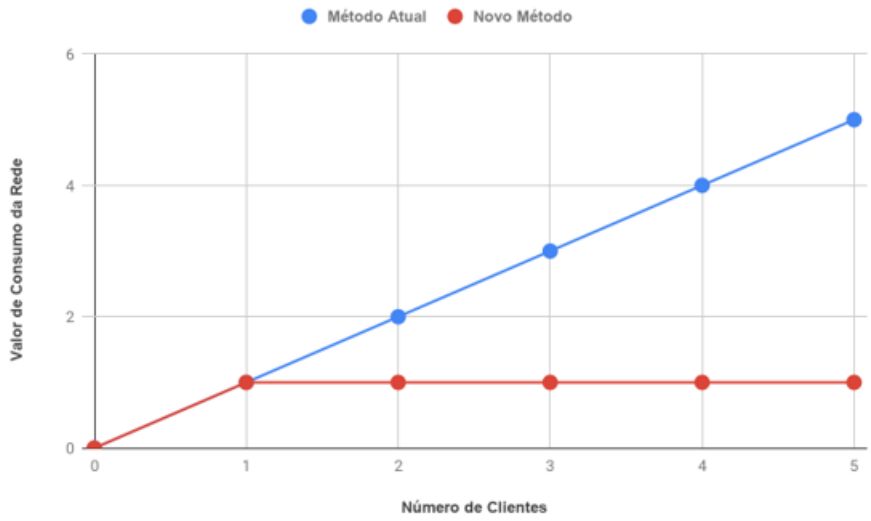

Figura 12 – Comparativo de consumo de rede.

Em um exemplo, com dois nodos de processamento, nodoA e nodoB, o fluxo original de dados poderia ser subdividido em outros dois A e B de modo que o nodoA irá receber os dados do fluxo A e o nodoB os dados do fluxo B. Esta estratégia permite evitar a sobrecarga de rede gerada pelo envio de todo o conjunto de dados para cada um dos nodos. A Figura 13 ilustra o processo de comunicação seguindo o novo padrão proposto.

A subdivisão do fluxo de eventos pode ser executada de forma a manter os eventos que necessitem ser analisados juntos, em uma mesma categoria, centralizando as regras de processamento com seus respectivos eventos a serem processados.

5.2 Modelo de Processamento

Nesta seção será apresentada a estratégia de processamento de eventos adotada pelo presente trabalho, a qual visa proporcionar uma arquitetura de processamento de eventos com escalabilidade, sendo voltada a ambientes altamente distribuídos que necessitem processar grandes volumes de eventos, características estas presentes

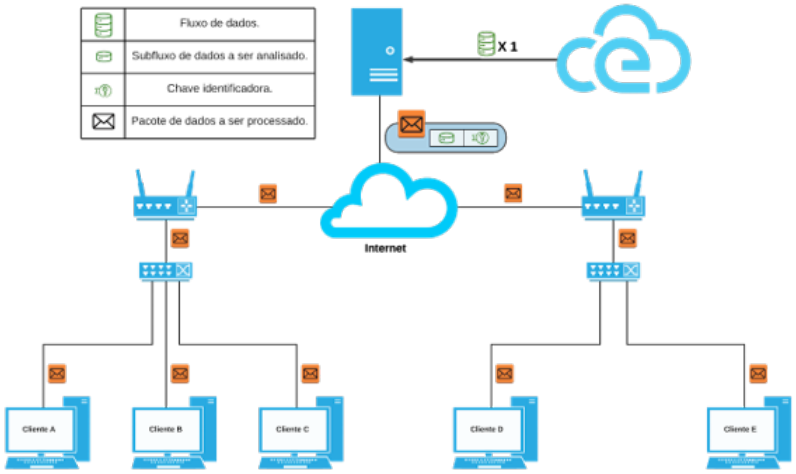

Figura 13 – Fluxo de comunicação proposto.

na IoT.

A estratégia de processamento de eventos adotada por este trabalho consiste na distribuição da carga de eventos para diferentes nodos de processamento, criando um modelo arquitetural descentralizado. Esta descentralização visa evitar gargalos de processamento adicionados a apenas uma máquina ou a um link de comunicação, proporcionando também uma maior flexibilidade na implementação, se adaptando a ambientes altamente distribuídos. A descentralização do processamento também proporciona a versatilidade para o controle da capacidade de processamento, onde caso seja necessário um *throughput* maior, a simples adesão de um novo nodo de processamento adiciona o aumento do mesmo. Tais nodos de processamento independem de arquiteturas com alto poder computacional, onde mesmo nodos executando sob máquinas de baixo custo, com poder computacional limitado, agregam na capacidade total de processamento disponibilizada no *cluster* de nodos, o que proporciona a vantagem de não necessitar o uso de máquinas especiais com grande capacidade de processamento para a execução.

A arquitetura distribuída proposta foi modelada com três tipos distintos de nodos:

- **Nodo de Pré-processamento:** responsável por modelar os eventos recebidos para um formato de mais fácil manipulação aos outros nodos de execução. Após a execução desta modelagem, o nodo de pré-processamento envia estes eventos ao nodo Broker, podendo executar a compactação destes eventos antes do envio, com o intuito de reduzir o consumo de rede gerado.

- **Nodo Broker:** este nodo é responsável por armazenar os eventos recebidos pelo pré-processamento e distribui-los sob demanda aos nodos de processamento. Este nodo também armazena os resultados dos eventos já processados, os disponibilizando para consulta das partes interessadas (administrador

de rede, ferramentas de terceiros...).

- **Nodo de Processamento:** nodo responsável por executar toda a tarefa de processamento e análise dos eventos. As informações são transferidas de um nodo Broker, extraídas caso tenham sido compactadas, processadas e analisadas com base nas tarefas definidas para este nodo. O nodo pode ainda enviar dados de notificação para um nodo Broker caso o mesmo tenha sido configurado para tomar ações como esta, quando identificado algum evento de interesse.

Para evitar a perda de informações durante a iteração entre os nodos, estes fazem o envio dos eventos por meio de transações, onde qualquer tipo de erro gerado durante o processamento da informação ou durante o envio, gera uma ação de *rollback* que desfaz qualquer transação malsucedida. O processo de iteração entre os nodos pode ser visto como produtor consumidor, onde o nodo de pré-processamento gera os eventos que serão consumidos pelo nodo Broker que por sua vez produz as informações a serem analisadas pelo nodo de processamento. Cada um dos três tipos de nodos pode possuir de 1 a N instâncias de execução, sendo cada uma destas, correspondente a uma das três categorias: o dos nodos de pré-processamento; os nodos Broker; e os nodos de processamento. A abordagem com vários nodos de execução descentralizados ajuda a aumentar a garantia da disponibilidade do serviço, já que para o serviço ficar indisponível, será necessário que todos os N nodos fiquem inativos e não apenas um como em abordagens de processamento não distribuídos. A Figura 14 ilustra de forma simplificada a iteração entre os nodos da arquitetura produtor consumidor.

5.3 Tecnologias Associadas

Nesta seção serão apresentadas as principais tecnologias empregadas por este trabalho, bem como suas principais características e funcionalidades.

5.3.1 Protocolo MQTT

Em uma rede de computadores “tradicional” existem diversos protocolos de comunicação sendo executados, os quais são responsáveis por gerenciar a transferência de dados entre os dispositivos. Quando a comunicação entre dois ou mais dispositivos é abordada, assim como acontece em uma rede IoT, surge a necessidade de se escolher um protocolo capaz de gerenciar está comunicação. Capaz de gerenciar a troca de mensagens e dados entre as coisas que estiverem conectadas a rede IoT de forma eficiente, levando em consideração as características e limitações impostas pelo ambiente. Um dos protocolos que se enquadram dentro destes requisitos é o *Message Queuing Telemetry Transport - (MQTT)* (MARTINS; ZEM, 2016).

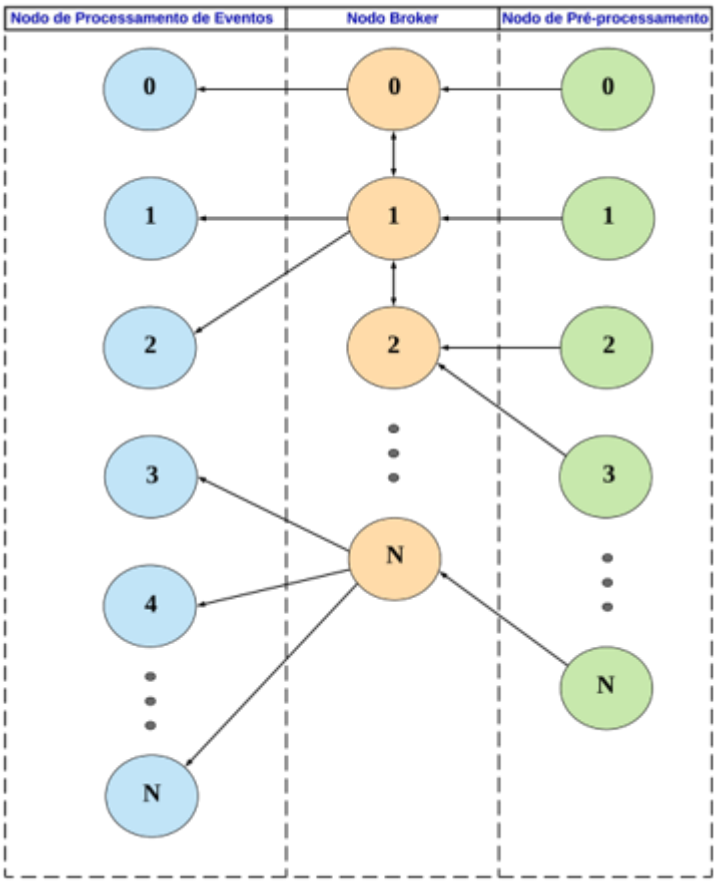

Figura 14 – Arquitetura simplificada Produtor Consumidor.

O MQTT foi desenvolvido por volta do ano de 1999, baseando-se na arquitetura *publish/subscribe* e voltado para redes inseguras com baixa largura de banda e alta latência (SONI; MAKWANA, 2017). O protocolo emprega confiabilidade na entrega das mensagens minimizando o uso da largura de banda e apresentando baixo custo de processamento, características que apresentam aplicabilidade em redes IoT.

O protocolo MQTT segue um modelo cliente-servidor onde cada um dos dispositivos da rede são os clientes, os quais se conectam-se a um servidor chamado de Broker usando o protocolo TCP. Após o cliente se conectar, as mensagens transmitidas são publicadas em tópicos. Os clientes podem se subscrever em tópicos, os quais são capazes de receber todas as mensagens que qualquer outro cliente publique nestes tópicos em especifico (MARTINS; ZEM, 2016).

bit	7	6	5	4	3	2	1	0
Byte 1	Tipo da Mensagem				Flag DUP	Nível QoS		RETAIN
Byte 2	Largura restante							

Figura 15 – Cabeçalho do protocolo MQTT.
Fonte: (MARTINS; ZEM, 2015).

A Figura 15 mostra o cabeçalho presente em cada uma das trocas de mensagens do protocolo MQTT, este possui 2 bytes de tamanho, sendo o primeiro byte usado para identificar o tipo da mensagem e os campos marcadores:

- ***Duplicate Delivery* (DUP):** é ativado quando o cliente ou o servidor tenta reenviar uma mensagem.
- ***Quality of Service* (QoS):** indica o nível de garantia da entrega de uma mensagem.
- ***Retain***: quando uma mensagem é enviada ao servidor com este marcador ela deve ser removida do servidor mesmo depois de ser entregue aos assinantes.

Por último, o segundo byte mostrado na Figura 15 é usado para representar a quantidade de bytes remanescentes na mensagem, ou seja, a quantidade de espaço livre em bytes ainda excedente no pacote.

5.3.2 Apache Kafka

O Apache Kafka é considerado uma plataforma de *streaming* distribuído de dados, sendo usada tanto para o consumo de mensagens *offline* quanto para o *online*. Dentre as características disponibilizadas pela plataforma, pode-se citar:

- **Plataforma Distribuída:** o Apache Kafka executa o particionamento dos dados, distribuindo-os sobre diversos servidores, executando desta forma a distribuição do consumo de recursos ao longo de um *cluster* de máquinas, mas continuando a garantir a entrega ordenada das mensagens, não ocasionando na perda da semântica da ordem dos dados. Sua arquitetura distribuída também permite o aumento do número de máquinas no *cluster* ou a redução, conforme a necessidade do usuário (GARG, 2013).
- **Tolerância a Falhas:** a plataforma é capas de fornecer garantia contra falhas onde o mesmo pode manter várias cópias dos dados no *cluster* sem ocasionar grandes perdas no desempenho. Estas cópias ajudam a manter a disponibilidade da informação, onde caso uma ou mais máquinas do *cluster* caiam, a informação irá continuar sendo disponibilizada pelas demais máquinas do *cluster*. O Kafka também prove garantia de entrega e recebimento das mensagens, evitando qualquer perda de informação causado por instabilidade de rede ou nos servidores (GARG, 2013).
- **Alta Capacidade:** o Kafka foi projetado para fornecer suporte ao envio e recebimento de milhões de mensagens por segundo garantindo baixas latências (GARG, 2013).

O processo de envio e recebimento das mensagens no Apache Kafka é normalmente implementado com uma arquitetura publicação e assinatura, onde produtores fazem o envio dos dados a tópicos específicos onde estes dados, dentro destes tópicos, são separados em partições. Estratégia essa que é adotada pelo Kafka com o objetivo de manter a ordenação dos dados e ainda oferecer suporte a vários consumidores e produtores manipulando de forma concorrente o mesmo tópico. Cada um dos consumidores no Kafka devem pertencer a um grupo, consumidores que pertençam a um mesmo grupo não podem consumir dados de uma mesma partição, já consumidores de grupos diferentes podem consumir dados de uma mesma partição, porém os mesmos dados desta partição será enviado a todos os consumidores de grupos distintos (GARG, 2013). A Figura 16 ilustra o processo de envio e recebimento de mensagens no Apache Kafka.

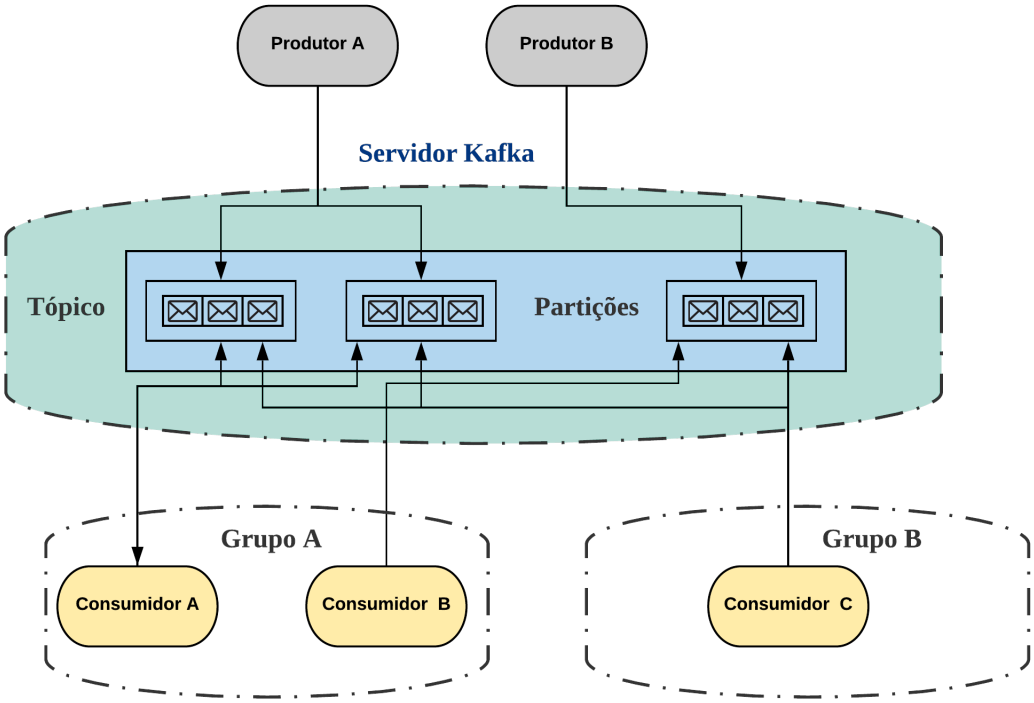

Figura 16 – Fluxo de comunicação Apache Kafka.

5.3.3 Apache Spark

O Apache Spark é um *framework* desenvolvido para executar o processamento de dados em larga escala. Este fornece uma API de alto nível para diversas linguagens de programação, como o Java por exemplo. Outras funcionalidades suportadas pelo Spark incluem o uso de SQL para a execução de filtros e consulta aos dados processados, esta funcionalidade é provida pela biblioteca Spark SQL (MENG et al., 2016).

O Spark se destaca por sua capacidade de manter grandes conjuntos de dados em

memória para executar o processamento destas informações em tempo de execução (SHORO; SOOMRO, 2015).

No Spark aplicações executam como conjuntos de serviços independentes, os quais são coordenados pelo SparkContext, um objeto presente no programa principal. Para que seja possível sua execução em *cluster*, o SparkContext deve se conectar a um gerenciador, o qual fica responsável por alocar os recursos aos serviços. Após efetuar a conexão, os objetos executors ficam responsáveis por realizarem os cálculos e armazenar os dados da aplicação. Por fim é enviado o código da aplicação no formato JAR ou Python aos executors para que assim o SparkContext possa designar tarefas a serem executadas pelos executors com estas aplicações submetidas (MENG et al., 2016).

5.3.4 Esper

O Esper é uma ferramenta de código aberto que oferece recursos para a execução de processamento de eventos complexos e análise de fluxos de dados em tempo real ou próximo do real. Seu principal objetivo é atender aos requisitos de aplicações que necessitem analisar e reagir a algum tipo de evento, como por exemplo, softwares de monitoramento de rede e de detecção de intrusão (SUHOTHAYAN et al., 2011).

Para a especificação dos eventos complexos o Esper faz uso de uma linguagem declarativa semelhante ao SQL (*Structured Query Language*) denominada EPL, esta linguagem inclui todos os operadores suportados pelo SQL, acrescentando ainda funções adicionais para a definição, interação de janelas e geração de saídas. A EPL e a API de processamento do Esper estão disponíveis como bibliotecas para as linguagens Java e .NET.

A linguagem EPL disponibilizada pelo Esper fornece duas sintaxes distintas para que se execute os filtros dos dados: a primeira se caracteriza pelo uso de restrições aninhadas, incluindo conjunções, disjunções, negações, sequências e iterações para realizar o filtro dos dados; já a segunda usa expressões regulares. Ambas as sintaxes oferecem a mesma capacidade de expressividade. O Esper também possibilita o uso da programação das políticas para executar a seleção de eventos explicitamente, explorando os modificadores every e every-distinct (CUGOLA; MARGARA, 2012).

5.4 Concepção da Arquitetura

Nesta seção será apresentado a nova arquitetura de processamento de eventos complexos distribuída para a IoT proposta por este trabalho, introduzindo as ferramentas acopladas a esta arquitetura, bem como suas funcionalidades e peculiaridades, pelas quais foram escolhidas para serem agregadas a este trabalho.

Para o desenvolvimento de uma arquitetura com capacidade de processar even-

tos complexos na IoT, necessitou-se primeiramente de uma forma simplificada de se abstrair a heterogeneidade dos eventos a serem analisados nestes ambientes. Para atingir este determinado objetivo optou-se por fazer uso de um *middleware* da IoT o qual se encarrega de abstrair toda e qualquer complexidade do ambiente.

O EXEHDA-SA consiste de um modelo de arquitetura distribuída de alto nível, o qual foi projetado com base nas formalizações definidas pelo *middleware* EXEHDA. Estas formalizações proporcionam ao EXEHDA-SA funcionalidades e características oportunas para o desenvolvimento da proposta deste trabalho. Porém apesar do EXEHDA-SA ser direcionado a ambientes distribuídos, as implementações disponíveis do mesmo apresentam a compreensão dos eventos em escopo local, gerando um gargalo de processamento em ambientes IoT altamente distribuídos. Desta forma para que o EXEHDA-SA seja moldado aos requisitos e objetivos propostos por este trabalho, executou-se as seguintes modificações:

- **Módulo de Compreensão:** neste módulo aplicou-se alterações de modo a permitir a execução da correlação dos eventos de forma distribuída, removendo o gargalo de processamento ao descentralizar esta tarefa, proporcionando também uma maior tolerância a falhas, já que com a descentralização da tarefa não á mais um único ponto de falha.

- **Módulo de Pré-processamento:** com o objetivo de implementar uma estratégia de distribuição dos dados com a preocupação do consumo de rede, o Módulo de pré-processamento foi alterado para que permitisse a inserção de chaves identificadoras, visando permitir a divisão lógica do fluxo de dados. Também se adicionou a funcionalidade de compactação dos dados a serem enviados, permitindo assim a redução do consumo de rede. Este método de comunicação e melhor detalhado na seção 5.1.

Na Figura 17 é destacado os módulos na arquitetura EXEHDA-SA que sofreram modificações. Com as alterações executadas, desenvolveu-se uma arquitetura distribuída com três tipos de nodos: pré-processamento, broker e processamento de eventos. Onde estes nodos se relacionam em um padrão de produtor consumidor. Cada um destes serão detalhados nas subseções seguintes.

5.4.1 Nodo de Pré-processamento

Este nodo é responsável por modelar os dados recebidos para um formato de mais fácil manipulação aos outros nodos de execução, também é responsável por inserir chaves identificadoras nos eventos para que assim se possa subdividir logicamente o fluxo de informação. Este nodo realiza estas tarefas da seguinte maneira: primeiramente os eventos recebidos são inseridos em uma fila de consumo prioritária, onde a precedência desta fila é baseada no campo de nível do risco, campo este adicionado

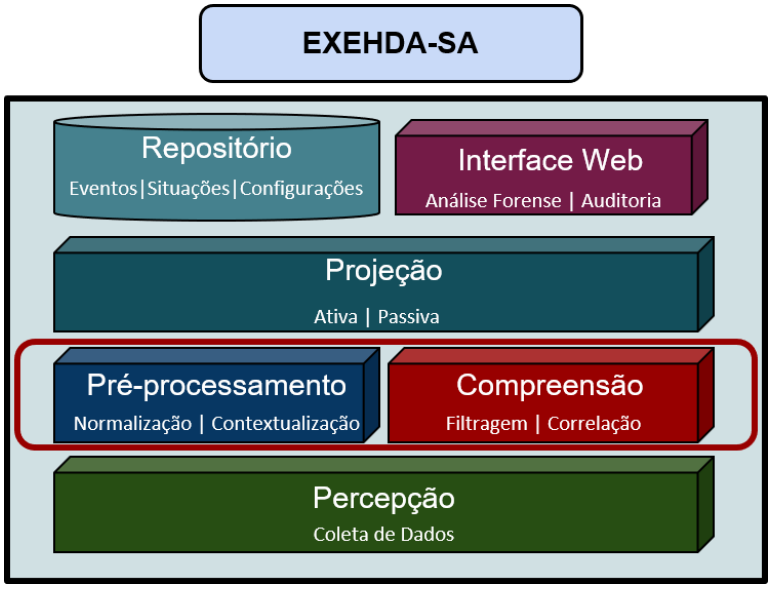

Figura 17 – Módulos a serem modificados no EXEHDA-SA.

previamente aos eventos pelo EXEHDA-SA. Esta fila tem por objetivo dar prioridade no processamento dos eventos que tenham um maior nível de risco de segurança. Após a inserção de algum evento a fila, o módulo de pré-processamento consome este evento modelando-o em um objeto Java, onde esta manipulação pode ser executada por dois sub-módulos distintos: o primeiro faz o uso de expressões lambdas no Java 8 para executar esta tarefa; já o segundo módulo faz uso do Apache Spark para executar a manipulação dos eventos para objetos Java. A escolha do sub-módulo de modelagem é definida pelo usuário através do arquivo de configuração.

O nodo de pré-processamento também é responsável por executar a inserção do campo identificador de fluxo, o qual é usado para direcionar fluxos de eventos aos nodos de processamento. A escolha da chave identificadora é baseada no campo TAG, este adicionado pelo EXEHDA-SA, sendo o mesmo gerado por meio da análise da origem dos eventos (quais tipos de dispositivos geraram este dado). A forma da escolha da chave identificadora também pode ser alterada pelo usuário com a introdução de expressões regulares no arquivo de configuração do nodo de pré-processamento, onde estas expressões podem selecionar qualquer parte dos eventos para serem usadas como chave.

Por fim o módulo cliente Kafka do nodo de pré-processamento executa o envio dos eventos a um nodo Broker, podendo executar a compactação destes antes de efetuar o envio destas informações, caso o usuário opte por executar a compactação, este deverá selecionar no arquivo de configuração qual o algoritmo de compactação deverá ser usado dentre o Snappy, LZ4 e GZIP. A Figura 18 ilustra a modelagem do nodo de pré-processamento.

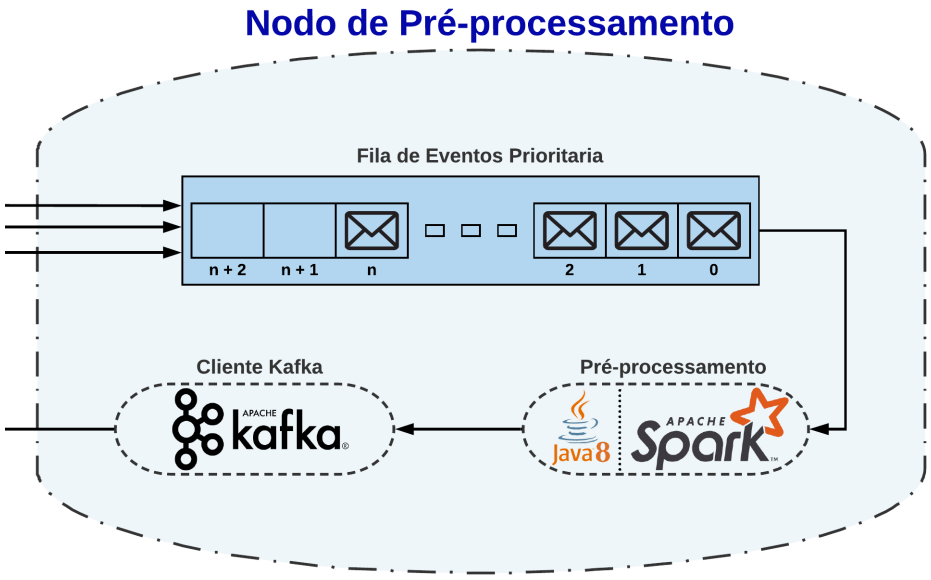

Figura 18 – Nodo de Pré-processamento.

5.4.2 Nodo Broker

Nodo responsável por armazenar os eventos recebidos pelo pré-processamento e distribui-los sob demanda aos nodos de processamento. Para a implementação deste nodo usou-se o Apache Kafka, onde em um nodo Broker encontram-se três tópicos Kafka, nos quais são armazenadas as seguintes informações:

- **Tópico de Trabalho:** tópico que armazena todos os eventos enviados pelo nodo de pré-processamento. Este tópico é subdividido em N partições, onde os dos eventos são designados a uma destas partições pelo Kafka, o qual faz uso das chaves identificadoras para executar esta tarefa, garantindo que eventos detentores de uma mesma chave identificadora serão designados a uma mesma partição. Estas partições auxiliam o Kafka no gerenciamento do acesso concorrente dos eventos, limitando a manipulação de uma mesma partição por no máximo um nodo, garantindo assim uma entrega ordenada e continua do fluxo armazenado nesta partição ao nodo de processamento que esteja consumindo a mesma. Esta estratégia evita a possível perda na identificação de eventos complexos, já que caso os eventos de um mesmo fluxo fossem divididos, eventos A e B que juntos caracterizariam um evento complexo, poderiam ser designados a nodos de processamento distintos, o que impossibilitaria a identificação deste evento complexo.

- **Tópico de Notificações:** neste tópico são armazenadas todas as notificações de identificação de eventos complexos, estas geradas pelos nodos de processamento. Estes dados de notificações armazenados no tópico podem por exemplo serem usados por administradores, com o intuito de adotar alguma ação base-

ado nos dados da notificação retornada ou ainda estas informações podem ser integradas para uso de ferramentas de terceiros.

- **Tópico de Regras:** responsável por armazenar conjuntos de regras CEP a serem enviadas aos nodos de processamento. Cada conjunto de regra possui uma chave identificadora idêntica a chave do fluxo para o qual esta deverá ser aplicada, permitindo assim que os nodos de processamento usem apenas as regras especificas para seus respectivos fluxos de eventos.

Todo o processo de comunicação executado entre o nodo Broker e os demais nodos de processamento é executado por meio de transações, de modo que quando ocorre qualquer erro ou falha, uma operação de *rollback* é executada, desfazendo qualquer modificação executada por uma operação malsucedida, garantindo que dados não sejam perdidos. Estas operações de comunicação executam por meio do protocolo MQTT, o que permite ao usuário, caso este necessite, o uso de forma facilitada de criptografia para o envio e recebimento das mensagens, algo imprescindível para ambientes vulneráveis que necessitem trafegar dados sensíveis de forma segura.

Assim como os demais nodos desta arquitetura, o nodo Broker pode possuir de 1 a N instâncias de execução, distribuindo os fluxos de dados entre estas instâncias. Como os fluxos de dados estão armazenados e divididos entre as instâncias, o usuário pode preferir ter a garantia contra a indisponibilidade destas informações caso um dos nodos Broker falhe, para isto o nodo pode manter uma cópia dos dados armazenada nas outras instâncias, sendo definido na configuração deste nodo o parâmetro de fator de replicação, onde este número vai de zero ao número de instâncias do nodo Broker em execução menos um. Onde o valor zero não dá garantias contra falhas, já o valor um garante que caso um nodo Broker caia, não haverá perda das informações. Deste modo, quanto maior for o fator de replicação adicionado, mais nodos poderão falhar sem que ocorra a indisponibilidade dos dados, porém este fator de replicação gera uma demanda maior de recursos computacionais aos nodos, recursos estes necessários para que se possa manter o *backup* dos dados atualizado entre todas as instâncias. A Figura 19 ilustra o nodo Broker com as respectivas tecnologias empregadas.

5.4.3 Nodo de Processamento

Neste nodo é executado toda a e qualquer tarefa de processamento e análise de eventos. Primeiramente para executar esta função, quando um nodo de processamento é instanciado, o módulo cliente Kafka deste comunica sua disponibilidade para processar eventos ao Nodo Broker, o qual irá designar uma parte dos fluxos de eventos para este processar. A designação dos fluxos é executada com base na quantidade de nodos de processamento disponíveis, quanto mais nodos processando, menos flu-

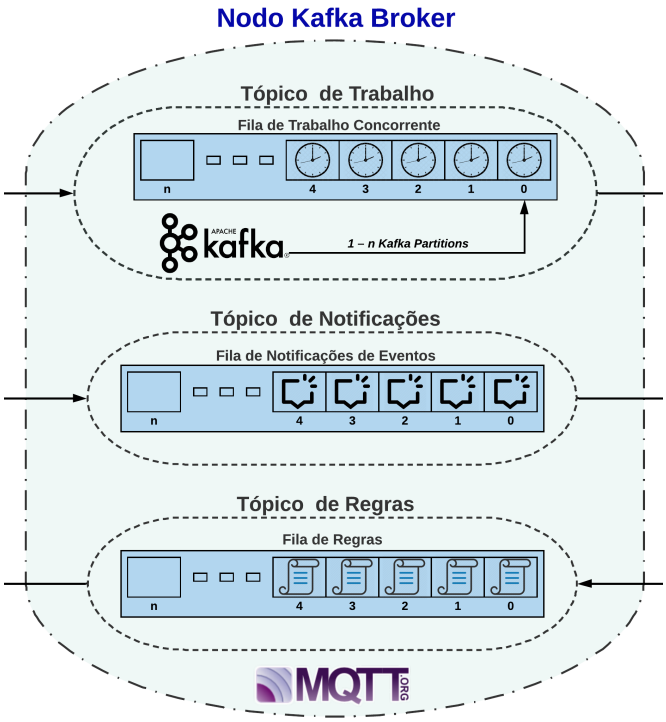

Figura 19 – Nodo Broker.

xos de eventos serão designados a um nodo, com o limite de um único fluxo por nodo. Após receber uma parte dos fluxos, o módulo cliente Kafka requisita ao nodo Broker as regras de processamento correspondente aos fluxos designados para este. A correspondência dos fluxos com as regras é feita pela chave identificadora, onde a chave da regra corresponde a chave do fluxo para o qual esta deve ser aplicada. O módulo cliente Kafka verifica periodicamente por novas regras de processamento, permitindo que o usuário adicione novas regras em tempo de execução.

Após o recebimento dos eventos a serem processados, é executada a descompactação dos mesmos, caso o usuário tenha optado por executar a compactação destes no nodo de pré-processamento. Seguindo a descompactação, os eventos e as suas correspondentes regras são submetidos ao Esper o qual fica responsável por aplicar as regras de processamento sobre os fluxos de eventos, visando assim identificar a ocorrência de eventos complexos.

Caso haja a identificação de algum evento complexo, o comportamento padrão definido pela arquitetura é o envio de uma notificação ao nodo Broker com os dados que geraram esta identificação do evento complexo e a regra de processamento que fez a detecção deste evento. Este comportamento descrito pode ser alterado pelo usuário por meio da passagem de um objeto Java como parâmetro da classe principal do nodo de processamento, assim este objeto irá representar o novo padrão comportamental a ser executado. A Figura 20 ilustra a modelagem do nodo de processamento de dados.

Todos os nodos de processamento são organizados em um mesmo grupo Kafka,

isto possibilita a distribuição dos eventos de forma concorrente, já que nodos pertencentes a um mesmo grupo de processamento dividem o conjunto de informação armazenado em um tópico, trabalhando em parcelas distintas do mesmo.

O nodo de processamento, assim como os demais, implementa mecanismos de tolerância a falhas, onde caso algum dos nodos de processamento, por um motivo qualquer ficar ocioso, deixando de se comunicar, o nodo Broker percebe imediatamente a sua inatividade e distribui os fluxos de eventos antes detidos por este nodo ocioso aos demais nodos de processamento disponíveis. Isto evita que estas informações permaneçam sem serem processadas até que o nodo fique ativo novamente, o que poderia ocasionar na perca da identificação de eventos complexos, ou ainda na identificação tardia dos mesmos, tendo em vista que a detecção de certos eventos complexos só faz sentido se for efetuada em um intervalo de tempo muito curto.

Um exemplo deste cenário citado, seria a detecção de uma tentativa de invasão em andamento aos dados sensíveis de um banco de dados, caso o administrador seja notificado deste possível ataque em andamento, antes que o mesmo seja bem sucedido por parte dos invasores, este administrador poderá tomar alguma decisão de modo a mitigar esta tentativa de invasão.

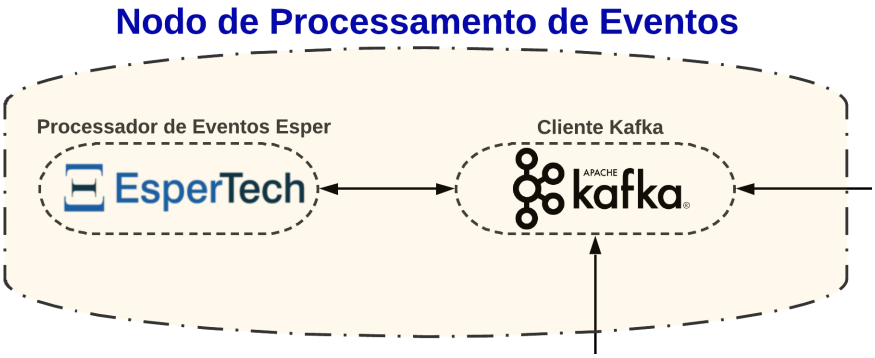

Figura 20 – Nodo de processamento.

5.4.4 Visão geral da arquitetura

Na Figura 21 é apresentada a visão geral da arquitetura proposta bem como os fluxos de comunicação entre cada um dos três nodos.

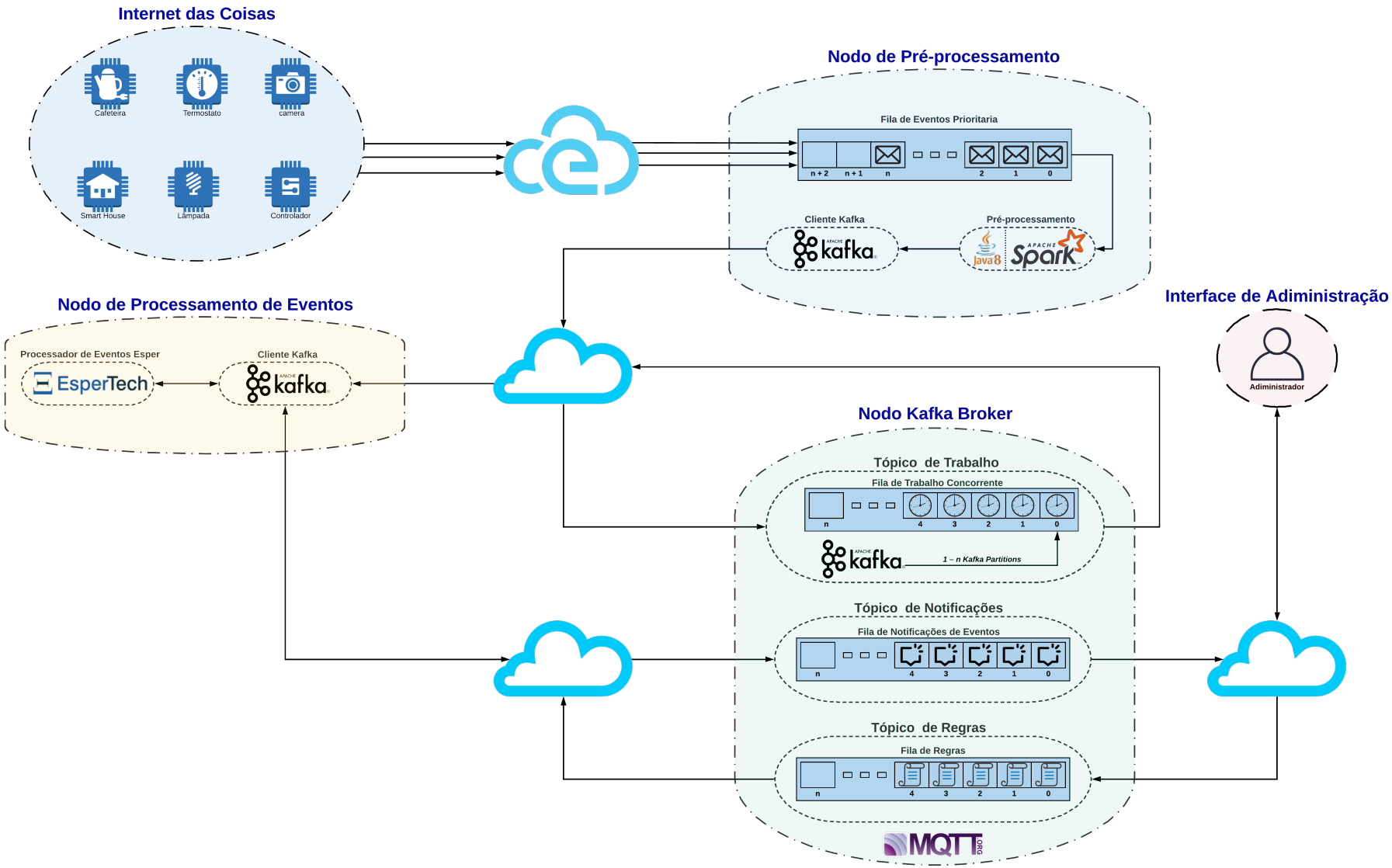

Figura 21 – Visão geral da arquitetura.

6 EXEHDA-DEP: AVALIAÇÕES E RESULTADOS

Neste capitulo são apresentados os testes de escalabilidade e consumo de rede executados com a presente arquitetura proposta, bem como os resultados obtidos com a implementação destas avaliações. Neste capitulo também será descrito o cenário de caso de uso aplicado para a execução dos testes, incluindo as motivações e justificativas para o uso do mesmo.

6.1 Cenário de Aplicação

O cenário de aplicação proposto para a execução dos testes teve como referência o ambiente computacional da Universidade Federal de Pelotas (UFPel) o qual possui uma infraestrutura com características da Computação Ubíqua. Dentre estas características presentes, pode-se citar:

- o grande número de dispositivos heterogêneos com hardwares e recursos distintos conectados a rede, como computadores com sistemas operacionais diversos, impressoras, servidores, dispositivos móveis, câmeras de segurança, entre outros;
- um ambiente descentralizado, com dispositivos localizados em campus geograficamente separados, os quais enviam e recebem informações entre si continuamente;
- a necessidade de trafegar um considerável volume de eventos por *links* de conexão com baixa largura de banda;
- a falta de padronização das informações geradas por serviços oferecidos pelos *datacenters* onde eventos de sistemas legados geram *logs* em diversos formatos e padrões.

Com o enfoque de demonstrar a capacidade da arquitetura processar um considerável volume de eventos em um ambiente ubíquo com escalabilidade e mantendo o consumo de rede estável, o ambiente computacional da UFPEL se adéqua a estes

requisitos, o qual almeja por uma arquitetura apta a lidar com o processamento distribuído e heterogêneo dos eventos e com capacidade de executar em ambientes com limitações de largura de link.

Dentro deste cenário da UFPEL, optou-se por executar a análise de eventos da segurança da informação, tendo em vista a pertinência da identificação de eventos deste tipo, já que há uma grande dificuldade e necessidade de se identificar e manter redes de grande porte seguras contra ataques de Hackers (JOSHI; SINGH, 2017).

O processamento de eventos de segurança favorece também a execução do estudo da capacidade do EXEHDA-DEP em lidar com um considerável volume de eventos quase que em tempo real, já que neste caso há necessidade de se analisar boa parte do trafego de rede gerado por milhares de dispositivos na UFPel, em busca de eventos de segurança da informação. Estes eventos caracterizam uma possível quebra de segurança, de modo que caso sejam detectados, deve-se notificar o administrador de rede de imediato, para que assim este possa tomar as ações cabíveis. Assim, as regras EPL aplicadas ao ambiente computacional da UFPEL visam a identificação da execução de ataques de Hackers como de força bruta, injeção de código, dentre outros. Estas regras EPL foram desenvolvidas e testadas por trabalhos anteriores do grupo de pesquisa (ALMEIDA et al., 2019).

Outra característica presente na análise de eventos de segurança que se adéqua aos objetivos deste trabalho é a ausência de padronização nas estruturas dos dados a serem analisados, visto que há diferentes dispositivos com protocolos e recursos distintos se comunicando na rede, o que gera a necessidade do EXEHDA-DEP em trabalhar com dados heterogêneos, assim como em um ambiente da IoT (HANES et al., 2017).

6.1.1 Ambiente de Teste

Devido a restrições de segurança para o acesso direto de terceiros ao ambiente computacional da UFPEL, optou-se por executar a virtualização deste ambiente. Para isto fez-se uso de um *cluster* para executar a simulação dos diferentes dispositivos presentes no ambiente computacional da UFPEL, sobre os quais os nodos da arquitetura proposta poderão executar.

As configurações do *cluster* usado para a simulação consiste:

- **Processamento:** 8 processadores Intel Xeon E5-4650V3 de 2.1GHZ fornecendo 96 núcleos de processamento.
- **RAM:** 64 módulos de 8 *gigabytes* DDR4, totalizando 512GB de memória.
- **Armazenamento:** 2 discos rígidos de 2 terabytes com velocidade de 12Gb/s executando em RAID 1.

Já a virtualização das máquinas no *cluster* foi executado por meio do Docker[1], o qual fornece uma camada de abstração e automação para a virtualização de sistemas, o que permitiu a execução do isolamento dos recursos de hardware do *cluster*(memória RAM, CPU, HD e largura de banda) permitindo uma melhor aproximação da simulação ao ambiente real da UFPEL.

As restrições dos recursos computacionais foram executadas da seguinte maneira:

- **CPU -** o isolamento dos recursos de processamento foi elaborado com a limitação do número de núcleos de processamento disponíveis para cada uma das máquinas simuladas, bem como quais destes núcleos disponibilizados pelo *cluster* seriam usados por cada uma destas máquinas virtuais. A quantidade de núcleos de processamento disponibilizado as máquinas simuladas depende do teste em questão a ser executado, podendo variar conforme o necessário.

- **HD -** a restrição do armazenamento foi elaborada através da limitação da velocidade de leitura e escrita no disco rígido disponibilizada a cada uma das máquinas virtuais, visando assim simular a independência do armazenamento entre as máquinas virtuais. Não foram feitas limitação quanto a quantidade de HD(*Hard Disk*) disponibilizada por cada nodo, ficando disponível as máquinas virtuais a quantidade total livre no *cluster*.

- **Rede de Comunicação -** o Docker permite a virtualização de redes de comunicação, incluindo a simulação de roteadores efetuando a distribuição de IPs para as máquinas virtuais. Esta funcionalidade possibilitou a simulação e análise do consumo de tráfego de rede gerado pela comunicação entre nodos de processamento executando em máquinas distintas.

- **Memória RAM -** o controle da memória RAM limitou-se a disponibilizar uma quantidade X para cada uma das máquinas virtuais, onde o volume de memória disponibilizado para cada máquina pode variar dependendo das necessidades dos testes em questão a ser executado.

Para a simulação do tráfego de rede usou-se um conjunto de *logs* de tráfego de rede de diferentes dispositivos (Firewall, Roteadores, Switchs...) usados na UFPEL. Estes dados foram enviados ao nodo de pré-processamento de forma periódica e não sequencial, visando simular a geração natural dos *logs* no ambiente da UFPEL. O envio destes dados foi executado com o auxílio de uma ferramenta desenvolvida em Java, esta também projetada por este trabalho, se encontrando disponível para acesso e uso junto da arquitetura principal projetada. Estes *logs* de dispositivos empregados por este trabalho já foram usados anteriormente em outro trabalho desenvolvido pelo presente grupo de pesquisa (ALMEIDA et al., 2019).

[1] `https://www.docker.com/`

Na Figura 22 é apresentado um diagrama do fluxo da execução do EXEHDA-DEP, onde eventos de registro da atividade de rede na UFPel são capturados e padronizados pelos demais módulos previamente concebidos do *middleware* EXEHDA e modelo arquitetural EXEHDA-SA (ALMEIDA et al., 2019), onde estes eventos são posteriormente enviados ao nodo de pré-processamento o qual empacota e classifica este evento usando uma chave identificadora, o enviando em seguida por uma transação a um nodo Broker.

Por sua vez, o nodo Broker recebe o pacote contendo o evento, o despachando imediatamente para um nodo de processamento, se baseando para isto na chave identificadora adicionada pelo nodo de pré-processamento. Este nodo também armazena os registros de notificações disparados pelos nodos de processamento, bem como as regras EPL definidas pelo administrador de rede na interface administrativa, onde as quais serão distribuídas por este nodo, aos nodos de processamento, se baseando para isto nos fluxos de eventos que estes estiverem analisando.

O nodo de processamento recebe o pacote do evento por meio de uma transação aberta com um nodo Broker. Este nodo, executa a análise de eventos complexos, por meio do emprego de regras EPL definidas pelo administrador de rede, as quais são inseridas em uma interface administrativa. Estas regras aplicadas pelos nodos de processamento, foram definidas de modo a permitirem a identificação de possíveis tentativas de invasão ou quebra de segurança a rede da UFPel. Sempre que um nodo de processamento faz a identificação de um evento complexo, o comportamento padrão executado é o envio de uma notificação ao nodo Broker, o qual armazena esta mensagem, para que o administrador possa por meio da interface administrativa, analisar os incidentes de segurança da informação detectados e assim tomar alguma ação cabível ao mesmo.

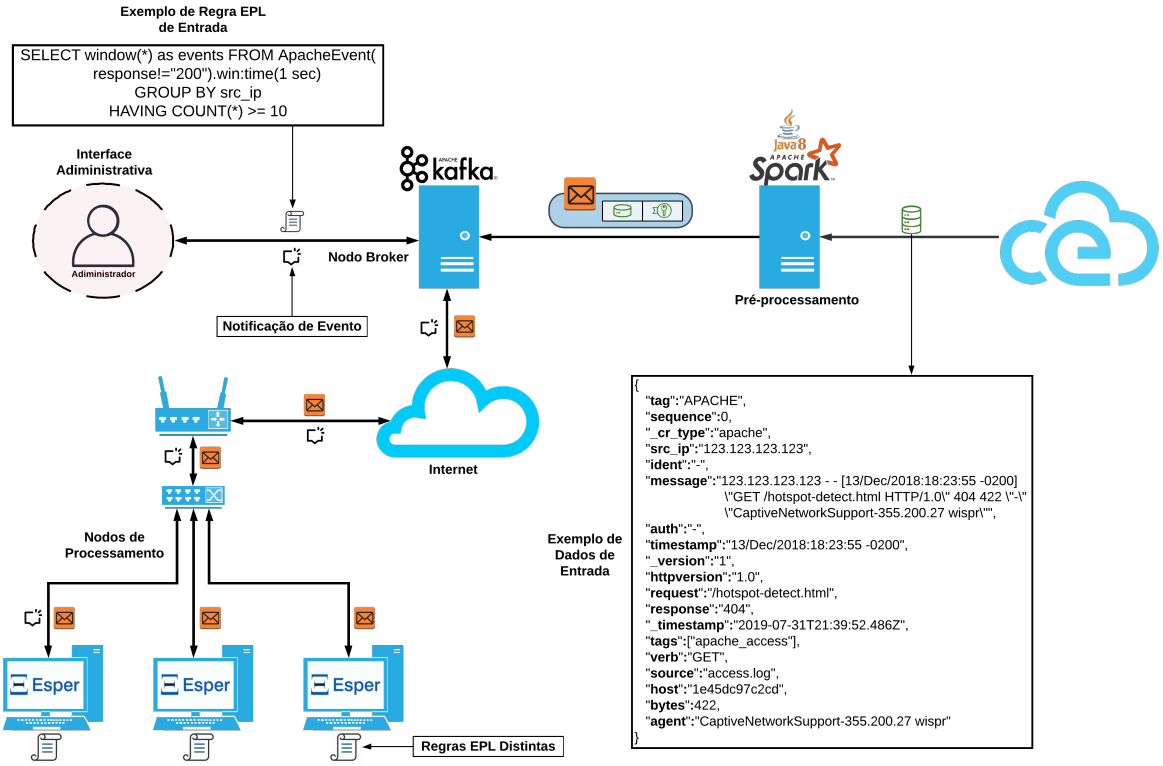

Figura 22 – Fluxo de execução do EXEHDA-DEP.

6.2 Escalabilidade

Nesta seção será apresentado os resultados dos testes de escalabilidade executados com os nodos de pré-processamento e processamento. Todos os testes de desempenho apresentados nesta seção foram executados com uma arquitetura de cinco nodos Broker, estes executando em máquinas virtuais geradas pelo Docker com as seguintes configurações: oito núcleos de processamento do *cluster*; dezesseis *gigabytes* de RAM; cinquenta *megabits* de velocidade máxima de leitura e escrita em disco rígido.

6.2.1 Escalabilidade Vertical

Primeiramente executou-se testes com o objetivo de mostrar a capacidade da arquitetura escalar verticalmente, isto é, demonstrar que com o aumento dos recursos computacionais disponíveis, há algum aumento significativo na taxa de processamento da arquitetura. Para isto simulou-se com o Docker uma máquina virtual inicialmente com os seguintes recursos computacionais: um núcleo de processamento do *cluster*; seis *gigabytes* de memória RAM; uma taxa de leitura e escrita de disco de dez *megabits* por segundo. Após a simulação inicial nesta arquitetura, repetiu-se o mesmo teste de processamento, porém com os valores de todos os recursos computacionais (número de núcleos de processamento, memória RAM e velocidade de leitura e es-

crita do HD) multiplicados por dois, três, quatro, cinco, dez e quinze. Esperando-se obter com a execução destes testes um aumento contínuo e progressivo dos valores médios de eventos processados por segundo, conforme os recursos computacionais são incrementados nos diferentes testes de processamento citados, visando assim demonstrar a capacidade da arquitetura escalar verticalmente.

Cada uma das simulações citadas fora executada com os nodos de Pré-processamento e de Processamento, onde cada um destes testes foram realizados trinta vezes. O valor médio da taxa de processamento obtido com as trinta iterações e seus respectivos desvio padrão podem ser visto nas Figuras 23 e 24.

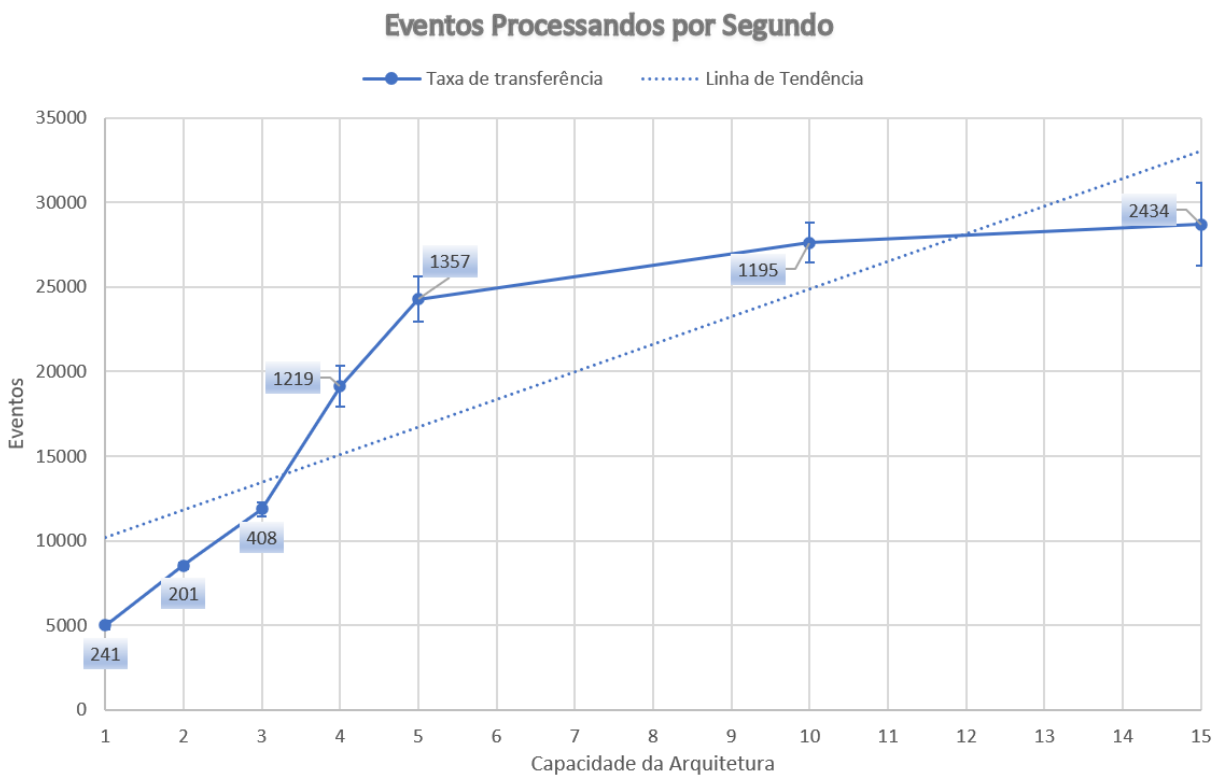

Figura 23 – Escalabilidade Vertical nodo de pré-processamento.

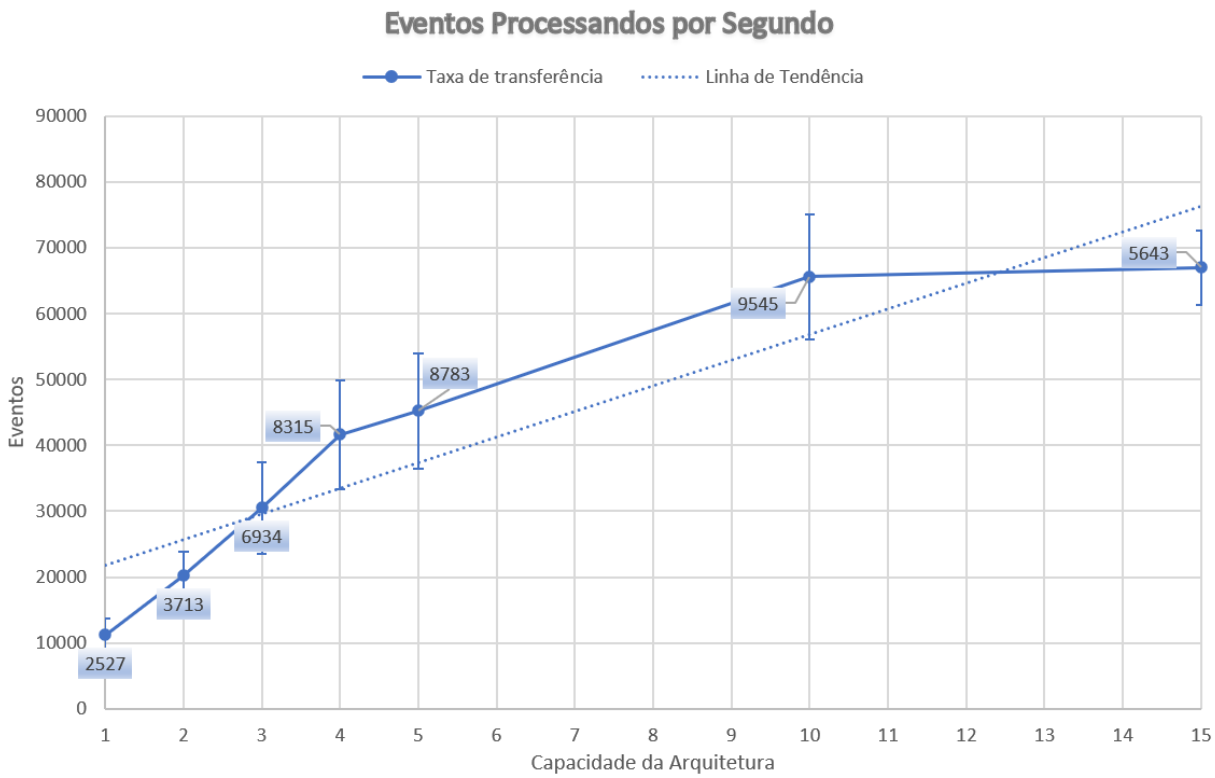

Figura 24 – Escalabilidade Vertical nodo de processamento.

Os dados obtidos com os testes, apresentados nas Figuras 23 e 24 demonstram a capacidade dos nodos de pré-processamento e processamento de escalar verticalmente, demonstrando um aumento significativo na taxa de processamento. Porém, também pode-se perceber pelo gráfico que os ganhos na capacidade de processamento diminuem ao longo do aumento dos recursos computacionais, porém mesmo com um aumento de quinze vezes da capacidade computacional inicial disponibilizada aos nodos de pré-processamento e processamento, estes continuaram apresentando algum ganho na taxa de transferência, enfatizando a capacidade da arquitetura proposta de escalar verticalmente.

6.2.2 Escalabilidade Horizontal

Para demonstra a capacidade da arquitetura proposta escalar horizontalmente, isto é, de demonstrar que quanto maior o número de nodos, maior será o ganho na taxa total de processamento disponibilizada pela arquitetura, foram executados testes visando analisar a média de eventos processados pelos nodos de pré-processamento e processamento, primeiramente em uma única máquina virtual, esta possuindo os seguintes recursos computacionais: um núcleo de processamento do *cluster*; seis *gigabytes* de memória RAM; uma taxa de leitura e escrita de disco de dez *megabits* por segundo. Após a análise do processamento de uma instância de execução dos nodos em uma única máquina virtual respectivamente, repetiu-se esta simulação au-

mentando igualmente o número de máquinas virtuais e nodos, mas sempre mantendo uma única instância dos nodos de pré-processamento ou processamento por máquina virtual. Estes testes foram executados com: dois, três, quatro, cinco, dez e quinze máquinas virtuais e nodos igualmente.

Assim, com a análise da média total de eventos processados em cada um destes testes, espera-se obter um aumento contínuo e progressivo do processamento destes eventos, conforme o número de nodos de processamento aumenta em cada um dos respectivos testes executados, demonstrando desta forma a capacidade da arquitetura escalar horizontalmente.

Cada uma das simulações citadas fora executada com os nodos de pré-processamento e de processamento, sendo cada um destes testes executados trinta vezes. O valor médio da taxa de processamento obtido com as trinta iterações e seus respectivos desvio padrão pode ser visto nas Figuras 25 e 26.

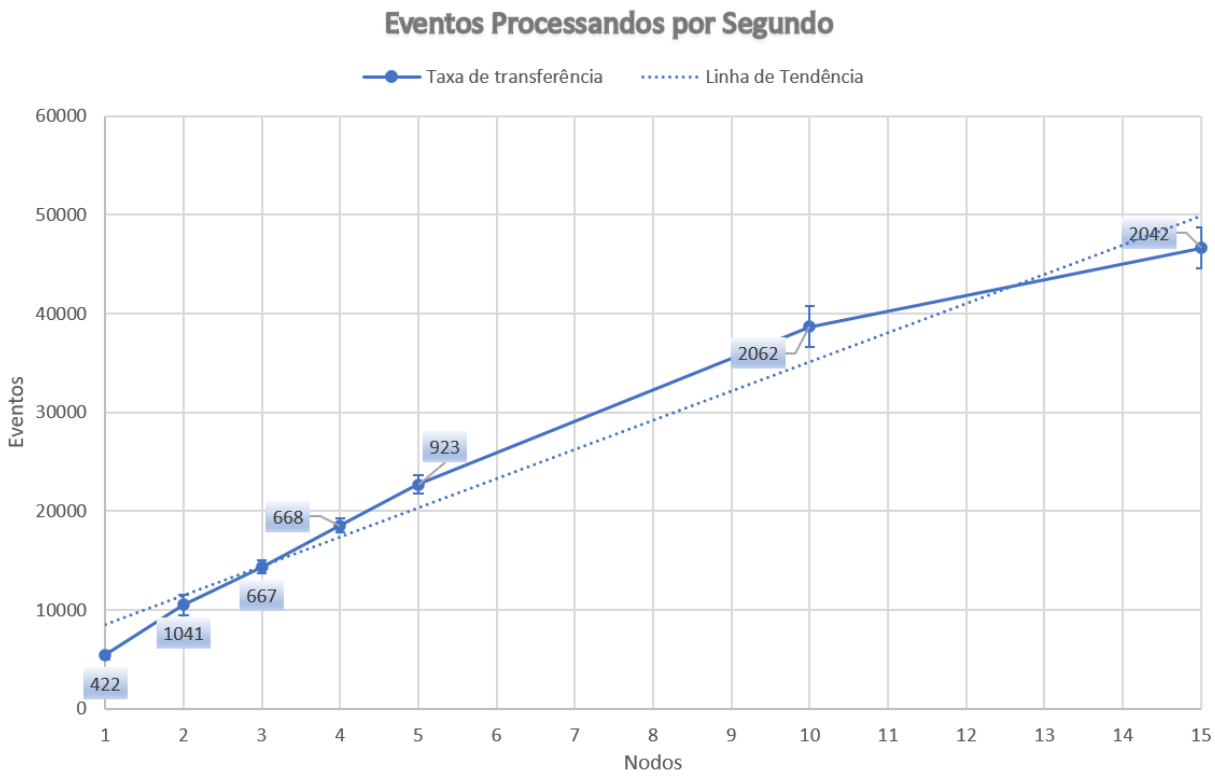

Figura 25 – Escalabilidade horizontal nodo de pré-processamento.

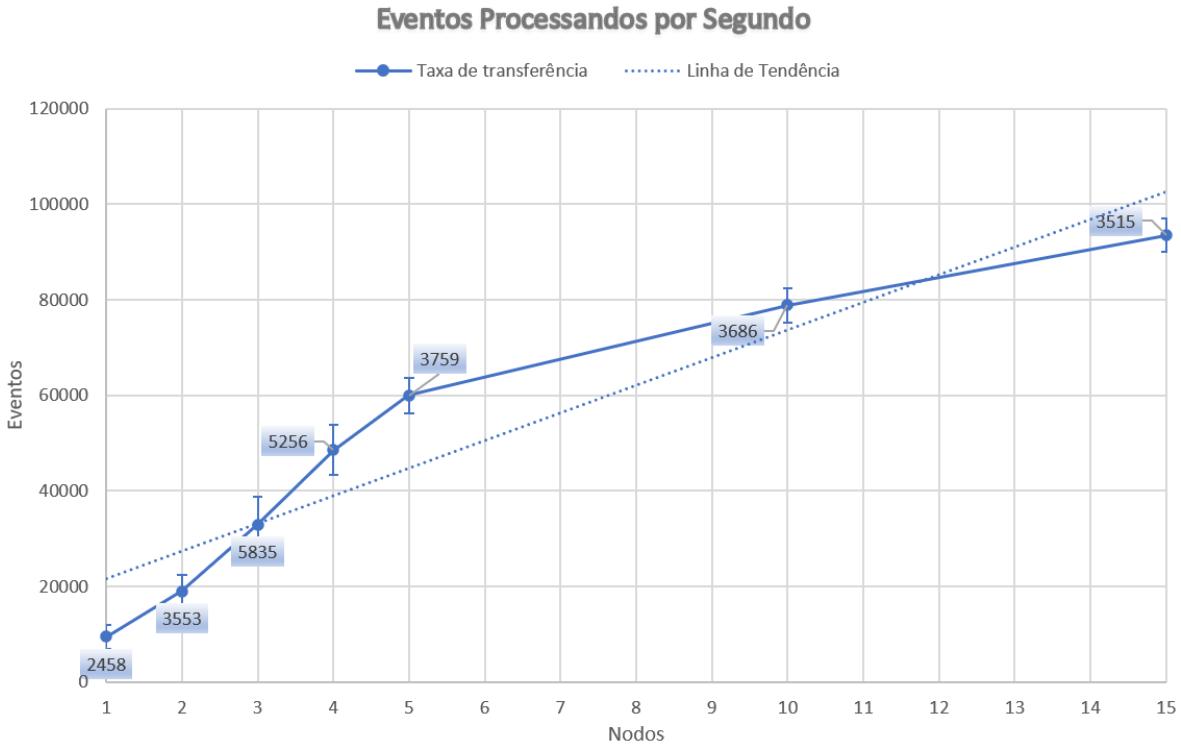

Figura 26 – Escalabilidade horizontal nodo de processamento.

Os dados obtidos com os testes, apresentados nas Figuras 25 e 26 demonstram a capacidade dos nodos de pré-processamento e processamento escalar horizontalmente, apresentando aumento significativo na taxa de processamento. Pode-se destacar o aumento contínuo e significante da taxa de processamento obtido nos testes, aumento este se mantendo significativo mesmo com quinze nodos executando ao mesmo tempo, provendo assim altas taxas de processamento, destacando a capacidade desta arquitetura de executar em ambientes altamente descentralizados e que necessitem altas taxas de processamento, características estas presentes na IoT.

6.3 Estabilidade e Consumo de Recursos

Nesta seção será demonstrada a estabilidade da proposta, estabilidade esta destacada pela capacidade dos nodos de pré-processamento e processamento de executar continuamente por longos períodos sem apresentar instabilidades ou variações bruscas na taxa de processamento. Também será mensurado o consumo dos recursos computacionais gerado pelos testes executados nesta seção, visando demonstrar a possibilidade de se executar a proposta em *hardware* de baixo custo e ainda destacar o gerenciamento eficaz dos recursos computacionais, isto é, demonstrar que a presente arquitetura não exerce um consumo elevado de memória RAM, aumentando o uso continuamente ao longo do tempo, sem nenhuma causa necessária aparente, ou mesmo que a proposta não possua períodos de ociosidade de processamento ou o

não uso da total capacidade disponibilizada pela CPU(*Central Processing Unit*).

Visando demonstrar a estabilidade da taxa de processamento dos nodos de pré-processamento e processamento executou-se a medição da taxa média de eventos consumidos ao longo do período de meia hora(1800 segundos), visando assim identificar qualquer instabilidade que afete a execução da arquitetura, gerando alguma variação brusca na taxa de processamento dos eventos. Os seguintes recursos computacionais foram disponibilizados a máquina virtual simulada para a execução deste teste: um núcleo de processamento do *cluster*; seis *gigabytes* de memória RAM; uma taxa de leitura e escrita de disco de dez *megabits* por segundo. Os dados da taxa de processamento de eventos complexos por segundo obtida com as simulações de meia hora dos nodos de pré-processamento e processamento podem ser visto nas Figuras 27 e 28.

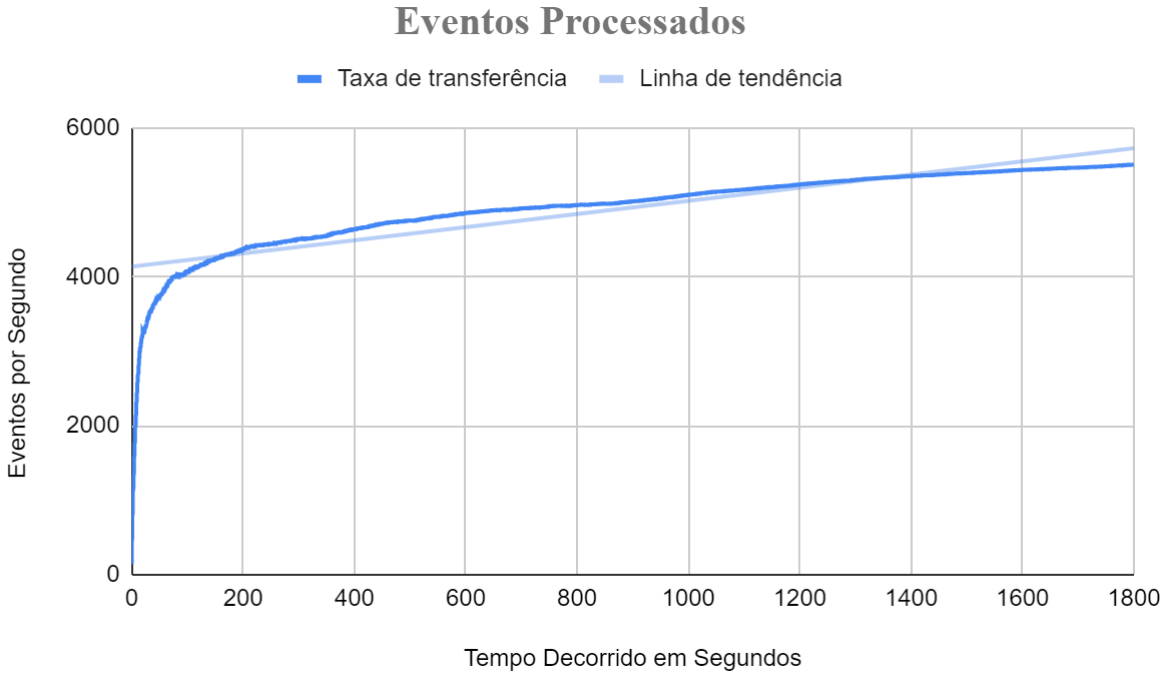

Figura 27 – Estabilidade nodo pré-processamento.

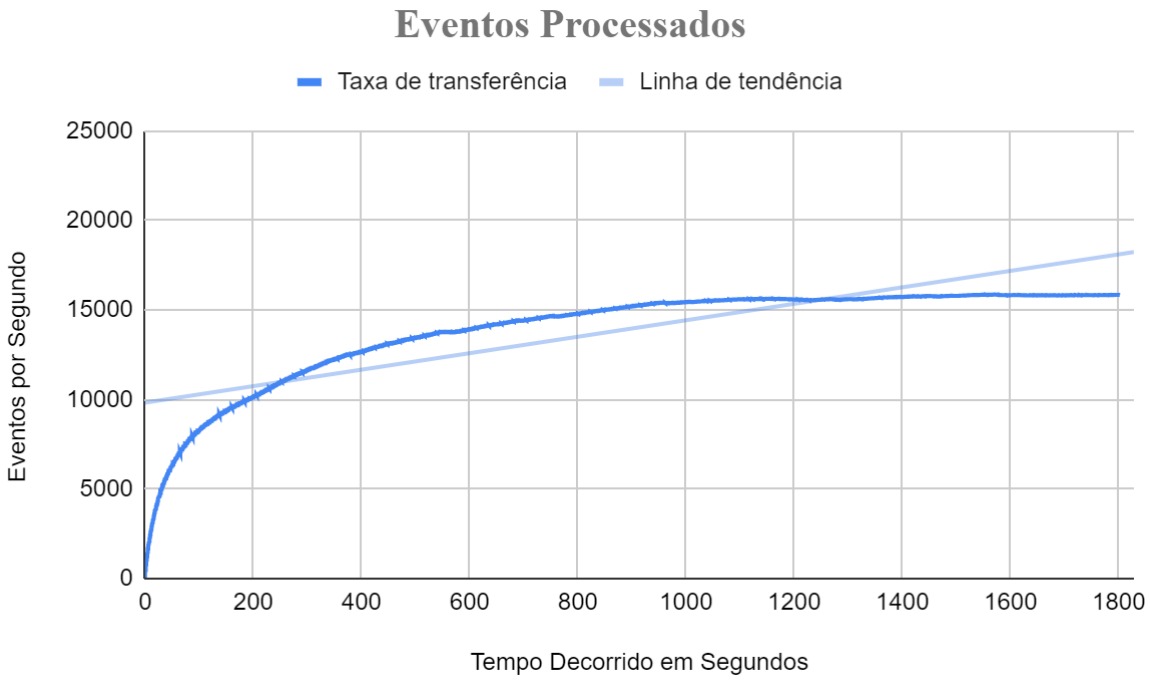

Figura 28 – Estabilidade nodo processamento.

Os dados obtidos com os testes, apresentados nas Figuras 27 e 28 demonstram a estabilidade na execução do processamento de eventos complexos pela arquitetura proposta, onde a mesma apresenta uma progressão estável e continua na taxa de processamento, não apresentando grandes picos ou variações ao longo do tempo de execução. Os testes visam demonstrar que a presente arquitetura proposta está apta a ser introduzida a meios computacionais que necessitem de soluções confiáveis, capazes de lidar com grandes fluxos de dados de forma continua e estável.

Durante os testes de estabilidades executados, executou-se também a medição do consumo de recursos computacionais gerado por cada um dos nodos, pré-processamento e processamento, durante os testes citados. A execução destas medições visa demonstrar a administração satisfatória dos recursos computacionais por parte da arquitetura proposta, isto é, demonstrar que a presente arquitetura não exerce um consumo elevado de memória RAM, aumentando o uso continuamente ao longo do tempo, sem nenhuma causa necessária aparente, ou mesmo que a proposta não possua períodos de ociosidade de processamento ou o não uso da total capacidade disponibilizada pela CPU. Os dados coletados estão ilustrados nas Figuras 29 e 30.

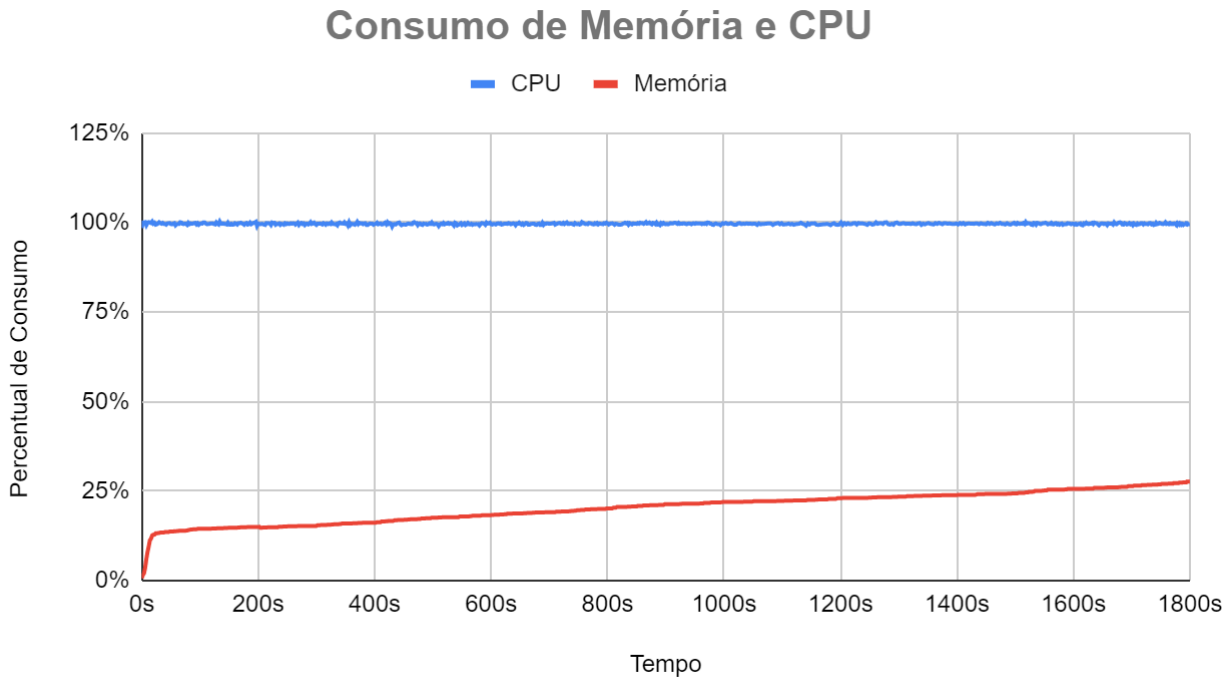

Figura 29 – Consumo de recursos pré-processamento.

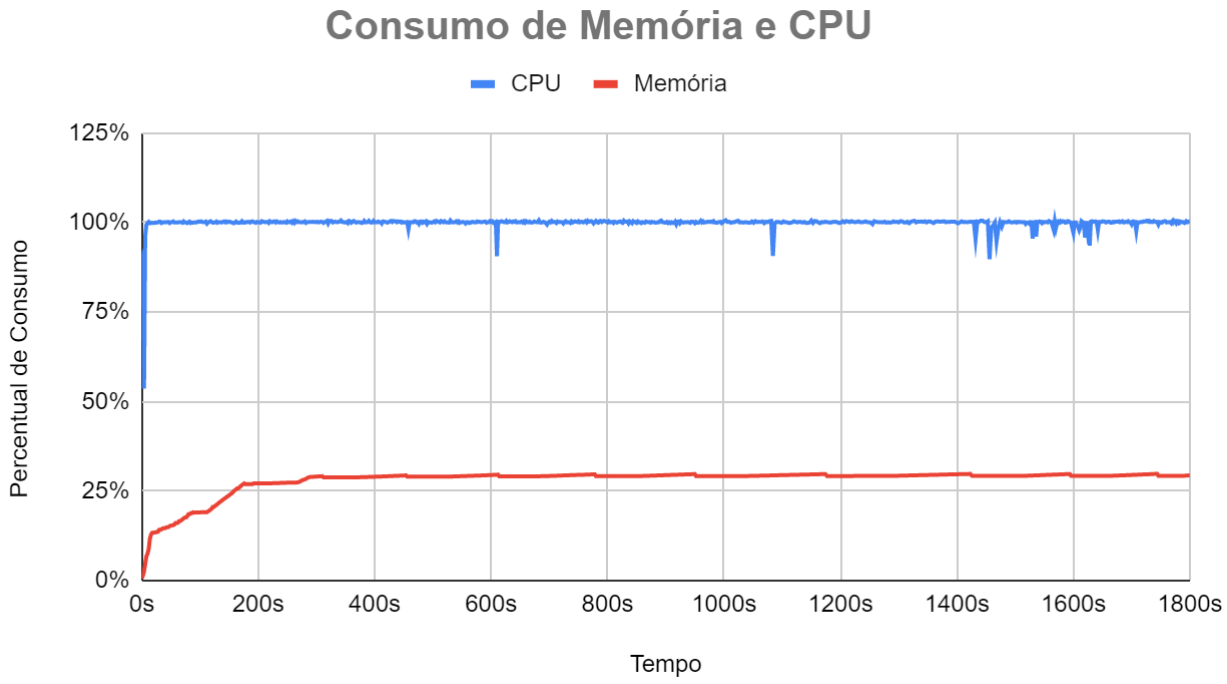

Figura 30 – Consumo de recursos processamento.

A Figura 29 demonstra o consumo estável e contínuo do tempo de processamento disponibilizado pela máquina virtual ao nodo de pré-processamento, não apresentando períodos de ociosidade ou grandes variações na média do processamento ao

longo do tempo da coleta dos dados. Já o consumo de memória RAM apresentou uma leve progressão ao longo do tempo analisado, progressão esta que pode ser explicada com a observação da Figura 27, a qual apresenta os dados da taxa média de processamento ao longo do tempo deste nodo, onde percebe-se um contínuo aumento ao longo do tempo da taxa média de processamento, o que acaba gerando consequente mente um leve e contínuo aumento do consumo de memória RAM ao longo do tempo, até que esta taxa de processamento se estabilize.

Na Figura 30 são apresentados os dados do nodo de processamento, o qual demonstra um consumo de tempo de processamento estável com leves variações, tendo em vista que a identificação de um evento complexo acarreta em um chamada subsequente de uma ação designada para este fluxo de eventos complexos, estas variações podem ser explicadas por períodos nos quais o número médio de eventos complexos identificados teve uma leve alteração. Como os dados analisados pelas regras EPL são gerados de forma aleatória, o que mais se aproxima do comportamento do ambiente real, a ocorrência durante o tempo de execução de períodos com maior e menor identificação de eventos complexos, desta forma, leves variações no consumo médio de CPU são esperadas.

Já o consumo de memória RAM exercido pelo nodo de processamento não apresenta quaisquer variações, estabilizando aos 300 segundos de simulação em pouco mais de 25% e se mantendo neste valor até o final do teste, demonstrando que a arquitetura oferece um gerenciamento adequado deste recurso.

6.4 Consumo de Rede

Nesta seção serão apresentados os testes de consumo de rede, onde estes visam demonstrar a capacidade da arquitetura proposta de gerenciar a transmissão dos dados pela rede de modo a proporcionar a distribuição dos dados com um consumo de banda estável, oferecendo opção também de usar a compactação dos eventos para assim fornecer uma redução do consumo do tráfico de rede gerado pela distribuição dos eventos, favorecendo o uso desta proposta em ambientes com baixa largura de banda, ou que estejam com os *links* de comunicação saturados pela necessidade de transmitir grandes volumes de informações.

Todos os testes demonstrados nesta seção foram executados com uma arquitetura de cinco nodos Broker, estes executando em máquinas virtuais geradas pelo Docker com as seguintes configurações: oito núcleos de processamento do *cluster*; dezesseis *gigabytes* de RAM; cinquenta *megabits* de velocidade máxima de leitura e escrita em disco rígido.

Primeiramente com o objetivo de demonstrar que a presente arquitetura proposta não gera custos adicionais ao tráfego de rede para executar a distribuição dos dados,

efetuou-se a simulação de máquinas virtuais, para que nestas fosse executado os nodos de processamento. Assim, gerou-se vinte e quatro milhões de eventos para que estes fossem distribuídos pelos nodos Broker aos nodos de processamento, visando desta forma mensurar o tráfego total de rede gerado nos *links* das máquinas virtuais que os nodos de processamento executam. Este teste foi executado com: um, dois, três, quatro, cinco, dez e quinze máquinas virtuais, onde em cada uma destas executava uma instância do nodo de processamento. Os recursos computacionais disponibilizados a cada uma das máquinas virtuais durante a execução destes testes foram: um núcleo de processamento do *cluster*; seis *gigabytes* de memória RAM; e uma taxa de leitura e escrita de disco de dez *megabits* por segundo.

Com a transmissão deste volume de eventos a uma ou mais máquinas virtuais nas quais executam os nodos de processamento, visa se possibilitar o comparativo da carga total de rede exercida sobre os *links* de comunicação de cada uma das máquinas virtuais simuladas, possibilitando-se constatar de que a presente tem a capacidade de efetuar o processamento de eventos complexos de formar distribuída sem exercer qualquer carga adicional sobre os *links* de comunicação da rede. Os resultados destes testes podem ser vistos na Figura 31.

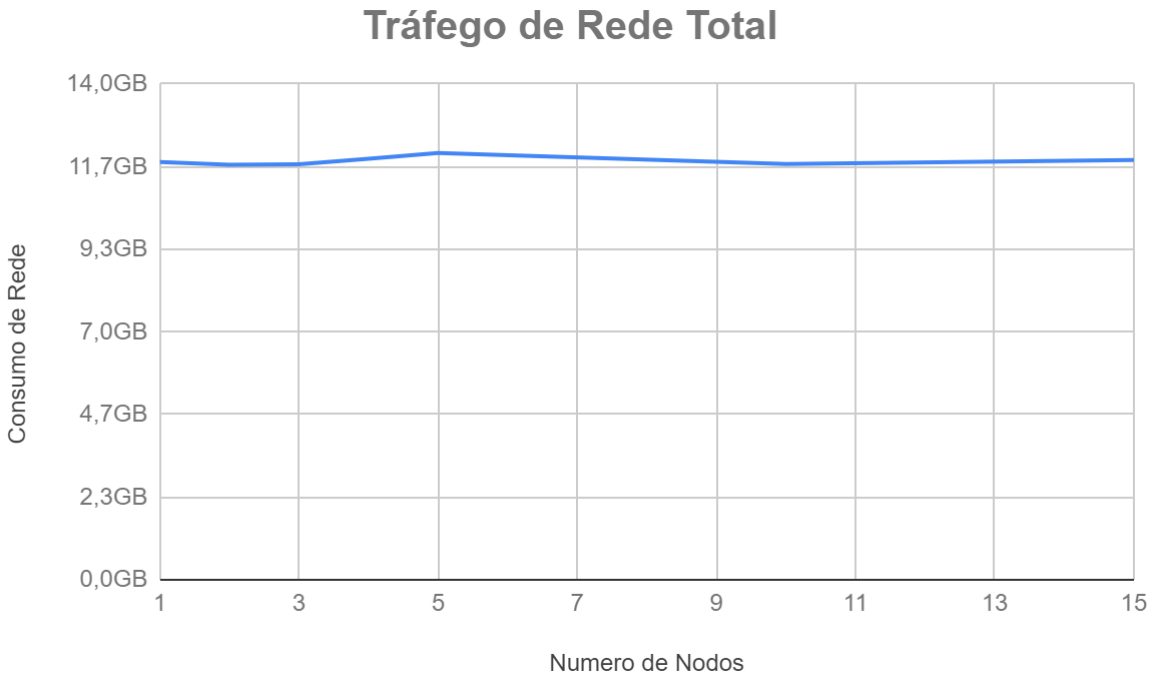

Figura 31 – Consumo de rede com a distribuição.

Os dados apresentados na Figura 31 demonstram um consumo de rede estável fornecido pela arquitetura proposta, onde o valor exigido do *link* é o mesmo, independente de quantos nodos estiverem trabalhando em conjunto. As pequenas variações vistas no gráfico podem ser explicadas por comunicações adicionais executadas pe-

los sistemas operacionais das máquinas virtuais. Como os dados foram coletados analisando o consumo de rede total gerado no *link* de comunicação disponibilizado a cada uma das máquinas, o tráfego gerado ocasionalmente pelo sistema operacional ou por suas aplicações acaba sendo considerado em conjunto, ou mesmo transmissões mal sucedidas, que por este motivo necessitaram ser executadas novamente, geram uma pequena discrepância nos dados de tráfego de rede. Porém estes valores não interferem para a constatação de que a presente proposta prove uma arquitetura com consumo de rede estável, não exercendo carga adicional para a distribuição dos dados aos nodos de processamento.

Outra funcionalidade provida pela arquitetura proposta é a capacidade de compactar os dados trafegados para que assim se possa diminuir o consumo de rede total gerado no tráfego das informações. Esta funcionalidade visa facilitar o transporte dos dados aos nodos de processamento em ambientes que tenham baixa largura de banda ou ainda que sofram com *links* de comunicação saturados pelo grande volume de dados trafegado nestas redes.

Para demonstrar esta capacidade de redução de tráfego gerada pela compactação, executou-se a produção de dezoito milhões de dados aos nodos Broker, os quais redirecionaram estas informações a um nodo de processamento, o qual executava em uma máquina virtual com as seguintes configurações: um núcleo de processamento do *cluster*; seis *gigabytes* de memória RAM; e uma taxa de leitura e escrita de disco de dez *megabits* por segundo. Este teste foi executado para cada um dos tipos de compactações suportadas pela arquitetura: Nenhuma, Snappy, LZ4 e GZIP. Os resultados obtidos podem ser visto na Figura 32.

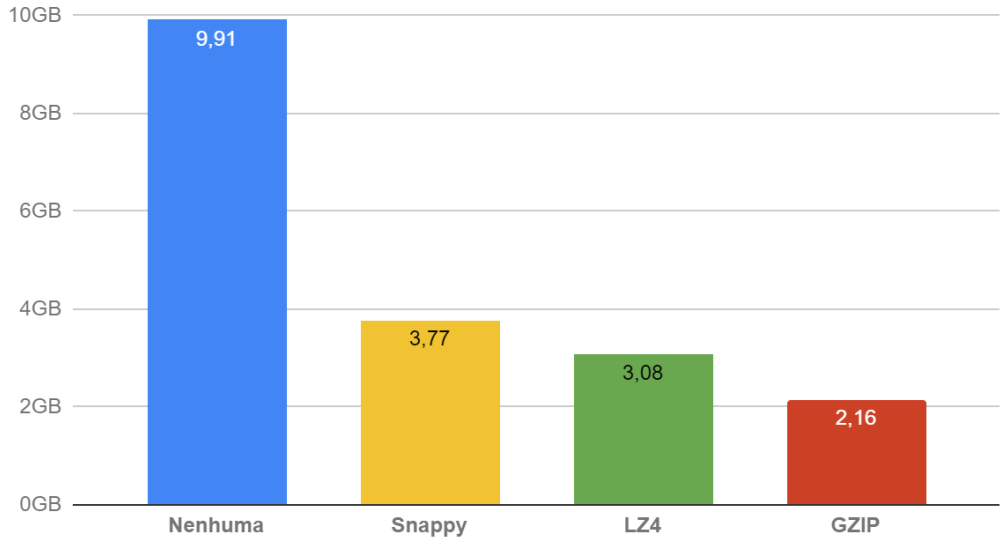

Figura 32 – Consumo de rede por algoritmos de compactação.

Na Figura 32 pode-se perceber um considerável declínio no consumo de rede gerado pelo uso dos algoritmos de compactação, se comparado ao método que não

aplica nenhuma destas estratégias. Dentre estas estratégias de compactação aplicadas, pode-se destacar o algoritmo GZIP, onde seu uso proporcionou uma redução de mais de quatro vezes o tráfego de rede gerado, destacando a capacidade da arquitetura executar em redes que tenham *links* de comunicação saturados ou de baixa velocidade. Esta redução do tráfego de rede pode ainda proporcionar uma melhora significativa no tempo de latência entre as comunicações, já que uma quantidade maior de dados pode ser trafegada no mesmo período de tempo, proporcionando a arquitetura oferecer um tempo de resposta mais curto. Tais reduções favorecem o uso em aplicações que necessitem de processamento em tempo real, com baixas latências, fornecendo a identificação de eventos complexos quase que de imediato a sua ocorrência.

Visando identificar os possíveis impactos gerados pelo uso dos algoritmos de compactação na taxa de processamento, executou-se um teste para verificar a média de eventos processada ao longo do tempo com o uso das quatro alternativas distintas de compactação empregadas. Para executar esta simulação, fez-se uso de uma máquina virtual na qual o nodo de processamento irá executar, de modo que este consuma dados continuamente pelo período de meia hora (1800 segundos) permitindo assim a verificação do fluxo de processamento durante este período. As configurações aplicadas na máquina virtual para executar esta simulação foram: um núcleo de processamento do *cluster*; seis *gigabytes* de memória RAM; e uma taxa de leitura e escrita de disco de dez *megabits* por segundo. Este teste foi executado para cada uma das técnicas de compactação onde os dados obtidos com estas execuções podem ser vistos na Figura 33.

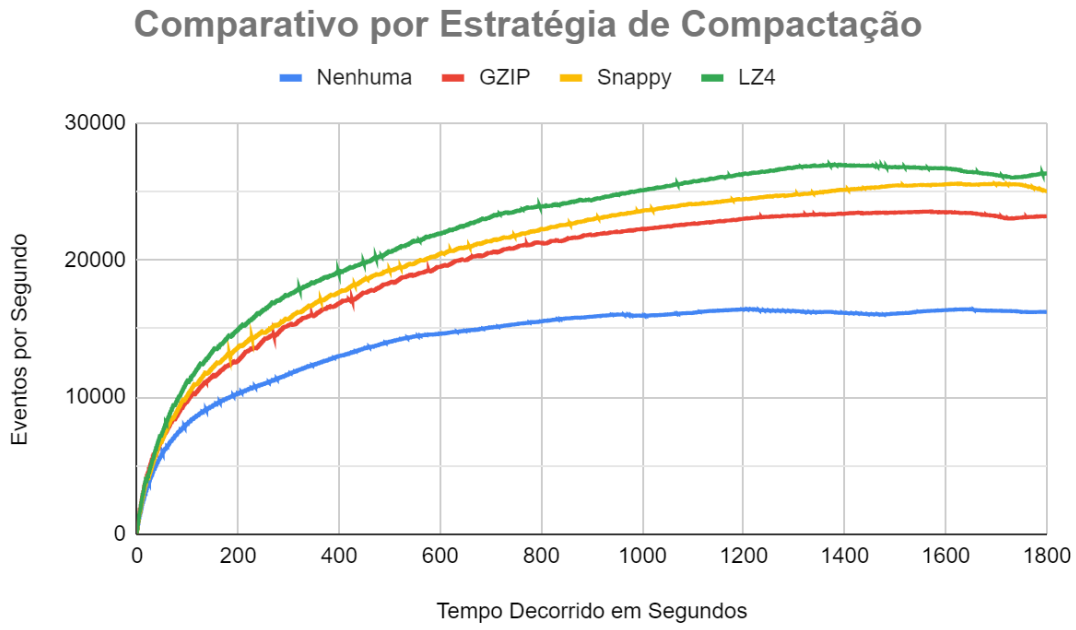

Figura 33 – Fluxo de processamento com compactação.

Na Figura 33 pode ser identificado que todos os algoritmos de compactação usados tiveram uma taxa de processamento significativamente maior se comparados com a técnica que não usava uma estratégia de compactação. Isso se justifica que mesmo estes métodos “perdendo tempo” antes de enviar ou receber os dados compactando e descompactando a informação, esta tarefa ainda não se torna o gargalo do fluxo da arquitetura, sendo este gargalo, neste caso testado, a velocidade de comunicação dos *links* entre o nodos Broker e o nodo de processamento, demonstrando que o tempo gasto para executar esta compactação dos dados é menor que o tempo ganho por conseguir transmitir uma quantidade de dados maior consumindo o mesmo percentual de largura de rede.

Os resultados obtidos, destacam ainda mais a importância da consideração do consumo de rede na distribuição dos dados, tendo em vista que mesmo uma arquitetura simplificada, com um único núcleo de processamento, a velocidade dos *links* de comunicação pode ainda ser o gargalo do fluxo de uma arquitetura. Estes dados demonstram a aplicabilidade desta arquitetura em ambientes com *links* de comunicação saturados, a qual tem o potencial de reduzir o consumo da rede e ainda proporcionar um aumento na taxa de processamento de dados.

Pode-se observar também pela Figura 33 que dentre as técnicas de compactação disponibilizadas pela a arquitetura, a LZ4 foi a que atingiu a maior taxa de processamento, sendo também a segunda com o maior nível de compactação dos dados.

7 CONSIDERAÇÕES FINAIS

O avanço tecnológico tem proporcionado mudanças significativas na sociedade, introduzindo a computação aos mais variados dispositivos, como geladeiras e cafeteiras. Esta introdução da tecnologia computacional tem sido chamada de Internet das Coisas e tem por finalidade fornecer computação com conectividade constante e mobilidade, de forma transparente e integrada ao ambiente.

Porém, em conjunto com a IoT, surge uma gama de desafios a serem solucionados, para que assim este novo paradigma computacional possa efetivamente vir a tornar-se uma tecnologia onipresente a todos os tipos de usuários. Dentre estes obstáculos se pode citar: o tratamento da heterogeneidade das informações; o processamento do considerável volume de eventos; as limitações dos ambientes em que estes dispositivos estão inseridos.

No âmbito destes desafios citados, este trabalho teve como objetivo o desenvolvimento de uma arquitetura de processamento distribuído de eventos para a IoT escalável e apta a lidar com a heterogeneidade destes ambientes, aplicando ainda estratégias que visem aprimorar a eficiência do consumo de largura de banda. Defende-se que este objetivo pôde ser alcançado com a concepção do EXEHDA-DEP.

Para que o EXEHDA-DEP dispusesse da capacidade em lidar com a heterogeneidade dos ambientes da IoT, foi adotado como escopo de desenvolvimento o *middleware* EXEHDA e o modelo arquitetural EXEHDA-SA, os quais favoreceram ao EXEHDA-DEP esta competência.

Já para que fosse possível oferecer uma arquitetura com capacidade de processar um volume de eventos considerável, como o gerado na IoT, este trabalho adotou uma abordagem distribuída na concepção do sistema, o que possibilitou que a taxa de processamento pude-se ser escalada conforme a necessidade do usuário. Esta capacidade foi evidenciada na seção 6.2, onde os resultados obtidos com testes de processamento demonstraram que com o acréscimo de um novo nodo à arquitetura, a taxa de eventos consumida por segundo aumenta significativamente, destacando a aplicabilidade do EXEHDA-DEP à IoT, visto que estes ambientes tendem a crescer consideravelmente, variando o volume de dados gerado no mesmo.

A arquitetura do EXEHDA-DEP foi ainda projetada de modo a permitir que novos nodos de processamento possam ser adicionados ou removidos sem gerar quaisquer impactos no sistema como um todo, onde os resultados obtidos com os testes executados e demonstrados na seção 6.4 enfatizam que o número de nodos processando não influencia no consumo médio de rede, onde os quais podem ser removidos e inseridos conforme a necessidade do usuário, sem produzir qualquer alteração do consumo médio de rede. O sistema permite ainda que o usuário possa adicionar em tempo de execução novas regras CEP aos nodos de processamento.

Para solucionar o desafio da aplicabilidade da arquitetura em ambientes com baixa largura de banda, foi introduzido o uso de diferentes técnicas de compactação ao EXEHDA-DEP, onde os resultados obtidos com os testes executados e demonstrados na seção 6.4, enfatizaram a capacidade destas de reduzir o consumo de rede da arquitetura projetada, onde nos quais uma redução de até quatro vezes foi obtida com o uso do algoritmo GZIP, isto quando comparada com a execução que não faz uso de nenhum tipo de compactação.

Foram também realizados testes visando demonstrar o impacto dos algoritmos de compactação na taxa de processamento dos eventos. Os resultados demonstraram que as técnicas de compactação nestes testes não reduzem a taxa de processamento de eventos, mas sim aumentam a mesma. Tais resultados levaram a identificar que, nos testes executados por este trabalho, a largura de banda é o gargalo para o processamento dos eventos complexos, salientando a aplicabilidade do EXEHDA-DEP em ambientes com baixa largura de banda, onde o mesmo possui não só a capacidade de reduzir o tráfego de rede gerado pela distribuição dos eventos complexos, mas também de incrementar a taxa do processamento dos mesmos.

Já para obter-se a competência de executar em ambientes que tenham *links* de comunicação saturados, onde perdas de pacotes podem ser recorrentes, o EXEHDA-DEP faz uso de transações, estas semelhantes às usadas em sistemas de gerenciamento de banco de dados, as quais são exemplificadas na seção 5.1. Assim, durante a execução do sistema, caso algum dos nodos de processamento não notifique o recebimento de uma informação previamente enviada, a transação que executou o despacho é imediatamente desfeita e a informação é enviada novamente. Este procedimento é de suma importância na execução do processamento de eventos complexos em ambientes que possam gerar perdas de pacotes recorrentes, já que estas podem acarretar na privação da detecção de um evento complexo, o que dependendo do cenário de aplicação pode não ser aceitável.

7.1 Contribuições

Dentre as contribuições proporcionadas por meio do desenvolvimento do EXEHDA-DEP pode-se destacar:

- o desenvolvimento de uma arquitetura distribuída de processamento de eventos complexos.
- a concepção de uma arquitetura apta a lidar com o considerável volume de eventos na IoT e sua heterogeneidade.
- proporcionar ao EXEHDA a capacidade de processar os eventos de forma distribuída e escalável.
- a concepção de uma arquitetura com capacidade de executar em ambiente com restrições de conexão.
- a possibilidade do usuário poder optar entre diferentes tipos de técnicas de compactação, selecionando a que mais se adapte as suas necessidades.
- uma arquitetura com consumo de rede constante, independente do número de nodos processando eventos.
- a capacidade de se adicionar e remover dinamicamente novos nodos de processamento, sem gerar qualquer interrupção no sistema.
- a criação e inserção de novas regras CEP em tempo de execução.

Assim, dados os objetivos almejados com a proposta deste trabalho e as contribuições previamente citadas, conclui-se que por meio da concepção do EXEHDA-DEP foi possível atingir estes determinados fins com êxito, gerando ainda contribuições adicionais.

7.2 Trabalhos Futuros

Os testes e as análises do EXEHDA-DEP foram executadas com base em um estudo de caso da segurança da informação, onde dados de *logs* de diferentes dispositivos de rede são gerados de modo a simular o ambiente da UFPel. Estes dados são analisados pelo EXEHDA-DEP por meio de regras CEP definidas pelo administrador de rede, visando assim identificar possíveis incidentes da segurança da informação.

Assim, visando proporcionar ao EXEHDA-DEP a capacidade de identificar eventos de segurança da informação de forma autônoma, sem a necessidade que o usuário especifique regras CEP para o mesmo é proposto a introdução de técnicas de inteligencia artificial de modo a favorecer a identificação de possíveis ataques os quais ainda sejam desconhecidos pelo administrador do sistema.

REFERÊNCIAS

AGRAWAL, S.; VIEIRA, D. A survey on Internet of Things. **Abakos**, Brasil, v.1, n.2, p.78–95, 2013.

ALMEIDA, R. B. et al. A distributed event-driven architectural model based on situational awareness applied on internet of things. **Information and Software Technology**, Amsterdam, Netherlands, v.111, p.144–158, 2019.

APPEL, S.; FRISCHBIER, S.; FREUDENREICH, T.; BUCHMANN, A. Event stream processing units in business processes. In: **Business Process Management**. Caceres, Spain: Springer, 2013. p.187–202.

CHEN, Y.; KUNZ, T. Performance evaluation of IoT protocols under a constrained wireless access network. In: INTERNATIONAL CONFERENCE ON SELECTED TOPICS IN MOBILE & WIRELESS NETWORKING (MOWNET), 2016., 2016, Piscataway, NJ, USA. **Anais...** IEEE, 2016. p.1–7.

CRUZ, T. et al. A cybersecurity detection framework for supervisory control and data acquisition systems. **IEEE Transactions on Industrial Informatics**, Piscataway, NJ, USA, v.12, n.6, p.2236–2246, 2016.

CUGOLA, G.; MARGARA, A. Processing flows of information: From data stream to complex event processing. **ACM Computing Surveys (CSUR)**, New York, NY, USA, v.44, n.3, p.15, 2012.

DAYARATHNA, M.; PERERA, S. Recent advancements in event processing. **ACM Computing Surveys (CSUR)**, New York, NY, USA, v.51, n.2, p.33, 2018.

FALL, K. R.; STEVENS, W. R. **TCP/IP illustrated, volume 1**: The protocols. Boston, Massachusetts, USA: addison-Wesley, 2011.

FITZGERALD, E. et al. Common Event Expression (CEE) Overview. **Report of the CEE Editorial Board**, Bedford, England, 2010.

GARG, N. **Apache Kafka**. Birmingham, England: Packt Publishing Ltd, 2013.

GONÇALVES, A. R. S. M. **Research of the internet of things business models in Portugal**. 2017. Tese (Doutorado em Ciência da Computação) — .

HANES, D. et al. **IoT fundamentals**: Networking technologies, protocols, and use cases for the internet of things. Hoboken, NJ, USA: Cisco Press, 2017.

HEINZ, C. et al. **Complex event processing (CEP) based system for handling performance issues of a CEP system and corresponding method**. Piscataway, NJ, USA: Google Patents, 2019. US Patent 10,229,162.

HERDRICH, A. J. et al. **Power efficient processor architecture**. Piscataway, NJ, USA: Google Patents, 2018. US Patent 9,870,047.

HU, D.; HONG, P.; CHEN, Y. FADM: DDoS flooding attack detection and mitigation system in software-defined networking. In: GLOBECOM 2017-2017 IEEE GLOBAL COMMUNICATIONS CONFERENCE, 2017, Piscataway, NJ, USA. **Anais...** IEEE, 2017. p.1–7.

JOSHI, C.; SINGH, U. K. Information security risks management framework–A step towards mitigating security risks in university network. **Journal of Information Security and Applications**, Amsterdam, Netherlands, v.35, p.128–137, 2017.

KAMIENSKI, C. et al. Smart water management platform: Iot-based precision irrigation for agriculture. **Sensors**, Basel, Switzerland, v.19, n.2, p.276, 2019.

KOTENKO, I. V.; SAENKO, I.; KUSHNEREVICH, A. Parallel big data processing system for security monitoring in Internet of Things networks. **JoWUA**, Dobong-gu, Korea, v.8, n.4, p.60–74, 2017.

KRUMM, J. **Ubiquitous computing fundamentals**. London, England: Chapman and Hall/CRC, 2016.

LOPES, J. L. et al. A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing. **J. UCS**, San Diego, USA, v.20, n.9, p.1327–1351, 2014.

MACHADO, R. d. S. et al. EXEHDA-HM: A compositional approach to explore contextual information on hybrid models. **Future Generation Computer Systems**, Amsterdam, Netherlands, v.73, p.1–12, 2017.

MARTINS, I. R.; ZEM, J. L. Estudo dos protocolos de comunicação MQTT e COaP para aplicações machine-to-machine e Internet das coisas. **Revista Tecnológica da Fatec Americana**, Brasil, v.3, n.1, p.24p–24p, 2015.

MARTINS, I. R.; ZEM, J. L. Estudo dos protocolos de comunicação MQTT e COaP para aplicações machine-to-machine e Internet das coisas. **Revista Tecnológica da Fatec Americana**, Brasil, v.3, n.1, p.24, 2016.

MENG, X. et al. Mllib: Machine learning in apache spark. **The Journal of Machine Learning Research**, New York, NY, USA, v.17, n.1, p.1235–1241, 2016.

MINBO, L.; ZHU, Z.; GUANGYU, C. Information service system of agriculture IoT. **automatika**, England, v.54, n.4, p.415–426, 2013.

NGU, A. H. et al. IoT middleware: A survey on issues and enabling technologies. **IEEE Internet of Things Journal**, Piscataway, NJ, USA, v.4, n.1, p.1–20, 2016.

NOCERA, F.; DI NOIA, T.; MONGIELLO, M.; DI SCIASCIO, E. Semantic IoT Middleware-enabled Mobile Complex Event Processing for Integrated Pest Management. In: CLOSER, 2017, Setubal, Portugal. **Anais...** SCITEPRESS, 2017. p.610–617.

PARK, T.; ABUZAINAB, N.; SAAD, W. Learning how to communicate in the Internet of Things: Finite resources and heterogeneity. **IEEE Access**, Piscataway, NJ, USA, v.4, p.7063–7073, 2016.

PÉREZ-VEREDA, A.; FLORES-MARTÍN, D.; CANAL, C.; MURILLO, J. M. Complex Event Processing for health monitoring. In: INTERNATIONAL WORKSHOP ON GERONTECHNOLOGY, 2018, Caceres, Spain. **Anais...** Springer, 2018. p.3–14.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering : An update. **Information and Software Technology**, Alberta, Canada, v.64, p.1–18, 2015.

PREUVENEERS, D.; BERBERS, Y.; JOOSEN, W. SAMURAI: A batch and streaming context architecture for large-scale intelligent applications and environments. **Journal of Ambient Intelligence and Smart Environments**, Clifton, USA, v.8, n.1, p.63–78, 2016.

RAZZAQUE, M. A.; MILOJEVIC-JEVRIC, M.; PALADE, A.; CLARKE, S. Middleware for internet of things: a survey. **IEEE Internet of things journal**, Piscataway, NJ, USA, v.3, n.1, p.70–95, 2015.

RUIZ-RUBE, I. et al. Block-Based Development of Mobile Learning Experiences for the Internet of Things. **Sensors**, Puerto Real, Spain, v.19, n.24, p.5467, 2019.

SHORO, A. G.; SOOMRO, T. R. Big data analysis: Apache spark perspective. **Global Journal of Computer Science and Technology**, USA, 2015.

SONI, D.; MAKWANA, A. A survey on mqtt: a protocol of internet of things (iot). In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATION, POWER ANALYSIS AND COMPUTING TECHNIQUES (ICTPACT-2017), 2017, Germany. **Anais...** ResearchGate, 2017.

SOTO, J. A. C.; JENTSCH, M.; PREUVENEERS, D.; ILIE-ZUDOR, E. CEML: Mixing and moving complex event processing and machine learning to the edge of the network for IoT applications. In: INTERNATIONAL CONFERENCE ON THE INTERNET OF THINGS, 6., 2016, New York, NY, USA. **Proceedings...** ACM, 2016. p.103–110.

SUHOTHAYAN, S. et al. Siddhi: A second look at complex event processing architectures. In: ACM WORKSHOP ON GATEWAY COMPUTING ENVIRONMENTS, 2011., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.43–50.

WANG, Q.; SHANG, Y. A Distributed Complex Event Processing System Based on Publish/Subscribe. In: **Recent Developments in Intelligent Computing, Communication and Devices**. Singapore: Springer, 2019. p.981–990.

XAVIER, M. S. R. d. B. **Smart Homes no mercado downstream de Oil & Gas**. 2016. Dissertação (Mestrado em Ciência da Computação) — FEUC.

YAMIN, A. et al. EXEHDA: adaptive middleware for building a pervasive grid environment. In: SELF-ORGANIZATION AND AUTONOMIC INFORMATICS (I), 2005., 2005, New York, NY, USA. **Proceedings...** ACM, 2005. p.203–219.

YANG, S. IoT stream processing and analytics in the fog. **IEEE Communications Magazine**, Piscataway, NJ, USA, v.55, n.8, p.21–27, 2017.