

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Sandy Ferreira da Costa Bezerra

**Arquitetura de Fusão Hierárquica de Dados para Sistemas de
Computação em Névoa**

Fortaleza – CE

2 de fevereiro de 2021

Sandy Ferreira da Costa Bezerra

Arquitetura de Fusão Hierárquica de Dados para Sistemas de Computação em Névoa

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Orientador: Atslands Rego da Rocha

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

2 de fevereiro de 2021

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

- B574a Bezerra, Sandy Ferreira da Costa.
Arquitetura de Fusão Hierárquica de Dados para Sistemas de Computação em Névoa / Sandy Ferreira da Costa Bezerra. - 2020.
62 f. : il. color.
- Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus Fortaleza, 2020.
Orientação: Profa. Dra. Atslands Rego da Rocha.
1. Computação em Névoa. 2. Internet das Coisas. 3. Fusão de Dados. 4. Processamento de Eventos Complexos. I. Título.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Sandy Ferreira da Costa Bezerra

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

Orientador: Atslands Rego da Rocha

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Carina Teixeira de Oliveira

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Flavia Coimbra Delicato

Universidade Federal Fluminense (UFF)

Paulo Armando Cavalcante Aguilár

Universidade Federal do Ceará (UFC)

Fortaleza – CE
2 de fevereiro de 2021

Dedicatória

À minha família, por sua capacidade de acreditar e investir em mim. Mãe, seu cuidado e dedicação foram fundamentais para que eu chegasse até aqui. Amor, obrigada pelo carinho, paciência, suporte e por sua capacidade de me trazer paz na correria que é nossa vida.

Agradecimentos

Aos meus pais, irmão e sobrinhos, pelo amor, incentivo e apoio incondicional e em especial ao meu marido que nas horas difíceis de desânimo esteve ao meu lado e nos momentos de minha ausência dedicados ao estudo, sempre me fez entender que o futuro é feito a partir da constante dedicação no presente!

A Prof. Dr. Atslands Rego da Rocha por me orientar, pela paciência, pelas correções e incentivos.

A todos os professores por me proporcionarem o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

"O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis"

José de Alencar

Resumo

O paradigma da computação em névoa possibilita que os recursos computacionais sejam levados para a borda da rede, de forma que a latência e a largura de banda resultantes da computação em nuvem possam ser reduzidas. A combinação desse paradigma com técnicas de fusão de dados e processamento de eventos, a partir da aquisição de dados brutos, tem grande potencial de uso para gerar conhecimento de valor e ajudar a tomada de decisão em sistemas de Internet das Coisas (IoT). Nesse contexto, foi proposta uma arquitetura de fusão de dados em três níveis (nó sensor, névoa e nuvem) que, aliada ao uso de técnicas de processamento de eventos complexos, promove a tomada de decisões com menor latência, enquanto reduz o custo de transmissão de dados na rede. Os experimentos mostram uma redução de 77% no tempo médio de envio de mensagens na rede e 82% de melhoria na vazão média com a utilização da arquitetura proposta. Além disso, foi realizada a validação e auditoria de um modelo de Processamento de Eventos Complexos (CEP) baseado em conjuntos de regras.

Palavras-chave: Computação em Névoa; Internet das Coisas; Fusão de Dados; Processamento de Eventos Complexos.

Abstract

The fog computing paradigm enables computational resources to be provided at the edge of the network so that the latency and bandwidth resulting from cloud computing can be reduced. Combining this paradigm with data fusion and Complex Event Processing (CEP) techniques from raw data acquisition has excellent potential for generating valuable knowledge and aiding decision making processes in IoT systems. In this context, we propose a three-level data fusion architecture (sensor node, fog, and cloud) that, combined with CEP techniques, promotes timely decisions while reducing the cost of data transmission in the network. The experiments show a reduction of 77 % in the average time of sending messages in the network and an improvement of 82 % in the average flow rate when the proposed architecture is used. Besides, an event processing validation and auditing of a CEP model based on rule sets were executed.

Keywords: Fog computing; Internet of Things. Fusion Data; Complex Event Processing.

Lista de ilustrações

Figura 1 – Comportamento da Camada da Névoa	32
Figura 2 – Categorias da Fusão com Base no Relacionamento Entre as Fontes dos Dados (Fonte: (Torres, 2017))	33
Figura 3 – Primitivas CEP	37
Figura 4 – Arquitetura conceitual da fusão de dados hierárquicos em três níveis. .	43
Figura 5 – Etapas de Processamento de Eventos	46
Figura 6 – Ciclo de Processamento de Eventos	46
Figura 7 – Arquitetura e Primitivas CEP	48
Figura 8 – Primitiva CEP de acordo com as camadas	49
Figura 9 – Gerência de um Nó Adicionado a Rede	50
Figura 10 – Topologia da Rede: Cenário 1 e 2	58
Figura 11 – Topologia da Rede 3: Situação Inicial e Final da Simulação	58
Figura 12 – Vazão dos Cenários 1 e 2	61
Figura 13 – Vazão do Cenário 3	62
Figura 14 – Tempo Médio de Envio de Mensagens nos 3 Cenários	62
Figura 15 – Fluxo Iniciando com o Evento 1	63
Figura 16 – Fluxo Iniciando com o Evento 2	64
Figura 17 – Fluxo Iniciando com o Evento 3	64
Figura 18 – Fluxos do Modelo CEP	65
Figura 19 – Logs dos Eventos 1, 2 e 3	66
Figura 20 – Log do Evento 6	67
Figura 21 – Logs dos Eventos 8, 9 e 10	67

Lista de tabelas

Tabela 1	–	Questões de Pesquisa	28
Tabela 2	–	Padrões de Eventos Complexos	35
Tabela 3	–	Exemplos de Arquiteturas CEP	38
Tabela 4	–	Tabela Comparativa dos Trabalhos Relacionados	42
Tabela 5	–	Experimentos	59
Tabela 6	–	Validação do Modelo CEP	65

Lista de abreviaturas e siglas

IoT - *Internet of Things*
CEP - *Complex Events Processing*
PA - *Precision Agriculture*
RE - *Rules Engine*
CPU - *Central Process Unit*
RAM - *Random Access Memory*
RSSF - Redes de Sensores Sem Fio
CEE - *Collaborative Edge Environment*
GPU - *Graphics Processing Unit*
EPA - *Event Processing Agent*
CQL - *Continuous Query Language*
SQL - *Structured Query Language*
ECA - Evento-Condição-Ação
LDF - *Low Data Fusion*
MDF - *Medium Data Fusion*
HDF - *High Data Fusion*
DF - *Data Fusion*
CC - Condição de Campo
TR - Turno de Rega
Et₀ - Evapotranspiração
ITN - Irrigação Total Necessária
EV - Evento
ID - Identificador
INMET - Instituto Nacional de Meteorologia
YAFS - *Yet Another Fog Simulator*
NS - Nó Sensor
NN - Nó Névoa
MB - *Megabyte*
GB - *Gigabyte*
RNA - Regra Não Alcançável
PND - Processamento Não Determinístico

Lista de símbolos

ψ - Potencial Mátrico

θ - Umidade

Sumário

1	INTRODUÇÃO	25
1.1	Motivação para a Dissertação	27
1.2	Questões de Pesquisa e Hipóteses	28
1.3	Objetivos Geral e Específicos	29
1.4	Produção científica	29
1.4.1	Artigo Publicado	29
1.5	Estrutura da Dissertação	29
2	FUNDAMENTOS	31
2.1	Computação em Névoa	31
2.2	Fusão de Dados	32
2.3	Processamento de Eventos Complexos (CEP)	34
2.3.1	Primitivas de Processamento	36
3	TRABALHOS RELACIONADOS	39
3.1	Metodologia da Revisão Sistemática	39
3.2	Trabalhos Selecionados	39
4	UMA ARQUITETURA HIERÁRQUICA DE FUSÃO DE DADOS EM NÉVOA	43
4.1	Definição das Camadas da Arquitetura	43
4.2	Definição das Regras e Eventos	47
4.2.1	Regra de Monitoramento	47
4.2.2	Regra de Estado Crítico	47
4.3	Definição da Arquitetura CEP	48
4.4	Definição da Autoconfiguração dos Nós Sensores	49
5	ESTUDO DE CASO: AGRICULTURA DE PRECISÃO	51
5.1	Estudo de Caso: Regras de Monitoramento	52
5.2	Estudo de Caso: Regras de Estado Crítico	53
5.3	Estudo de Caso: Definição da Arquitetura CEP	53
6	METODOLOGIA	57
6.1	Prova de Conceito: Arquitetura da Simulação e seus componentes	57
6.2	Validação e Auditoria do Modelo de Processamento de Eventos Complexos	59

7	AVALIAÇÃO EXPERIMENTAL: RESULTADOS E ANÁLISE	61
7.1	Experimento 1: Métricas de Desempenho	61
7.2	Experimento 2: Validação e Auditoria do Modelo CEP	63
7.2.1	Validação	63
7.2.2	Auditoria	65
8	CONCLUSÃO	69
8.1	Trabalhos Futuros	70
	REFERÊNCIAS	71

1 Introdução

A Internet das coisas (IoT, do inglês *Internet of Things*) propicia a instrumentação do mundo físico e sua conexão com o virtual a uma escala sem precedentes. O recente crescimento de serviços e aplicações da Internet contribuiu para um aumento nos requisitos de armazenamento e processamento de dados. Eles são diversificados em termos de recursos exigidos por diferentes aplicações, portanto, muitas vezes requerem soluções personalizadas. A computação de borda (*Edge Computing*) veio para superar algumas das limitações do uso da computação em nuvem como plataforma de *backend* para a IoT. Tais limitações incluem a latência imprevisível e a falta de percepção quanto a localização e mobilidade do usuário e de suas aplicações (Bierzynski; Escobar; Eberl, 2017). O conceito de computação de borda visa aproveitar as capacidades de armazenamento e processamento de um grande número de dispositivos IoT, *gateways*, estações radio-base, e de outros dispositivos de borda conectados à internet, fornecendo uma camada intermediária entre os dispositivos finais e a nuvem. Segundo (Caprolu et al., 2019a), o paradigma da computação em borda inclui três tipos de dispositivos: os que geram os dados brutos (sensores); os dispositivos que oferecem a capacidade de processamento e armazenamento ou outros serviços (ex: *gateways*) e os que recebem os dados processados (ex: *smartphones*). Com a presença dos dispositivos de borda, o envio de dados e a carga de computação na nuvem são reduzidos, uma vez que algumas das solicitações são processadas localmente, não necessitando, assim, da intervenção da camada superior. Esse aspecto, por sua vez, reduz o tempo de resposta da rede e permite a manipulação de grande parte dos dados nas camadas mais próximas do local monitorado.

O paradigma de computação de borda foi aplicado a várias tecnologias emergentes e adotados por aplicações IoT que tiram proveito dos diferentes recursos oferecidos pela computação distribuída. Uma das concretizações desse paradigma é a computação em névoa (*Fog Computing*), que é uma infraestrutura de computação descentralizada que consiste em nós de processamento na névoa, ou seja, colocados em qualquer ponto entre os dispositivos finais e a nuvem. De acordo com (Mouradian et al., 2018), a névoa é uma "nuvem mais próxima do solo". Ela estende a arquitetura tradicional de computação em nuvem até a borda da rede, permitindo o processamento e a análise de alguns dados nesse nível. Os nós na névoa são heterogêneos por natureza e, portanto, podem se basear em diferentes tipos de elementos, incluindo, mas não se limitando a, roteadores, pontos de acesso e *gateways* IoT. A heterogeneidade dos nós é oculta dos dispositivos finais, ao ser apresentada uma camada uniforme de abstração da névoa, que é composta por um conjunto de funções que fazem alocação e monitoramento de recursos, gerenciamento de dispositivos, juntamente com serviços de armazenamento e processamento. Essas funções são providas

por uma camada de organização de serviço, que recebe solicitações dos usuários finais e aloca recursos de acordo com os requisitos das solicitações.

No ambiente da Internet das coisas, os dados brutos são frequentemente gerados por dispositivos em uma frequência muito alta. Para extrair informações significativas, precisa ser executado algum trabalho de pré-processamento nos dados brutos. Embora as técnicas de fusão de dados tradicionais também tenham capacidade de processar e extrair informações dos dados, a combinação com outras técnicas de processamento melhora ainda mais sua eficiência. A fusão de dados (Nakamura; Loureiro; A.Frery, 2007) pode ser entendida como um processo de agregação e combinação de dados provenientes de múltiplas fontes de modo a evitar redundância, reduzindo a quantidade de dados que circula pela rede. A fusão de dados também pode contribuir para a melhoria da qualidade dos dados, utilizando dados de várias fontes para compor uma informação mais completa. Além da fusão, técnicas de processamento de eventos complexos (CEP, do inglês *Complex Event Processing*) também fornecem contribuições para a melhoria da informação gerada. O CEP vai além da simples consulta e transformação de dados e visa detectar padrões nos dados utilizando técnicas de filtragem, correlação e criação de novos eventos. Assim, os dados fusionados podem ser vistos como notificações de eventos que ocorrem no mundo externo utilizados para composição de eventos mais complexos.

Dentro desse contexto, neste trabalho é proposta uma arquitetura de fusão de dados aplicada em ambientes de computação em névoa, a qual atua em conjunto com o paradigma de CEP como uma solução para a alta latência e tempo de resposta em ambientes de computação em nuvem e seu uso em sistemas e aplicações de IoT. A fusão de dados é necessária para reduzir a quantidade de mensagens trocadas entre a área sensoreada e a nuvem. A ideia principal da proposta é identificar a ocorrência de um evento que desencadeia um processo de fusão de dados, visando identificar os nós que estão relacionados ao evento e extrair as características do local sensoreado. Dessa forma, informações são processadas em camadas diferentes, de acordo com a capacidade de cada dispositivo, otimizando o uso de recursos computacionais e diminuindo o tempo médio de envio de dados, tanto entre o sensor e a névoa quanto entre a névoa e a nuvem. Os dados são coletados, gerenciados e processados de acordo com os requisitos da aplicação e de forma descentralizada, onde diferentes níveis são responsáveis pela coleta, análise, tomada de decisão e suporte a sistemas IoT. Para dar suporte ao ambiente de computação de borda, neste trabalho é proposta também uma abordagem descentralizada de gerenciamento flexível para a autoconfiguração dos nós sensores em relação às névoas de acordo com parâmetros de distância e tipos de dados monitorados. Dessa forma, nós sensores podem ser adicionados e configurados rapidamente para se adaptar às alterações no ambiente.

As contribuições deste trabalho são (i) a proposta de uma abordagem hierárquica de fusão de dados visando tomadas de decisões oportunas; (ii) o uso de um mecanismo de

processamento de eventos complexos (CEP), incluindo o processamento de vários eventos simples; (iii) a implementação de um mecanismo que permite a adaptação a mudanças físicas no ambiente monitorado, baseado no tipo de dado medido e na distância entre o sensor e a névoa.

Para avaliação da arquitetura proposta, foram simulados cenários na agricultura de precisão como estudo de caso. Em geral, os campos possuem sensores que monitoram as plantações e *gateways* IoT/servidores locais que se conectam a um servidor remoto ou à nuvem. Entretanto, a conectividade pode ser um fator limitante, portanto, o uso de técnicas que forneçam informações confiáveis e baixo custo de transmissão de dados é de suma importância. Os experimentos mostram uma redução de 77% no tempo médio de envio de mensagens na rede e 82% de melhoria na vazão média com a utilização da arquitetura proposta em comparação com sistemas que não usam essas técnicas na borda. Além disso, foi realizada a validação e auditoria do modelo CEP utilizado como prova de conceito do trabalho.

1.1 Motivação para a Dissertação

Um desafio atual ao usar redes de sensores inteligentes é a comunicação com serviços em nuvem (Merkl; Cocos, 2019). Como os sistemas embarcados são comumente usados em redes de sensores, eles geralmente não têm recursos de hardware suficientes para realizar o monitoramento, a análise e a visualização dos dados. Além disso, a comunicação é desafiadora, pois a latência da rede é um fator limitante, especialmente no campo de aplicações em tempo real. Uma das abordagens propostas como solução desse problema é a computação de borda (Shi et al., 2016). Na computação de borda, os dispositivos finais (neste contexto, dispositivos IoT) estão se comunicando com uma camada intermediária que é a camada de borda. Os dispositivos de borda estão localizados próximos aos dispositivos finais, reduzindo a latência da rede e tornando possíveis as aplicações em tempo real.

A camada de borda como uma camada intermediária pode ou não se comunicar com os serviços em nuvem. Um cenário possível poderia ser a agregação de dados produzidos pelos dispositivos IoT e armazenados em um dispositivo de borda. O dispositivo na névoa pode, como uma das próximas etapas, transmitir os dados para um serviço em nuvem e usar os recursos disponíveis, como armazenamento, poder de computação, etc. A nuvem pode ser usada para executar tarefas que consomem mais recursos. As tarefas podem ser, por exemplo, a filtragem de grandes conjuntos de dados ou a aplicação de métodos de aprendizado de máquina. Os desafios no uso da computação de borda são a seleção de dispositivos apropriados para as aplicações executadas nos nós, a descarga de tarefas nos dispositivos de borda, alocação automatizada de tarefas entre os nós de borda e a camada de nuvem, etc (El-Sayed et al., 2018).

Os desafios mencionados e a falta de padronização da computação de borda vêm resultando em esforços de pesquisa e implementação de redes a fim de solucioná-los. A computação de borda tem muitos benefícios em cenários de uso onde o tempo de resposta é um fator limitante para arquiteturas de computação em nuvem, uma vez que a latência de rede e os tempos de resposta costumam ser altos para uma oferta de serviço confiável. Assim, os benefícios da computação em névoa entram em jogo, pois os requisitos de serviço exigem um acesso mais rápido aos dados do sensor ou atuadores e a redução da latência da rede. No entanto, é difícil estabelecer um padrão, pois muitas aplicações implementam a computação de borda de forma diferente, por exemplo, aplicações domésticas inteligentes (El-Sayed et al., 2018) ou a detecção de eventos catastróficos usando processamento de eventos complexos (Lan et al., 2019). O serviço de infraestrutura em nuvem usado tem o nível mais alto de abstração, enquanto a camada de borda está localizada mais perto dos dispositivos finais. A camada de névoa localizada entre a nuvem e a camada de sensor é usada para processar os dados dos dispositivos finais. Os processos executados na camada de névoa têm um nível de abstração mais alto em comparação com os da camada do sensor e, conseqüentemente, mais recursos de hardware são necessários.

Dessa maneira, o esforço de distribuir o processamento dos dados de modo a melhorar os requisitos relacionados ao tráfego (como latência e tempo de repostas) e que auxiliem na tomada de decisão das aplicações pode prover benefícios dentro dessa problemática. A partir dessa motivação foram levantadas as questões de pesquisas apresentadas na seção 1.2.

1.2 Questões de Pesquisa e Hipóteses

A partir do problema levantado neste capítulo, foram levantadas as seguintes questões de pesquisa (Tab. 1) e suas respectivas hipóteses:

Tabela 1 – Questões de Pesquisa

1	Quais técnicas de gerenciamento de dados melhoram o tráfego na rede e auxiliam na tomada de decisão?
2	Quais parâmetros podem auxiliar em uma autoconfiguração eficiente de nós?

Hipótese 1: A fusão dos dados hierárquica juntamente com o processamento de eventos complexos reduz a quantidade de dados que trafega na rede e auxilia a tomada de decisão mais acurada em ambientes de névoa.

Hipótese 2: Uma eleição de nó de borda com base no contexto da aplicação e na distância entre os nós melhora a eficiência do ambiente.

1.3 Objetivos Geral e Específicos

O objetivo geral da pesquisa é propor uma arquitetura hierárquica de fusão de dados para ambientes de computação em névoa, em que os nós da borda são eleitos de acordo com o contexto da aplicação e distância do nó sensor, a fim de reduzir o tráfego de dados na rede, melhorar a latência e oferecer suporte a tomada de decisão.

Para alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

- Propor técnicas de fusão de dados para reduzir a quantidade de dados transmitidos para a nuvem;
- propor técnicas CEP para criar e processar eventos relevantes referentes aos dados monitorados;
- Especificar um mecanismo de eleição de nós da névoa com base no contexto da aplicação e na distância entre os sensores e a névoa.

1.4 Produção científica

1.4.1 Artigo Publicado

Bezerra, S. et al. Fusão hierárquica de dados para sistemas de computação em névoa. In: Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Porto Alegre, RS, Brasil: SBC, 2020. p. 714–727. ISSN 2177-9384. Disponível em: <<https://sol.sbc.org.br/index.php/sbrc/article/view/12320>>.

1.5 Estrutura da Dissertação

Os capítulos restantes desta dissertação estão organizados da seguinte maneira:

- **Capítulo 2 - Trabalhos Relacionados:** apresenta uma revisão sistemática de trabalhos similares que abordam arquiteturas de computação em névoa e processamento de eventos complexos;
- **Capítulo 3 - Fundamentos:** aborda uma revisão da literatura sobre os principais conceitos que fundamentaram esta dissertação;
- **Capítulo 4 - Uma Arquitetura Hierárquica de Fusão de Dados em Névoa:** apresenta a proposta desta dissertação;
- **Capítulo 5 - Metodologia:** detalha o design experimental realizado, com os procedimentos seguidos, descrição do ambiente, implementação e os cenários dos experimentos;
- **Capítulo 6 - Avaliação Experimental: Resultados e Análise:** apresenta a análise de desempenho dos experimentos realizados e a validação e auditoria

do modelo CEP proposto;

- **Capítulo 7 - Conclusão:** traz a conclusão desta dissertação, objetivos alcançados, contribuições e trabalhos futuros.

2 Fundamentos

Este capítulo apresenta uma revisão da literatura acerca dos principais conceitos utilizados como base da pesquisa.

2.1 Computação em Névoa

Computação em Névoa (*Fog Computing*) foi um termo introduzido pela Cisco em 2013. A empresa descreve essa tecnologia como uma coleção de recursos que visa aumentar a capacidade de processamento e armazenamento em nuvem para suportar (mas não se limitar a) a IoT. Em seu sentido literal, a névoa é uma nuvem ao nível do solo. Logo, a computação em névoa é a aproximação da nuvem aos dispositivos que recolhem e transmitem dados. De fato, a computação em névoa provê o processamento de dados em uma camada mais próxima dos dispositivos finais.

A baixa latência exigida pelas aplicações modernas impulsionou uma alteração na arquitetura clássica da nuvem, onde os dados são enviados, na maioria das vezes, para um local geograficamente mais afastado para serem processados, aumentando assim o atraso da rede. Na arquitetura de computação em borda, os dados são processados no mesmo dispositivo que os produziu ou em um dispositivo externo localizado na borda da mesma rede, podendo também ser processados fora da rede, em uma localização geográfica mais próxima da origem dos dados.

A computação em névoa utiliza os chamados dispositivos de borda para realizar processamento, armazenamento e comunicação localmente. De acordo com (Caprolu et al., 2019b), esses dispositivos variam de pequenas plataformas IoT, que compreendem um único núcleo de CPU e menos de um megabyte de RAM, a servidores de pleno direito, compreendendo CPUs multicore, GPUs de vários núcleos e Gigabytes de RAM. A computação em névoa pode ser entendida como uma extensão da computação em nuvem e isso implica em uma série de características como descritas em (Coutinho; Carneiro; Greve, 2016):

- *Reconhecimento de localidade e baixa latência*
- *Distribuição geográfica*
- *Suporte a redes de sensores em larga escala*
- *Grande número de nós*
- *Suporte a computação móvel*
- *Interações em tempo real*

A computação em névoa adiciona uma nova camada entre a nuvem e os dispositivos

que coletam os dados. Essa nova camada, mais próxima dos dispositivos finais e iniciais, diminui a necessidade acesso a nuvem, como mostra a Figura 1. Portanto, a computação em borda se mostra uma arquitetura adequada para executar aplicativos em tempo real e serviços inteligentes de IoT.

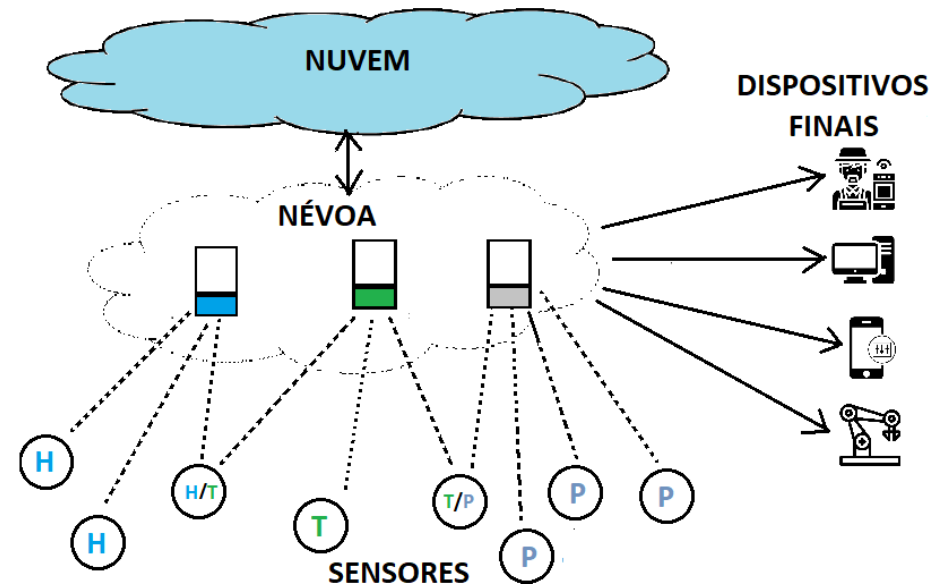


Figura 1 – Comportamento da Camada da Névoa

A arquitetura de ambiente em névoa possui diferentes níveis (sensor, névoa e nuvem). O nível baixo é composto pelos objetos inteligentes que coletam dados, os pré-processam e encaminham, caso necessário, para os nós da névoa. O nível médio é composto pelos nós de borda com um maior poder de processamento, quando comparados ao nível anterior, onde ocorre processamento massivo dos dados, além disso, é feita a atuação e a tomada de decisão sobre os dados que serão enviados e armazenados na nuvem. Na camada alta é feito o armazenamento dos dados persistentes, a fusão de informações (visão holística) e o processamento dos dados que os dispositivos da névoa não tiveram capacidade suficiente para processar. Devido a heterogeneidade dos objetos IoT e a sua localização na topologia da rede, os dados provenientes desses dispositivos devem ser processados no próprio nó sempre que possível, ou enviados para as unidades de processamento da névoa e finalmente, as informações persistentes são encaminhadas para os *datacenters* na nuvem, onde são agregadas a informações obtidas de outros banco de dados, seguindo o padrão da arquitetura de três níveis.

2.2 Fusão de Dados

A fusão de dados pode ser definida como a combinação de dados de múltiplos sensores com o intuito de prover informações mais específicas do que usando apenas um desses recursos individualmente. De acordo com (Zhang et al., 2018), a fusão de dados é

um pré-requisito para garantia de qualidade e mineração analítica de dados integrados. A fusão pode ser classificada levando em conta diversos aspectos. A classificação mais comum é a baseada no relacionamento entre as fontes dos dados, onde não existe dependência direta entre os sensores, ou seja, eles funcionam de maneira descentralizada. Esse relacionamento pode ser categorizado em (Nakamura; Loureiro; A.Frery, 2007): complementar, onde os sensores fornecem informações de partes diferentes de um cenário; redundante, em que os sensores fornecem o mesmo tipo de informação de um mesmo cenário; e cooperativo, onde as informações de sensores diferentes podem ser unidas para gerar uma nova informação (Fig 2).

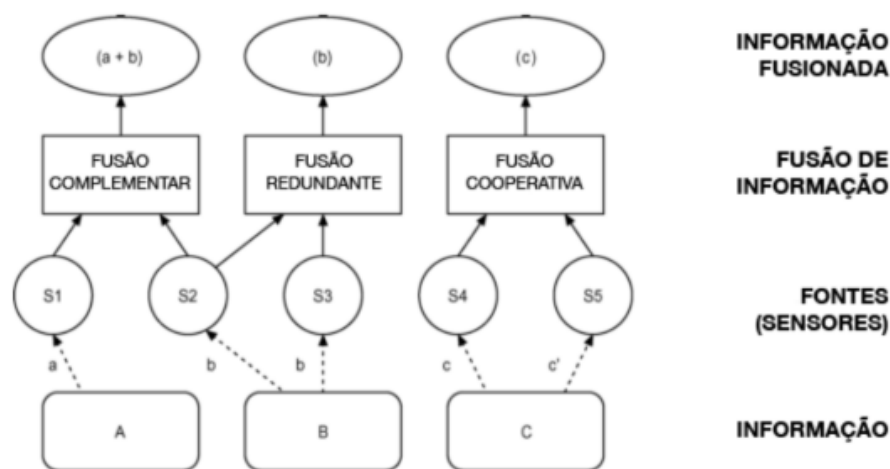


Figura 2 – Categorias da Fusão com Base no Relacionamento Entre as Fontes dos Dados (Fonte: (Torres, 2017))

A fusão complementar busca a totalidade, compondo novas informações de diferentes partes. Um exemplo de fusão complementar consiste em fusionar dados de nós sensores (por exemplo, uma amostra do sensor de umidade do solo no campo agrícola) em um mapa que descreve todo o comportamento do campo, portanto, resultando em informações mais amplas e completas. A fusão redundante pode ser usada para aumentar a confiabilidade e a precisão das informações como, por exemplo, sensores com medições de valores uniformes de umidades do solo fazendo o envio do dado apenas quando houver alteração no valor. Nas redes IoT, a fusão redundante pode fornecer informações de alta qualidade e impedir que os nós sensores transmitam informações redundantes o que resulta na economia de recursos. A fusão cooperativa também pode ser usada para reduzir o fluxo de dados em uma rede, visto que cada sensor é responsável apenas pelo envio de um tipo de informação. Um exemplo clássico de cooperação é o cálculo de um local de destino com base em informações de ângulo e distância. A fusão cooperativa deve ser aplicada com cuidado, pois os dados resultantes estão sujeitos às imprecisões e imperfeições de todas as fontes participantes.

Segundo (Nakamura; Loureiro; A.Frery, 2007), de acordo com o nível de abstração

dos dados manipulados, a fusão de informações é realizada em três categorias:

- **Fusão de baixo nível:** também conhecida como fusão de nível de sinal (medição), onde os dados brutos são fornecidos como entradas e combinados em novos dados mais precisos do que as entradas individuais. Nesse nível, a fusão ocorre nos nós sensores (fontes de dados) onde os dados estão sendo coletados.
- **Fusão de nível médio:** onde é feita a computação em um nível mais alto que o anterior, que pode ser executada com as informações pré-processadas, ou seja, os resultados do estágio de processamento anterior, no nó sensor, por exemplo, para obter estimativas de parâmetros de domínio espacial em uma determinada área ou implementar a extração de padrão ou ainda realizar cálculos redundantes.
- **Fusão de alto nível:** implementa o último estágio de processamento no nível mais alto de abstração em um domínio global, realizando inferência ou raciocínio complexo e tomada de decisão baseada nos dados provenientes das camadas inferiores ou implementando fusão espaço-temporal, por exemplo, explorando tarefas de cooperação.

2.3 Processamento de Eventos Complexos (CEP)

Antes de entender o processamento de eventos complexos, é necessário entender e definir o que é um evento. Segundo (Cugola; Margara, 2012), um evento pode ser entendido como a ocorrência de algo, uma "notificação". Por exemplo, alterações da umidade do solo e temperatura podem ser relacionados através do tempo agregação e causalidade. Para possibilitar a detecção desses eventos é possível implementar consultas em um estado contínuo e as executar com o intuito de analisar e reagir aos eventos assim que eles aparecem no fluxo dos dados. Essas consultas são denominadas fluxos de eventos e são as principais fontes de entrada na execução do processamento de eventos (Cugola; Margara, 2012).

Os eventos são criados pelos produtores, que são entidades (usuários, sensores) que encapsulam os dados do domínio da aplicação. Esses eventos, antes de serem processados, são denominados eventos simples (*raw events*). Novos eventos podem ser formados a partir da agregação de eventos simples e estes são chamados de eventos complexos. Segundo (M.Roriz et al., 2019), o processamento de eventos complexos é um paradigma de programação voltado a tratar, analisar e reagir a um fluxo de dados (encapsulados como eventos) quase em tempo real, isto é, em poucos segundos. Sendo assim, o resultado dos eventos criados e transformados pelos produtores é o fluxo de eventos, ou seja, o processo de consulta contínua executada no CEP.

Cada consulta realizada no CEP é executada por um Agente de Processamento de Eventos (*Event Processing Agent* - EPA). O EPA reage aos eventos recebidos, os analisa e

manipula continuamente gerando eventos complexos para os consumidores (Suhothayan et al., 2011). De acordo com (Paschke, 2008), a Tabela 2 sumariza um conjunto de padrões que correspondem a fundamentação das aplicações de eventos complexos.

Tabela 2 – Padrões de Eventos Complexos

Padrão	Descrição
<i>Filtering</i>	Avalia atributos de eventos utilizando uma condição lógica.
<i>In-Memory Caching</i>	Também conhecido como <i>window</i> (janela), armazena eventos em memória ambos recentes ou advindos de um banco de dados.
<i>Aggregation over Windows</i>	Usado para computar operações estatísticas em eventos na memória.
<i>Database Lookups</i>	Uso de dados históricos para enriquecer eventos de entrada.
<i>Database Writes</i>	Em determinadas situações, o armazenamento de eventos processados ou não, em banco de dados pode ser utilizado.
<i>Correlation (Joins)</i>	Correlaciona eventos advindos de diferentes streams.
<i>Event Pattern Matching</i>	Checa a ocorrência de eventos relacionados. Utiliza-se do tempo na composição de relacionamentos
<i>State Machines</i>	Usado nos casos em que comportamento e processos complexos devem ser modelados e rastreados. Eventos são usados para sinalizar transições de um estado para outro.
<i>Hierarchical Events</i>	Usado para criar eventos complexos que possuem uma relação hierárquica.
<i>Dynamic Queries</i>	Pode aparecer de 3 formas diferentes: - Registro de <i>queries</i> dinamicamente: Permite inserção de <i>queries</i> dinamicamente sem necessidade de reinicialização do servidor; - <i>Queries</i> de requisição/resposta: Realiza as análises em <i>streams</i> normalmente, mas só retorna os resultados quando explicitamente solicitado. - <i>Queries</i> de subscrição: Registra interesse em determinados eventos. Resposta é retornada para os subscritos

O Processamento de Eventos Complexos (CEP) que, ao contrário do processamento tradicional de fluxo, vai além da simples consulta e transformação de dados, visa detectar padrões de eventos complexos (Tab 2). Assim, do ponto de vista do CEP, as informações que chegam constantemente podem ser vistas como notificações de eventos que acontecem no mundo externo. O foco dessa perspectiva está na detecção de ocorrências de padrões específicos de eventos simples que desencadeiam novos eventos, os complexos. Uma consulta CEP busca resultados se, e somente se, um padrão correspondente de eventos for detectado. O CEP por si só não pode resolver o problema de heterogeneidade de várias fontes de dados, esta tarefa é delegada a uma implementação adicional do CEP, que permite a

representação e o formato comuns de dados como parte de seu modelo de programação. Ou seja, para poder manipular, por exemplo, temperatura e umidade do solo dentro de um único padrão CEP, é necessário definí-los como classes de programação (possivelmente como subclasses da mesma superclasse) com conjuntos de propriedades correspondentes. Quando as leituras correspondentes do sensor chegam ao sistema, elas são agrupadas nas classes CEP correspondentes e transmitidas para o mecanismo CEP implementado para processamento.

Essa implementação também é chamada de motor de inferência (*engine*), é onde os conceitos de CEP são instanciados. Esse motor de inferência fornece operações para definir o esquema, carga útil e primitiva dos eventos em tempo real para realizar as consultas contínuas e interconectá-las. Há várias implementações diferentes de motores de inferência CEP como, por exemplo, STREAM (Arasu et al., 2003), Sase+ (Diao; Immerman; Gyllstrom, 2007), Microsoft StreamInsight (Ali et al., 2011), Siddhi (Suhothayan et al., 2011), Apache Flink (Carbone et al., 2015) e Esper (EsperTech, 2019). As principais diferenças entre eles são os construtores e a semântica das linguagens usadas para definir os eventos e primitivas de processamento. Em geral, essas semânticas podem ser divididas em duas categorias: orientada a fluxo e orientada a regra. (M.Roriz et al., 2019)

A semântica orientada a fluxo de dados é baseada na linguagem de consulta contínua (*Continuous Query Language - CQL*), que é uma extensão da linguagem SQL (*Structured Query Language*) com operadores e primitivas específicas para tratamento e análise de fluxo de dados. Já a semântica orientada a regra é baseada em cláusulas de inferência Evento-Condição-Ação (ECA), onde a manipulação de eventos, verificação de condição e a execução de ação estão separadas em diferentes blocos. A ECA segue um fluxo de três passos, primeiro o agente de processamento é acionado quando o evento acontece, em seguida a condição é verificada e, por último, se a condição for atendida, a ação é executada. A principal diferença entre as duas semânticas é a implementação da sintaxe, no entanto, as duas seguem um mesmo conjunto de instruções chamadas primitivas de processamento.

2.3.1 Primitivas de Processamento

A principal função das primitivas CEP é converter o evento de entrada em um evento complexo usando os atributos do evento de entrada ou de uma fonte externa. (Akila; Govindasamy; Sandosh, 2016a). As primitivas podem ser divididas em primitivas de transformação e primitivas de detecção de padrão, como mostrado na Figura 3.

As primitivas de transformação são mais simples e fáceis de implementar. A filtragem retém ou deixa passar um atributo que compõe o evento. Já a divisão segmenta um ou mais atributos de acordo com a condição aplicada e possibilita transformar um evento em vários. Enquanto a composição combina atributos diferentes, advindos de uma mesma

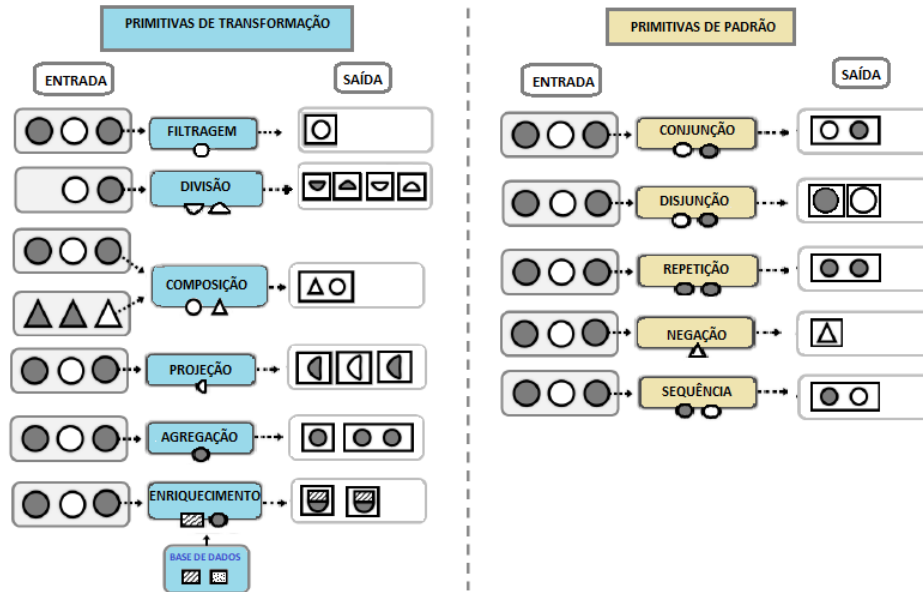


Figura 3 – Primitivas CEP

fonte ou de fontes diferentes, e compõe eventos de acordo com a necessidade da ação que será executada no próximo passo. A primitiva de projeção cria um evento complexo usando um subconjunto dos atributos contidos no evento anterior. A agregação permite combinar múltiplos eventos de entrada em um único de saída, empregando funções clássicas de agregação como média, mínimo e máximo. Por fim, o enriquecimento utiliza uma fonte de dados externa para criar um evento derivado contendo novas informações ou utilizando os dados para modificar os atributos de entrada.

Por outro lado, as primitivas de detecção de padrão são baseadas em operadores lógicos, como conjunção, disjunção, repetição, negação e sequência para definir como será feita a consulta contínua do CEP. A conjunção define quais os padrões dos eventos que ocorrem em conjunto (usando a lógica *AND*), analisando o fluxo de eventos de entrada a procura desses padrões. Por exemplo, considerando os eventos 'irrigar planta' e 'solo seco', a consulta detectará o padrão quando observar um ou mais eventos de 'irrigar planta' e pelo menos um evento 'solo seco' em um determinado intervalo de tempo. Semelhante a conjunção, a primitiva disjunção é satisfeita quando detecta pelo menos um dos eventos especificados (usando a lógica *OU*). A repetição detecta quando um determinado tipo de evento se repete pelo menos n vezes em um intervalo de tempo pré definido. A negação, por sua vez, define uma detecção de padrão quando há uma ausência de um determinado evento em um intervalo de tempo. Enfim, a primitiva de sequência captura o relacionamento entre eventos utilizando a ordem de acontecimento e o tipo deles.

As aplicações de CEP e suas primitivas usadas variam de acordo com a finalidade onde está sendo aplicada. O uso do processamento de eventos complexos é focado em áreas onde existe uma quantidade consideravelmente grande de dados, provenientes de diversas fontes, precisam ser processados. Alguns exemplos de domínios IoT que utilizam

CEP são: Cidades Inteligentes, *Smart Health*, Agricultura de precisão, dentre outros. E, dependendo do domínio da aplicação, diferentes arquiteturas CEP podem ser utilizadas, alguns exemplos dessas arquiteturas e suas utilizações estão na Tabela 3.

Tabela 3 – Exemplos de Arquiteturas CEP

Nome	Exemplo de uso
Cliente-Servidor	(Akbar et al., 2017)
Sistema SCEPter	(Ding; Gao; Ma, 2016) (Zhou; Simmhan; Prasanna, 2017)
CEP Proativo	(Wang; K.Cao, 2014)
Processamento de <i>stream</i> em tempo real para IoT	(Lee et al., 2019)
Processamento Ciente de Contexto para IoT	(Akbar et al., 2016)

Com base nesses conceitos, foi proposta a Arquitetura Hierárquica de Fusão de Dados em Névoa descrita no Capítulo 4.

3 Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos similares que abordam conceitos de computação em névoa, fusão de dados e processamento de eventos e suas aplicações na produção industrial, medicina e outros campos de pesquisa.

3.1 Metodologia da Revisão Sistemática

A revisão sistemática foi realizada com a finalidade de encontrar os trabalhos relacionados mais relevantes dentro do escopo da pesquisa e teve a seguinte estrutura:

- **String de Busca:**
 - ("data fusion"OR "Information fusion") AND (edge OR fog) AND (IOT OR COT)
- **Fontes de Dados:**
 - *ACM Digital Library*
 - *IEEE Digital Library*
 - *Science@Direct*
 - *Scopus*
- **Período:**
 - 2016 a 2020.
- **Critérios:**
 - Inclusão: Arquitetura de Fusão, Organização dos Nós;
 - Exclusão: *Survey*, Capítulo de Livro, documento não disponível.
- **Questões Norteadoras:**
 - Apresenta arquitetura de Fusão de Dados?
 - Utiliza Técnicas de Processamento Complexo de Eventos (CEP)?
 - Utiliza Técnicas de Gerenciamento de Nós?
- **Filtragem:**
 - Artigos obtidos por meio da *string* de busca: 106 (9 duplicados)
 - Artigos selecionados após aplicação dos critérios: 37
 - Artigos Selecionados seguindo as questões norteadoras: 13

3.2 Trabalhos Selecionados

Em (Lan et al., 2019) é proposto um mecanismo universal CEP de monitoramento IoT utilizando computação de borda. Além disso, é criado um modelo de evento complexo hierárquico formalizado, incluindo evento bruto, simples e complexo, para reduzir a

complexidade da modelagem de eventos. O modelo utiliza semânticas complexas de tempo e espaço para definir eventos flexíveis por meio de algoritmos. Já os autores em (Alturki; Reiff-Marganiec; Perera, 2017) utilizam uma abordagem híbrida, em que o processamento no nível da borda e no nível da nuvem trabalham em conjunto para criar uma análise de dados IoT eficaz. Especificamente, coletam-se dados brutos localmente e os recursos são extraídos aplicando técnicas de fusão de dados nos dispositivos com maior poder de processamento e então são enviados para a nuvem.

O trabalho de (Oliveira; Kniess, 2019) apresenta uma abordagem para redução de dados baseada na análise da dispersão dos dados de sensores, visando evitar o envio de detecções cujos valores sejam pouco dispersos. A abordagem proposta atingiu uma redução nas transmissões de dados, mantendo um baixo nível de erros e baixo consumo de energia. Em (Cremonesi et al., 2019) é proposto um sistema de três camadas hierárquicas (nuvem, névoa e dispositivos). O estudo foca na camada da névoa, composta por múltiplas redes capazes de estabelecer uma comunicação com a nuvem e armazenar identidades dos nós requisitadas pelos dispositivos. A camada da névoa pode possuir múltiplos níveis hierárquicos composta de diversos tipos de dispositivos com menor capacidade computacional se comparado com a nuvem (ex. pontos de acesso).

Em (Mazon-Olivo et al., 2018) é proposta uma arquitetura chamada “RECEP” para o processamento dinâmico de eventos gerados no contexto de IoT e Agricultura de Precisão (PA). O RECEP é composto por duas partes: o motor de regras (*Rules Engine* - RE) e o processador de eventos complexos (*Complex Event Processor* - CEP). O RE permite a configuração de regras dinâmicas condicionando dados de entrada de diferentes fontes e planejando ações de controle em atuadores, alertas e notificações para usuários finais ou aplicativos. O componente CEP fusiona os dados de entrada na taxa em que chegam, com as regras estabelecidas na RE e realiza uma análise prescritiva que consiste não apenas em prever ou detectar padrões de eventos, mas em tomar decisões automáticas. O RECEP foi implementado em uma máquina virtual com CPU de 1,9 GHz e 6 GB de RAM, depois foi integrado a um sistema de irrigação inteligente de uma bananeira experimental localizada em Machala-Ecuador. Um simulador de Redes de Sensores Sem Fio (RSSF) também foi utilizado para gerar dados do sensor em grandes quantidades, o CEP foi avaliado em vários casos de uso e os resultados mostram que ele consome recursos computacionais com tendência de crescimento, representado por um modelo de regressão logarítmica ($r\text{-quadrado} > 0,9$); ou seja, quanto mais eventos são processados, menos recursos são consumidos. Foi testado por quinze dias e cerca de 25 mil eventos por segundo foram processados.

O artigo de (Wei; Wu, 2019) propõe uma nova arquitetura de nuvem de sensor para IoT baseada em computação em névoa, que pré-processa dados brutos do sensor no nó de névoa, fornece armazenamento temporário dos resultados de pré-processamento,

se necessário, controla e gerencia diversos tipos de sensores em dispositivos de IoT por meio da virtualização de sensores e, por fim, fornece *Sensing-as-a-Service* dinâmico, sob demanda e padronizado para os usuários finais. Também projeta uma arquitetura com componentes de sensores baseados em névoa, fornece uma visão detalhada da função dos componentes e apresenta soluções preliminares para virtualização de sensores físicos, provisionamento dinâmico de grupo de sensores virtuais e instância de serviço.

O trabalho de (Dautov; Distefano; Buyya, 2019) aborda a fusão de dados hierárquica em redes IoT que contém dispositivos de borda, unidades de rede e comunicação e plataformas em nuvem. Diferentes fontes de dados são combinadas em cada nível para produzir resultados oportunos e precisos. Em (Vora et al., 2017) é proposto um sistema de monitoramento de pacientes baseado em computação em névoa. Os dados de movimento dos pacientes são coletados usando sensores vestíveis e são transmitidos usando os *gateways* da névoa. Para reduzir a carga na infraestrutura de comunicação, também é apresentado um algoritmo de clusterização para transmissão de dados.

Em (Kondo et al., 2019) foi desenvolvida uma plataforma de computação de borda que pode ser coordenada com cada dispositivo que a compõe, nó de névoa e nuvem. Os dados mais diversos provenientes dos nós de névoa e nuvem podem ser modularizados e ativamente implantados em cooperação com a rede de longa distância. Com esses recursos, é possível implantar e trabalhar os nós com flexibilidade e de acordo com as características dos dados e a finalidade de uso e fornecer várias funções em locais de rede apropriados. Esta plataforma traz também um ambiente de processamento transparente para dispositivos IoT de acordo com uma característica de dados ou finalidade de uso. Este artigo descreve um método de controle dinâmico para a plataforma com base em informações de vários sensores e mostra a sua implementação e avaliação.

O trabalho de (TORRES et al., 2020) propõe uma arquitetura de fusão de dados multinível para Internet das Coisas chamada Hydra, que visa melhorar a acurácia dos sensores, identificar eventos e possibilitar tomadas de decisões baseadas em regras definidas pelo usuário. A arquitetura Hydra é dividida em três níveis: baixo nível, com foco na fusão de dados dos sensores, incluindo a identificação e remoção de outliers; médio nível, que lida com eventos e decisões com base nos dados sensoreados e em regras definidas pelo usuário; e alto nível, que trata da fusão de decisões com base em múltiplas aplicações.

Em (R.Dautov; S.Distefano, 2017) é proposto um mecanismo que utiliza CEP e arquitetura hierárquica como método para habilitar a fusão de uma grande quantidade de dados em vários níveis. Outro trabalho que trata de grande quantidade de dados é (Zhang et al., 2016), onde é proposto um novo paradigma de computação, projetado para processamento de *big data* em ambiente de borda colaborativa. O trabalho propõe a fusão de dados geograficamente distribuídos, criando visualizações virtuais compartilhadas dos dados que são expostos aos usuários finais por meio de interfaces predefinidas pelos

proprietários dos dados.

O artigo de (Merkel; Cocos, 2019) propõe o uso de computação em névoa como um complemento às redes de sensores. Com o uso de nós *edge*, algumas das tarefas de computação intensiva são executadas próximas da rede de sensores, resultando em um pré-processamento de dados. Além disso, utiliza o mecanismo de processamento de eventos complexos *Siddhi* em um nó *edge* inserido em uma rede de sensores sem fio, que mede várias grandezas físicas para determinar a qualidade do ar interno.

De todos os trabalhos citados, o único que possui um modelo de evento com mecanismo de descrição hierárquico específico é o apresentado em (Lan et al., 2019). Porém, as camadas consideradas em (Lan et al., 2019) utilizam apenas uma modelagem CEP pura, sem utilizá-lo como ferramenta de fusão de dados, o que torna sua utilização mais complexa e dessa forma, não é capaz de atender a aplicações voltadas a eventos mais abrangentes de monitoramento. A proposta de (TORRES et al., 2020), é similar quanto a divisão dos níveis da arquitetura, porém não é aplicada em ambientes de névoa nem possui um mecanismo CEP distribuído entre as camadas. Outro ponto importante que difere nosso trabalho de (Lan et al., 2019) e (TORRES et al., 2020), bem como de todos os outros citados, é a capacidade de suportar uma lógica de gerenciamento flexível (autoconfiguração dos nós) que será descrita na Seção 4.4.

As principais contribuições e diferenças entre os trabalhos estão listadas na Tabela 4.

Tabela 4 – Tabela Comparativa dos Trabalhos Relacionados

Trabalho	Estrutura	CEP	Autoconfiguração
<i>Esta proposta</i>	<i>Hierárquica Fusão + Eventos (3 camadas)</i>	<i>Sim</i>	<i>Sim</i>
(Lan et al., 2019)	Hierárquica de Eventos (3 camadas)	Sim	Não
(Alturki; Reiff-Marganiec Perera, 2017)	Fusão Híbrida (2 camadas)	Não	Não
(Oliveira; Kniess, 2019)	Agregação de Dados	Não	Não
(Cremonezi et al., 2019)	Hierárquica (3 camadas)	Não	Não
(Dautov; Distefano; Buyya, 2019)	Hierárquica (3 camadas)	Não	Não
(Vora et al., 2017)	Fusão + Clusterização	Não	Não
(R.Dautov; S.Distefano, 2017)	Hierárquica Multinível	Sim	Não
(Zhang et al., 2016)	Colaboração de Névoas	Sim	Não
(Merkel; Cocos, 2019)	CEP na Névoa	Sim	Não
(Wei; Wu, 2019)	3 camadas	Não	Não
(Mazon-Olivo et al., 2018)	CEP (2 camadas)	Sim	Não
(Kondo et al., 2019)	Multi nodal colaborativo	Não	Não
(TORRES et al., 2020)	Hierárquica de Fusão (3 camadas)	Não	Não

4 Uma Arquitetura Hierárquica de Fusão de Dados em Névoa

Neste capítulo serão definidos os componentes da arquitetura hierárquica proposta neste trabalho. Além disso, serão definidas também as regras e eventos que fazem parte do modelo CEP juntamente com a lógica de autoconfiguração dos nós sensores.

4.1 Definição das Camadas da Arquitetura

Na arquitetura proposta são assumidos os diferentes níveis do ambiente da computação em névoa (sensor, névoa e nuvem) descritos na Seção 2.1. Os sistemas de IoT tendem a ir além da noção de monitoramento "de baixo para cima", ou seja, não somente os dados brutos são coletados pelos dispositivos de sensoriamento e transferidos pela rede, mas também implementam a comunicação de *feedback* "descendente" entre os dispositivos gerenciados e gerenciador (por exemplo, para tomadas de decisão e/ou atuação baseada nos dados). Como parte dessa comunicação bidirecional, a abordagem proposta utiliza a noção de coordenação, responsável pela comunicação entre as camadas descritas. Mais especificamente, a coordenação é uma funcionalidade em duas direções.

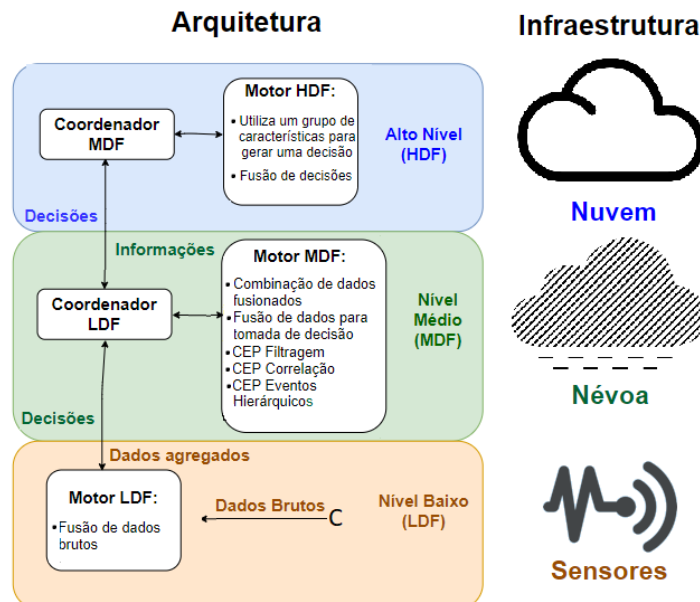


Figura 4 – Arquitetura conceitual da fusão de dados hierárquicos em três níveis.

A arquitetura proposta é composta por três níveis, como ilustrado na Figura 4. O nível baixo consiste em uma infraestrutura de sensores que faz a coleta dos dados brutos. A partir desses dados, o motor LDF (*Low Data Fusion* ou Fusão de Baixo Nível), por fazer

parte do nível com menor poder de processamento, realiza uma fusão de dados baseada em 3 eventos simples (descritos na Seção 4.2). Após o processamento, o motor LDF envia um conjunto de dados agregado, resultante dos eventos simples processados, ao coordenador LDF. Tendo como base as informações recebidas, o motor MDF (*Medium Data Fusion* ou Fusão de Nível Médio) realiza a fusão dos dados de acordo com as regras dos 7 eventos complexos (descritos na seção 4.2) que são criados a partir do processamento dos eventos simples criados na camada do sensor. Esse processamento é realizado pelo Agente de processamento de eventos (EPA, do inglês *Event Processing Agent*), que está inserido no motor MDF. O EPA atua de acordo com as regras definidas em cada evento se baseando nas primitivas de filtragem, correlação e eventos hierárquicos. Como o nível médio da arquitetura não possui uma visão global do sistema, para realizar esse processamento ele acessa informações contidas na camada mais alta da arquitetura hierárquica. Se mesmo após esse acesso a névoa não for capaz de realizar o processamento, o coordenador MDF repassa as informações para a camada acima. A camada da nuvem, composta por uma infraestrutura com o maior poder de processamento da arquitetura, utiliza o grupo de características geradas pela névoa e realiza o processamento e gera uma decisão de acordo com a necessidade do sistema.

Por um lado, o dispositivo da névoa recebe dados de dispositivos de nível inferior, coletando e despachando para processadores e motores de fusão de dados, bem como para o nível superior, se necessário. Por outro lado, esse mesmo dispositivo carrega novas regras de processamento existentes, de acordo com as mudanças observadas nos dados. Como resultado, a arquitetura conceitual de fusão de dados em três níveis aqui proposta é realizada por meio da implantação e execução de instâncias da lógica de fusão de dados em dispositivos, constituindo sistemas de IoT de vários níveis. A separação **conceitual** de tarefas em todos os três níveis é ilustrada na Figura 4. Essas tarefas possuem instâncias de mecanismos de fusão de dados (DF) dedicadas, ou seja, fusão de dados de baixo nível (LDF), de nível médio (MDF) e de alto nível (HDF) (Dautov; Distefano; Buyya, 2019). Os dois níveis superiores incluem componentes de coordenação (coordenadores LDF e MDF), responsáveis pela comunicação bidirecional entre os nós de nível inferior e superior e o gerenciamento de solicitações recebidas dos nós de nível inferior.

Sendo assim, como ilustrado na Figura 5, podemos dividir as tarefas de cada camada da seguinte maneira:

- nos nós sensores ocorre a coleta dos dados e uma fusão simples baseadas em regras e eventos simples;
- na camada da névoa ocorre a produção dos eventos complexos baseados nos dados e eventos desencadeados pela camada anterior. Além disso, é realizado a fusão de informações que pode desencadear uma atuação ou no envio das informações para processamento na camada acima.

- na camada da nuvem ocorre a fusão mais robusta e a geração de informação mais refinada a partir de uma possível fusão de decisões que resultam no suporte para a tomada de decisão humana ou automatizada do sistema.

Na Figura 5 é ilustrado o processo inteiro, desde a geração dos dados que dão início aos eventos até a informação produzida ao final do processamento. Os dados simples são gerados e é feita a primeira fusão de dados redundantes pelos dispositivos sensores. Já no nó da névoa, ocorre a fusão de dados quantitativos de natureza complementar, onde são agregados os valores fusionados de diferentes nós sensores e os eventos simples se agregam formando eventos mais complexos. Os resultados dessa fusão criam novos eventos que podem resultar em uma tomada de decisão ou na geração de uma informação a ser processada no nó da nuvem. Na nuvem, a informação pode ser consumida por uma aplicação, armazenada ou processada utilizando recursos não disponíveis nas camadas com menor poder de processamento, dependendo do evento que será desencadeado. Vale ressaltar que o processamento de vários eventos simples nada mais é que o processamento complexo de eventos. Brevemente, os eventos processados pelos nós nas três camadas são:

1. Um evento simples é extraído a partir do resultado da fusão simples dos dados sensoreados, a agregação é feita a partir da realização de um número N de leituras realizadas pelo nó sensor;
2. O resultado da filtragem de eventos irrelevantes também gera um evento, por exemplo, a temperatura não mudou desde a última medição enviada, logo esse valor é ignorado, esse processo é realizado no nó névoa;
3. O conjunto dos eventos simples é processado gerando eventos mais complexos, e esse processamento é feito no nó névoa;
4. De acordo com a complexidade do evento é gerado um novo evento para continuação do processamento na camada com maior poder de computação. Por exemplo, um cálculo que precisa de uma visão global do sistema terá que ser realizado no nó nuvem, uma fusão que precisa apenas dos dados dos sensores é realizada no nó névoa;

No ambiente da Internet das coisas, os dados brutos são frequentemente gerados por dispositivos em modo massivo. Para extrair informações significativas, precisamos executar algum trabalho de processamento parcial nos dados brutos. Embora as técnicas de fusão de dados tradicionais também tenham capacidade de processar e extrair informações dos dados, a combinação com outras técnicas de processamento melhora ainda mais sua eficiência. Para trabalhar em conjunto com a fusão de dados, nossa proposta foi adaptada do modelo CEP de regras apresentado em (Akila; Govindasamy; Sandosh, 2016b) e respeita os seguintes princípios:

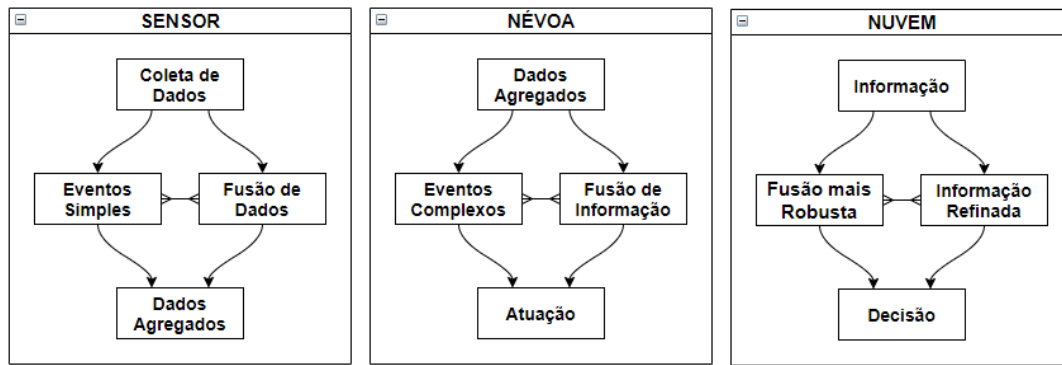


Figura 5 – Etapas de Processamento de Eventos

- Os eventos simples são gerados a partir da coleta dos dados brutos na camada dos sensores;
- Todos os eventos complexos da névoa são desencadeados a partir dos eventos simples realizados pela camada dos sensores.

Com o uso da arquitetura de fusão em conjunto com CEP, podemos separar facilmente o processo em parte de processamento parcial e parte tomada de decisão. Ao fazer isso, não apenas o processamento, mas também o latência da rede serão reduzidos. Por exemplo, ao detectar uma temperatura "anormal" o sensor envia apenas o dado para a parte de tomada de decisão que por sua vez irá lidar com tarefas de raciocínio semântico, como correlacionar os dados recebidos. De modo geral, a técnica CEP utilizada pode ser vista como um sistema que recebe e correlaciona um conjunto de eventos simples gerando eventos complexos. Ou seja, está relacionado a detecção de eventos simples, utilizando técnicas de filtragem, correlação, contextualização e análise de dados advindos de diversas fontes, em um ciclo, como mostra a figura 6.



Figura 6 – Ciclo de Processamento de Eventos

Além das regras citadas acima, o sistema proposto utiliza um mecanismo CEP que retransmite os dados fusionados para o processamento de eventos complexos feito na camada da névoa de acordo com o seguinte fluxo: (i) o nó sensor inicia um evento de fusão

simples de dados e envia seu resultado para a camada da névoa; (ii) na névoa existe um mecanismo de regras que descreve os eventos gerados a partir do valor recebido, ou seja, o resultado da média calculada na camada anterior vai influenciar diretamente em qual evento será executado nessa camada; (iii) a névoa processa o evento de acordo com o que foi descrito na regra; (iv) caso o evento necessite de uma visão global do sistema, um novo evento envia os dados para serem processado na camada superior. Segundo (Benarbia; Alaoui; Bennani, 2018), esses 4 mecanismos de fluxo podem ser chamados, respectivamente, de receptor de dados, mecanismo de regra, executor de eventos e encaminhador de evento. Esse mecanismo é gerenciado de acordo com as regras definidas nas seção a seguir.

4.2 Definição das Regras e Eventos

As regras que gerenciam os eventos são divididas em dois conjuntos: **Regra de Monitoramento** e **Regra de Estado Crítico**. Cada regra será composta por um conjunto de eventos que serão executados de acordo com as suas primitivas de padrão e transformação. Esses eventos serão alocados nas camadas de acordo com a capacidade dos dispositivos para não afetar a performance do sistema e influenciará diretamente na tomada de decisão, tanto pelo usuário quanto de forma automática de acordo com o comportamento do sistema.

4.2.1 Regra de Monitoramento

Os eventos da regra de monitoramento são baseados nas primitivas de **filtragem**, **agregação**, **projeção**, **agregação**, **enriquecimento** e **composição**. A projeção se encarrega de criar eventos complexos (que serão processados na camada da névoa) usando um subconjunto de seus atributos, gerados pela agregação (media de valores obtidos pelos sensores) e pela filtragem (obtenção dos valores de máximo e mínimo). O enriquecimento realiza a consulta dos dados que não são coletados pelos sensores que compõem a rede, mas são necessários para gerar uma informação. A composição utiliza os dados adquiridos através da primitiva de enriquecimento e realiza cálculos para a formação dessa nova informação.

4.2.2 Regra de Estado Crítico

A regra de estado crítico é composta por eventos baseados nas primitivas de **enriquecimento**, **composição**, **negação** e **sequência** que serão executadas na camada da **névoa**. Para a utilização da primitiva de sequência é necessário um controle de entrada que pode ser feito por meio de janela de tempo, utilizando um **timestamp**, ou por contexto, utilizando um identificador de evento.

4.3 Definição da Arquitetura CEP

O modelo de processamento de Eventos Complexos (CEP) proposto foi utilizado como parte do mecanismo de fusão de dados visando tratar, analisar e reagir aos fluxos de dados encapsulados como eventos. Os fluxos dos eventos são as principais fontes de entrada do modelo CEP, os eventos são criados por meio dos **produtores** (sensores, usuários) que fornecem os dados. O fluxo dos dados é executado por meio do agente de processamento de eventos do CEP, conhecido como EPA. Especificamente o EPA segue o seguinte fluxo: reage aos eventos, analisa e os manipula de acordo com as regras definidas e, se necessário, gera eventos derivados (complexos) para os consumidores do evento. O EPA age de acordo com as primitivas definidas em cada exemplo como ilustrado na Figuras 7 e 8.

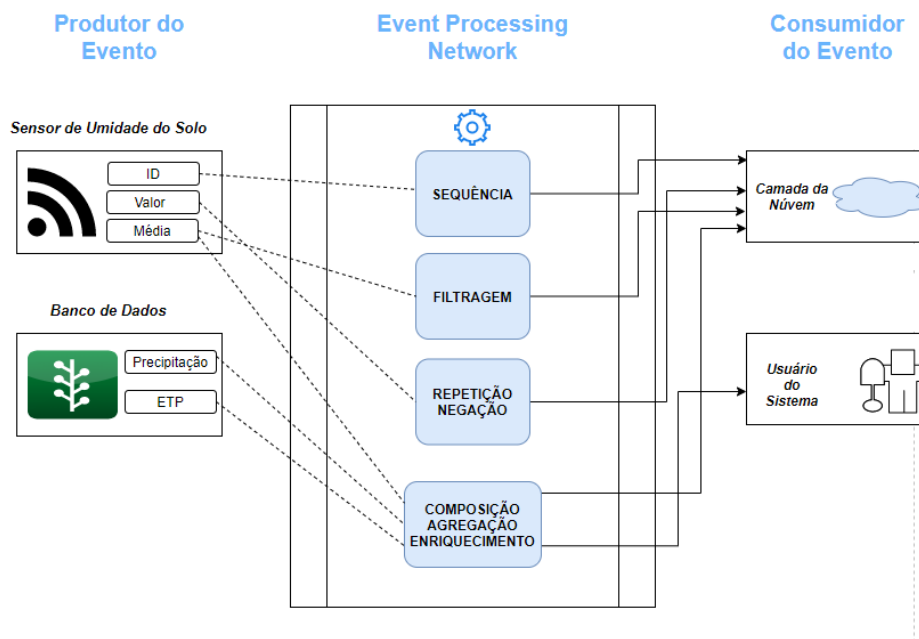


Figura 7 – Arquitetura e Primitivas CEP

Um motor de inferência CEP permite instanciar os conceitos apresentados. Ele fornece operações para definir os tipos de eventos (esquema e carga útil) e primitivas em tempo real para expressar as consultas contínuas (EPAs) e para interconectá-las.

Dessa forma, podemos citar como vantagens da arquitetura proposta de fusão com uso do CEP em três camadas:

- Os mecanismos CEP distribuídos na rede são capazes de equilibrar a carga de trabalho dos níveis com menor poder de processamento, gerando eventos complexos e processando-os nas camadas com maior poder de processamento;
- Um novo mecanismo CEP pode ser implantado em caso de aumento do volume de dados, alterando-se apenas o nível da hierarquia em questão;
- O tráfego de rede para transferir informações entre níveis pode ser reduzido significativamente, enviando apenas os dados necessários para o processamento

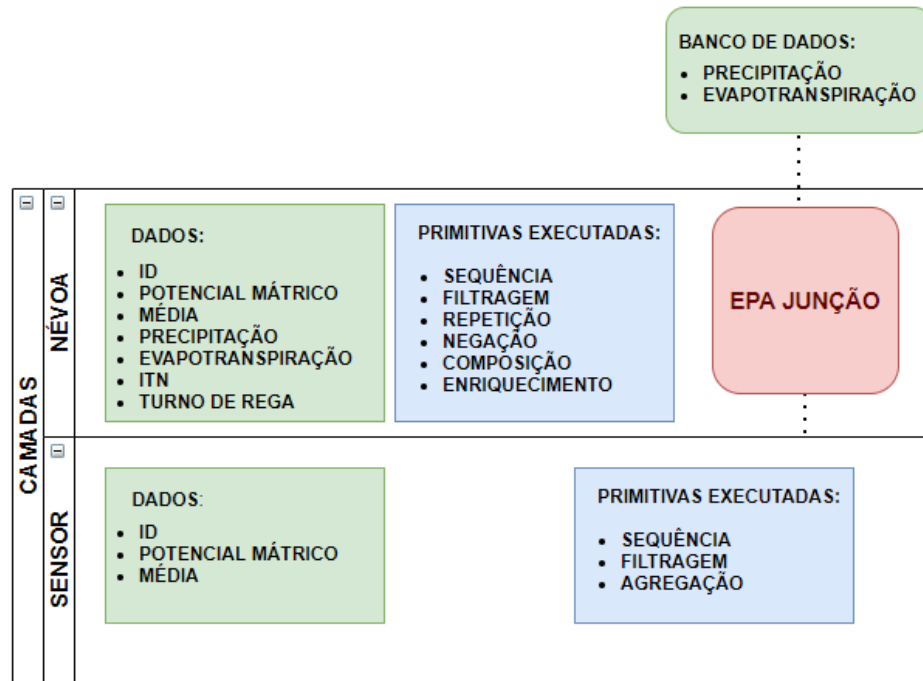


Figura 8 – Primitiva CEP de acordo com as camadas

dos eventos nos níveis apropriados.

Além disso, uma das principais contribuições dessa proposta é que a arquitetura hierárquica suporta uma lógica de gerenciamento flexível, realizada pela camada de nós névoa, onde os nós podem ser adicionados e configurados rapidamente para se adaptar às alterações no ambiente monitorado, que será descrita na seção 4.4.

4.4 Definição da Autoconfiguração dos Nós Sensores

A arquitetura proposta nesta dissertação também é composta por uma lógica de autoconfiguração dos nós processada pelos nós da névoa e decidida pela nuvem. Desse modo, quando um novo nó sensor é adicionado à rede, ele consegue ser conectado a uma névoa de maneira simples. Para isso, ele envia via *broadcast* para os nós névoa uma mensagem incluindo suas coordenadas (que podem ser entendidas como latitude e longitude) e o tipo de dado coletado. Se o nó da névoa for do tipo compatível, ele calcula a distância euclidiana entre eles e envia esse dado para o nó da nuvem. Essa compatibilidade é relacionada ao tipo de dado que a névoa está apta a processar. Por exemplo, um sensor que envia dados de temperatura, por exemplo, só vai poder se conectar a um nó de névoa que processa dados de temperatura. O nó nuvem verifica qual névoa possui o menor valor de distância calculado e retorna essa informação para o nó da névoa correspondente que, por sua vez, passa a informação ao nó sensor, como ilustrado na figura 9.

Ao fim dessa configuração, o nó sensor passará a enviar os dados coletados apenas

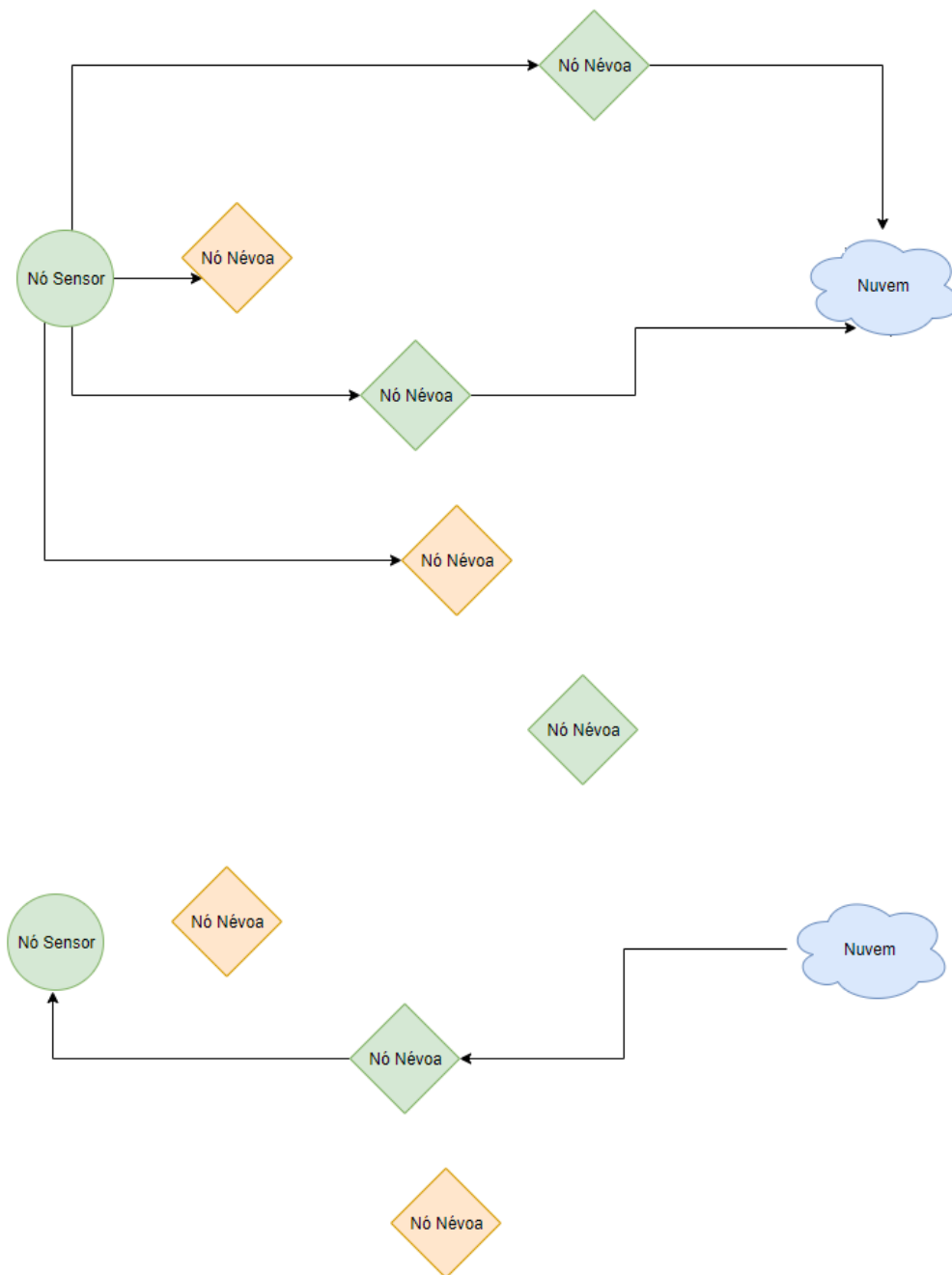


Figura 9 – Gerência de um Nó Adicionado a Rede

para a névoa correspondente para que seus dados sejam processados de acordo com a necessidade do sistema.

Para uma melhor compreensão das regras definidas, a arquitetura foi aplicada em um caso de estudo no domínio da agricultura de precisão e será apresentado no Capítulo 5.

5 Estudo de Caso: Agricultura de Precisão

No setor agrícola, alcançar o máximo rendimento da colheita a um custo mínimo é uma meta da produção. Assim, a tomada de decisão é complexa, pois há vários fatores que afetam todo o processo, visando a economia de recursos hídricos gastos principalmente com a irrigação.

O estudo da água no solo é de grande interesse para a agricultura porque influencia no desenvolvimento e produção das culturas. Uma planta em desenvolvimento deve ser capaz de suprir a demanda de água da atmosfera através da quantidade que ela pode extrair do solo. O armazenamento de água no solo é diminuído pela evapotranspiração e recomposto pela precipitação ou irrigação. A importância do armazenamento e da disponibilidade de água do solo na produção das culturas é reconhecida e muitas pesquisas são realizadas para caracterizar as propriedades do solo responsáveis pela absorção e retenção de água (Frizzone, 2017).

O solo é formado por materiais sólidos (minerais e orgânicos) e poros preenchidos por água e ar (vapor d'água, O_2 , CO_2 , N_2 , NH_3 , etc.). Logo, o solo é considerado um reservatório de água e de nutrientes de um cultivo. O solo, a água e o cultivo podem ser tratados como um sistema dinâmico contínuo, onde a água movimenta-se pelo solo até as raízes das plantas, passando por dentro delas e saindo para a atmosfera. Deste modo, a água modifica a energia potencial com que está retida no solo (Silva, 2019).

A irrigação faz com que o solo atinja sua Condição de Campo (CC) na qual reserva-se água o suficiente para a cultura em seu estágio de desenvolvimento, isso pode ser monitorado utilizando o potencial mátrico. O potencial mátrico (ψ) refere-se a interação entre a matriz do solo (material mineral e sólido) e a água, incluindo forças associadas à absorção e capilaridade das plantas, resultando na retenção de água no solo (Frizzone, 2017).

Existem três tipos de métodos usados no manejo da irrigação para determinar quando irrigar, a quantidade de água e o tipo de irrigação: (1) Turno de Rega Calculado, (2) Balanço Hídrico e (3) Potencial Mátrico. O turno de Rega (TR) é a frequência das irrigações, dado em dias e é calculado previamente para cada estágio de desenvolvimento da planta. O método do Balanço Hídrico é calculado apoiando-se no turno de rega e é utilizado quando o sistema de irrigação recebe água em dias fixos e é realizada a distribuição de água em rotação. Há também a forma de distribuição de água por demanda, ou seja, quando a água está sempre disponível, ou pelo projeto de irrigação ou bombeada pelo próprio usuário, sendo utilizada quando houver necessidade do cultivo. Nesse tipo de distribuição é utilizado o método baseado no Potencial Mátrico.

O método do Potencial Mátrico determina o momento de irrigar monitorando o potencial mátrico (ψ) do solo da camada onde existe a maior concentração de raízes do cultivo (geralmente na camada de solo mais superficial monitorada por tensiômetro). Vale ressaltar que o sistema de irrigação deve aumentar a umidade θ até a umidade da capacidade de campo θ_{cc} do solo para que o cultivo extraia a água pela sua zona radicular durante o processo de evapotranspiração (Frizzzone, 2017). Outra forma de determinar o momento de irrigar é através da umidade crítica θ_{cr} , também denominada umidade ideal para irrigação, que consiste no teor de água do solo a partir do qual o rendimento da cultura começa a ser reduzido havendo a possibilidade de diminuição da evapotranspiração (consumo de água pelo cultivo). θ_{cr} é necessária apenas para a camada de solo mais superficial que é monitorada por tensiômetro. A evapotranspiração da cultura (Et_0) é o suprimento de água necessário para os processos fisiológicos da planta em um determinado clima (Silva, 2019) e ela depende do monitoramento de variáveis climáticas e do cultivo. Outra varável importante a ser considerada tendo em vista a necessidade da planta é a Irrigação Total Necessária (ITN). Segundo (Frizzzone, 2017), a ITN consiste na quantidade de água provida totalmente pela irrigação para compensar a perda de água devido a evaporação. Ela é obtida através de um cálculo considerando a precipitação, a eficiência do sistema de irrigação do campo e o deficit de água no solo.

Baseados nesses conceitos, foram definidas regras e eventos do caso de estudo que serão apresentadas nas seções a seguir.

5.1 Estudo de Caso: Regras de Monitoramento

Os três primeiros eventos da regra de monitoramento são baseados nas primitivas de **filtragem**, **agregação** e **projeção** que são realizados no nível do **sensor**. A regra desse primeiro conjunto é composta pelos seguintes eventos:

- **EV1 (SIMPLES)**: Alerta de Potencial Mátrico baixo;
- **EV2 (SIMPLES)**: Alerta de Potencial Mátrico alto;
- **EV3 (SIMPLES)**: Alerta de envio de Potencial Mátrico diário + ID do sensor.

As saídas desses eventos serão **mensagens enviadas para a camada da névoa**.

O segundo conjunto de eventos da regra de monitoramento é composto pelas primitivas de **agregação**, **enriquecimento** e **composição** e é executado na camada da névoa.

- **EV4 (SIMPLES)**: Monitoramento de precipitação diária;
- **EV5 (SIMPLES)**: Monitoramento de Evapotranspiração;
- **EV6 (COMPLEXO)**: Monitoramento da variação do Potencial Mátrico;
- **EV7 (COMPLEXO)**: Monitoramento da ITN diária;

- **EV8 (COMPLEXO):** Verificação do turno de rega diário;

As saídas desses eventos serão **mensagens exibidas em um visor e enviadas ao usuário**.

5.2 Estudo de Caso: Regras de Estado Crítico

Esse conjunto de regras é baseado nas primitivas de **enriquecimento**, **composição**, **negação** e **sequência** e é composto pelos seguintes eventos:

- **EV9 (COMPLEXO):** Umidade crítica (ideal para irrigação);
- **EV10 (COMPLEXO):** Deficit máximo de água.

Esses eventos resultarão em atuações automáticas na área monitorada.

5.3 Estudo de Caso: Definição da Arquitetura CEP

Os motores de inferência dos eventos, aplicados a agricultura de precisão, são mostrados a seguir.

EV1: Monitoramento Potencial Mátrico (ψ) Baixo

R1: Potencial Mátrico ≤ 55 kPa

- Enviar o valor do ψ , ID e zona do sensor para a camada da névoa
- Acionar EV6

EV2: Monitoramento Potencial Mátrico (ψ) Alto

R2: Potencial Mátrico ≥ 100 kPa

- Enviar o valor do ψ , ID e zona do sensor para a camada da névoa
- Acionar EV6

EV3: Monitoramento Potencial Mátrico (ψ) Diário

R3: Número de leituras (n) = 12

- Calcular a média do ψ
- Enviar o valor do ψ médio e ID do sensor para a camada da névoa
- Acionar EV4

EV4: Monitoramento de Precipitação Diária (p)

R4: Após EV3

- Consultar valor de precipitação diária no site INMET (ou outro Banco de Dados)
- Armazenar na variável p
- Acionar EV5

EV5: Monitoramento de Evapotranspiração (E_t)**R5: Após EV4**

- Consultar valor de Evapotranspiração no Banco de Dados
- Armazenar na variável E_t
- Acionar EV7

EV6: Monitoramento da Variação do Potencial Mátrico (ψ)**R6: Após EV1 ou EV2****R6.1: Se EV1**

- Consultar ψ de sensores com IDs diferentes do que gerou o evento na mesma zona

R6.1.1: Se $60 \text{ kPa} < \psi \leq 100 \text{ kPa}$

- Enviar mensagem ao usuário para verificar possível mal funcionamento do sensor

R6.1.2: Se $\psi \leq 55 \text{ kPa}$

- Acionar EV10

R6.2: Se EV2

- Consultar (ψ) de sensores com IDs diferentes do que gerou o evento na mesma zona

R6.2.1: Se $60 \text{ kPa} < \psi \leq 100 \text{ kPa}$

- Enviar mensagem ao usuário para verificar possível mal funcionamento do sensor

R6.2.2: Se $\psi \geq 100 \text{ kPa}$

- Verificar Precipitação

R6.2.2.1: Se $p = \text{NULL}$

- Enviar mensagem ao usuário para verificar vazamento de água

EV7: Monitoramento ITN**R7: Após EV5**

- Calcular ITN
- Enviar mensagem com o valor ao usuário
- Acionar EV8

EV8: Definição do Turno de Rega**R8: Após EV7**

- Calcular e programar o tempo de rega
- Enviar mensagem com o tempo ao usuário

EV9: Umidade Crítica

R9: $ITN \leq$ Umidade Critica (60kPa)

- Acionar irrigação da zona
- Enviar mensagem ao usuário

EV10: Deficit máximo de água

R10: Apos EV6 (Se R6.1.2)

- Acionar irrigação da zona
- Enviar mensagem com o tempo ao usuário

Afim de verificar o comportamento da arquitetura proposta, o capítulo 6 mostra uma prova de conceito no ambiente de agricultura de precisão através da realização de experimentos em diferentes cenários.

6 Metodologia

Neste capítulo é apresentada a metodologia, descrevendo os experimentos realizados como prova de conceito da arquitetura proposta neste trabalho e a validação do modelo CEP usado como mecanismo para a fusão de dados hierárquica.

6.1 Prova de Conceito: Arquitetura da Simulação e seus componentes

Para esse experimento foi utilizado um simulador de eventos discretos focado, mas não restrito, a ambientes de névoa chamado YAFS (*Yet Another Fog Simulator*). O YAFS foi projetado para analisar o *design* de aplicações e incorpora estratégias para posicionamento, programação e roteamento. O YAFS possui características similares as do simulador iFogSim e o utiliza como referência (Lera; Guerrero; Juiz, 2019). De acordo com (Gupta et al., 2017), o iFogSim é o simulador de nevoeiro mais utilizado para simulações de cenário utilizando computação em névoa e em nuvem, sendo ele uma extensão do conhecido simulador CloudSim (Calheiros et al., 2011). O YAFS inclui mais funcionalidades do que os simuladores atuais para modelar cenários de IoT e apresenta uma maior facilidade de adicionar extensões.

Para a prova de conceito da proposta, o ambiente de agricultura de precisão foi utilizado como estudo de caso. Os campos agrícolas simulados incluem nós sensores que monitoram dados de temperatura e umidade do solo de uma plantação de coqueiros. Cada nó sensor está conectado e envia seus dados para um nó névoa localizado no campo, enquanto cada nó da névoa está conectado, por sua vez, a um nó da nuvem. Especificamente, foram executadas simulações em três cenários de agricultura de precisão considerando (i) uma rede com topologia fixa e de tamanho relativamente pequeno, (ii) uma rede com uma quantidade média de nós e (iii) uma rede em que a quantidade de nós vai aumentando no decorrer do tempo, ou seja, a topologia é dinâmica. Para cada cenário, a quantidade específica de cada tipo de nó é descrita a seguir:

1. O Cenário I (Fig. 10) é composto por 25 nós sensores, 5 nós névoa e um nó nuvem. No experimento utilizando este cenário foram executadas 30 rodadas de simulação.
2. O Cenário II (Fig. 10) é formado por 100 nós sensores, 10 nós de névoa e uma nuvem. No experimento utilizando este cenário também foram executadas 30 rodadas de simulação.

3. No Cenário III (Fig. 11), a rede é composta inicialmente por 16 nós sensores, 4 nós névoa e 1 nó nuvem. A cada nó névoa estão conectados 4 nós sensores, e os 4 nós névoa estão conectados ao nó nuvem. A cada rodada da simulação são adicionados nós sensores e nós névoa, de acordo com a Equação 6.1, sem alteração na quantidade de nó nuvem, que permanece 1 (um).

$$NS = NN^2 \quad (6.1)$$

onde:

$$3 < NN < 33$$

NS é o número de nós sensores;

NN é o número de nós névoas.

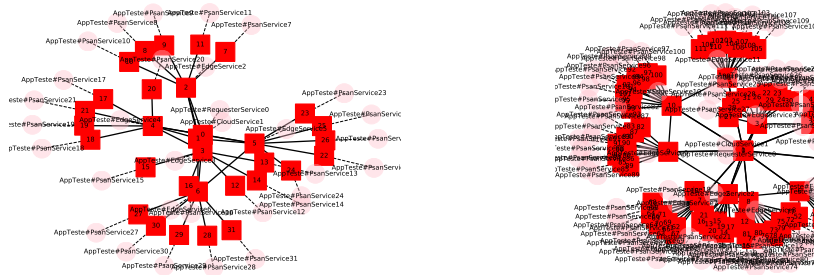


Figura 10 – Topologia da Rede: Cenário 1 e 2

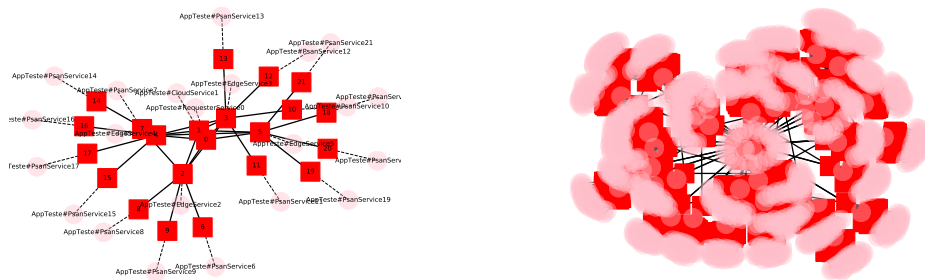


Figura 11 – Topologia da Rede 3: Situação Inicial e Final da Simulação

A Figura 11 mostra a plotagem da topologia da rede no cenário 3 e suas conexões no início e fim da simulação respectivamente. A rede se autoconfigura e conecta cada nó sensor ao nó névoa baseado no dado coletado e na distância euclidiana entre os nós (descrito na Seção 4).

Os nós sensores possuem 16 MB de RAM, enquanto os da névoa possuem 4 GB e os da nuvem possuem 8 GB. Esses valores foram escolhidos de forma a ficar mais próximos dos valores dos dispositivos usados no mundo real. Os parâmetros configurados nos experimentos realizados são resumidos na Tabela 5.

Tabela 5 – Experimentos

Experimento	Cenário	Avaliação
1	I: 25NS, 5NN, 1 Nuvem	vazão e tempo de envio em densidade baixa de nós;
2	II: 100NS, 10NN, 1 Nuvem	vazão e tempo de envio em densidade alta de nós;
3	III: $NS = NN^2$, 1 Nuvem	vazão e tempo de envio em cenário dinâmico.

6.2 Validação e Auditoria do Modelo de Processamento de Eventos Complexos

Segundo (Etzion; Niblett, 2011) existem duas ferramentas para validação de um modelo de técnicas de processamento de eventos complexos: Análise Estática e Análise Dinâmica da rede de processamento. Em suma, essas análises fornecem observações sobre a rede de processamento de eventos que apontam possíveis problemas na aplicação desses eventos. A auditoria é a capacidade de investigar se os processos foram aplicados de maneira adequada. Isso pode se referir a se um processo está em conformidade com regulamentos externos ou políticas internas, ou se uma decisão, tomada a partir desse processo, é apropriada. Enquanto a validação tem como objetivo validar a implementação, a auditoria tem como objetivo validar o uso do sistema e os processos por trás de um determinado sistema de informação.

Tendo como base essas definições, a validação foi realizada por meio de uma análise estática da rede de processamento tendo como base o algoritmo utilizado para a criação das regras e eventos do modelo CEP proposto. Para a auditoria, foi feita a consulta do *log* resultante do processamento de cada evento, seus antecedentes e suas consequências, com o intuito de verificar se os eventos são acionados corretamente de acordo com as necessidades do sistema.

No capítulo a seguir (Cap 7) será feita uma análise baseada nos experimentos realizados e descritos acima.

7 Avaliação Experimental: Resultados e Análise

Neste capítulo, apresenta-se uma avaliação de desempenho dos experimentos realizados e divididos em duas partes. A primeira, refere-se aos experimentos executados para avaliação do impacto da proposta no ambiente de névoa usando as métricas vazão e tempo de envio dos pacotes. A segunda parte se refere a validação e auditoria do CEP como ferramenta de processamento de dados e tomada de decisão.

7.1 Experimento 1: Métricas de Desempenho

Avaliação e resultado de experimentos em relação as seguintes métricas: vazão e tempo de envio dos pacotes transmitidos entre os nós sensores, a névoa e a nuvem.

Nos cenários 1 e 2 (Fig. 10), a topologia da rede é fixa, pois estes cenários foram gerados com o intuito de ilustrar o funcionamento da proposta em diferentes densidades. O cálculo da vazão (Fig. 12) foi realizado por meio de uma média aritmética da quantidade de mensagens enviadas em cada rodada de simulação. Além disso, o gráfico apresenta o comportamento dos cenários 1 (C1) e 2 (C2) retirando-se a proposta do nível da névoa para fins de comparação. Os resultados apresentados na Figura 12 mostram a diferença da quantidade de mensagens enviadas com o processamento de dados. Devido a fusão de dados ser executada nos três níveis da rede, existe uma diminuição na quantidade de dados enviados entre os níveis, mesmo com esse aumento considerável na densidade.

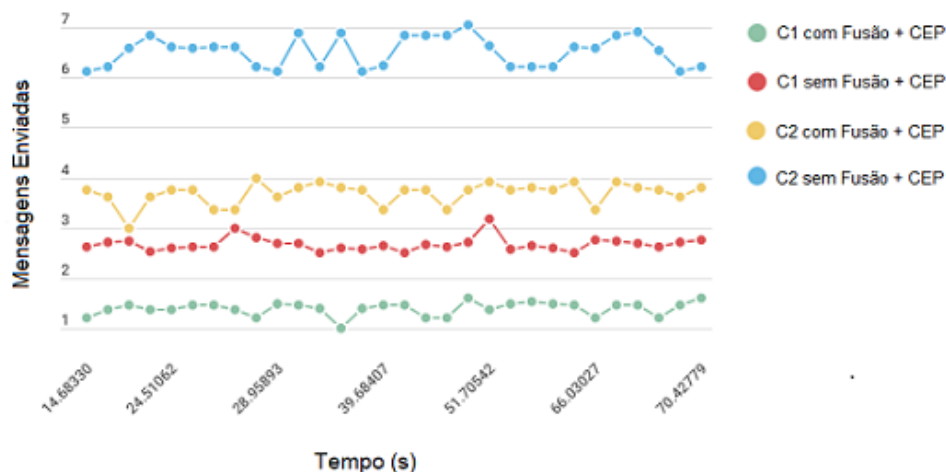


Figura 12 – Vazão dos Cenários 1 e 2

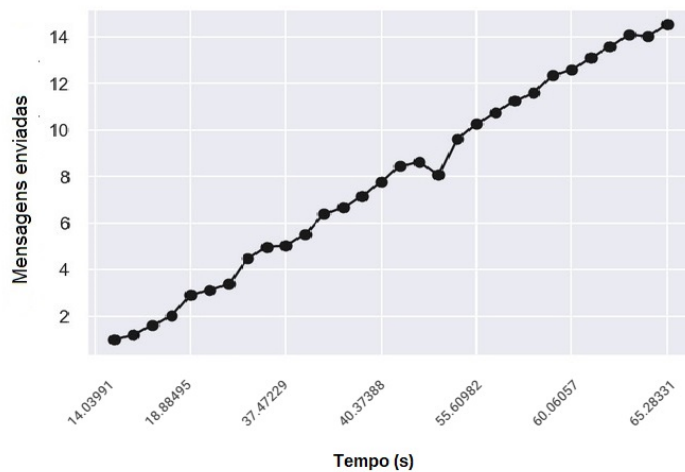


Figura 13 – Vazão do Cenário 3

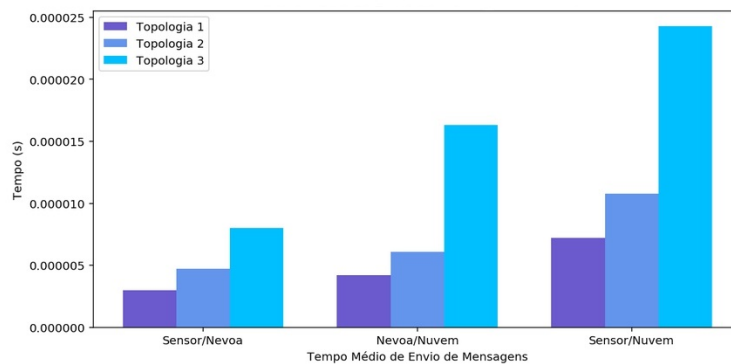


Figura 14 – Tempo Médio de Envio de Mensagens nos 3 Cenários

No cenário 3 a topologia da rede foi sendo alterada com o passar do tempo. Logo, além do tráfego de dados ocorrido nos outros cenários, a rede também precisou trocar informação para alocar os novos dispositivos que estavam sendo adicionados à topologia. Em suma, o aumento gradual da vazão de dados (Fig. 13) se dá tanto pelo aumento dos dados gerados na rede quanto pela troca de mensagens necessárias para sua autoconfiguração. Observadas as vazões nos cenários simulados, é possível verificar que houve redução média de 82% quando comparada a vazão da rede sem a utilização das técnicas.

Outra métrica a ser analisada é o tempo de envio médio de uma mensagem até o destino final. Através do tratamento de dados e eventos realizados pela fusão hierárquica e baseado no modelo CEP, alguns dados que inicialmente seriam enviados à nuvem, são tratados e resolvidos nos níveis anteriores. Com base nos resultados da figura 14, pode-se verificar que o tempo médio de envio de mensagens é 77% menor quando o processamento é feito no nível da névoa, sem necessidade de envio de todos os dados para a nuvem.

7.2 Experimento 2: Validação e Auditoria do Modelo CEP

7.2.1 Validação

Tendo em vista os experimentos apresentados no capítulo 6, a validação do modelo CEP foi feita através da análise estática da rede em relação as seguintes métricas propostas em (Etzion; Niblett, 2011): problema de finalização, evento não utilizado, regra (EPA) não alcançável, regra (EPA) que leva a um caminho morto e possível processamento não determinístico. Cada uma destas métricas está descrita abaixo.

- **Problema de finalização:** o fluxo de eventos na rede CEP não termina devido a um *loop* infinito;
- **Evento não utilizado:** esta observação descobre eventos que são produzidos e não usados. Um produtor pode produzir um evento que não seja consumido por nenhum consumidor ou EPA.
- **Regra não alcançável (RNA):** um EPA nunca é executado porque um ou mais de seus eventos de entrada nunca ocorrem.
- **Caminho morto:** identifica se a regra testada não produz um evento derivado.
- **Processamento não determinístico (PND):** este tipo de observação identifica o fato de que dois ou mais EPAs podem ser executados em qualquer ordem e podem produzir resultados diferentes dependendo de sua ordem de execução.

Primeiro, foi observado o problema de finalização. Para isso foram verificadas as 3 possibilidades de inicialização de eventos, como mostrados nos fluxos das figuras 15, 16 e 17.

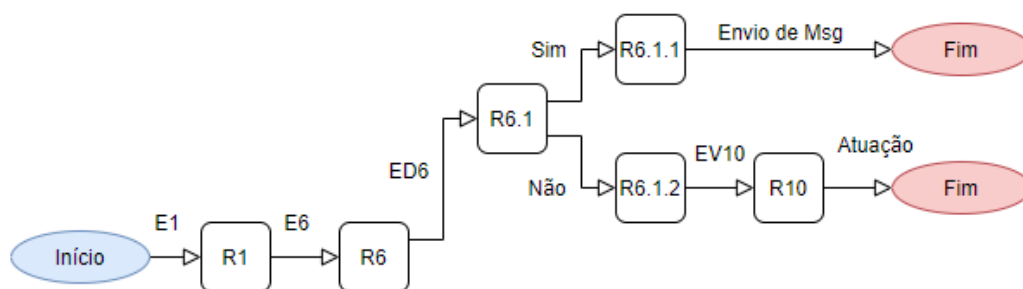


Figura 15 – Fluxo Iniciando com o Evento 1

O propósito dessa observação é verificar ciclos de eventos em cada bloco de fluxo. Um ciclo pode indicar um problema, caso ocorra um laço infinito, onde o processamento fica preso infinitamente entre eventos e não consegue concluí-los. Para que não houvesse a ocorrência de ciclos, utilizou-se o conceito de eventos derivados (ED), como pode ser observado nas figuras 15 e 16. Os eventos derivados são testes realizados a partir de novas

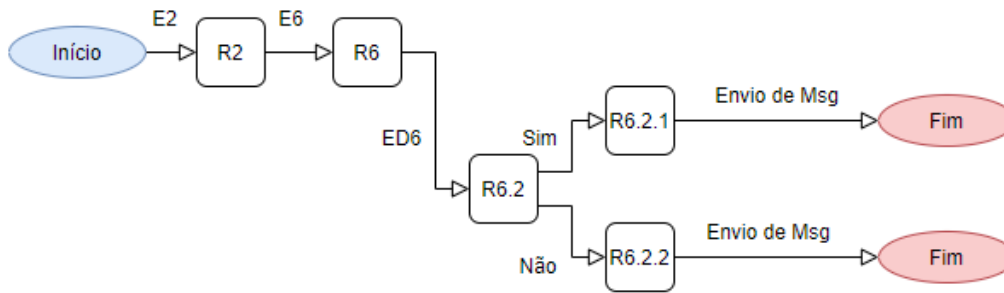


Figura 16 – Fluxo Iniciando com o Evento 2

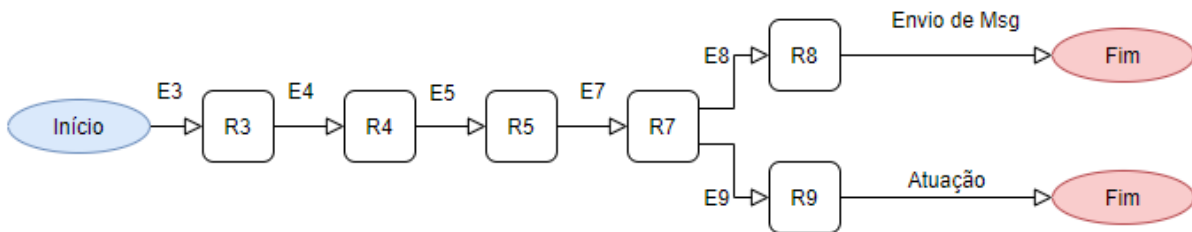


Figura 17 – Fluxo Iniciando com o Evento 3

regras (EPA) dentro de um mesmo evento, removendo a necessidade da ocorrência de ciclos.

Observando todos os fluxos possíveis do modelo CEP, ilustrado na Figura 18 pode-se verificar as próximas três etapas do processo de validação de um modelo CEP. Seguindo todos os fluxos foi observado que não existe nenhum evento criado que não é utilizado, indicando que não há nenhum problema de design no modelo. Pode-se, assim, constatar também que todas as regras são alcançáveis, inclusive as de eventos derivados, que é onde ocorre a maior parte dos erros de RNA. Além disso, não existe nenhum caminho morto, todos os caminhos do fluxo levam a finalização dos processos.

Para verificar a última etapa da validação do modelo CEP utilizando análise estática podemos utilizar o fluxo da Figura 18. Utilizando como exemplo o evento 7 (E7), ele precisa das informações obtidas a partir dos eventos 3, 4 e 5. Sabendo disso, o processamento foi modelado de forma que a regra 7 (R7) só é acionada após a finalização dos eventos anteriores, como pode ser verificado também na descrição das regras contidas no capítulo 4. Isso serve como uma indicação para especificar a ordem de prioridade desses EPAs, ou para redesenhar o fluxo do modelo, caso essa ordem não determinística realmente importe para a aplicação, como é o caso do nosso processamento.

Em suma, essas ferramentas de validação também podem fornecer informações sobre a proveniência do evento. Elas podem operar para frente, mostrando todas as

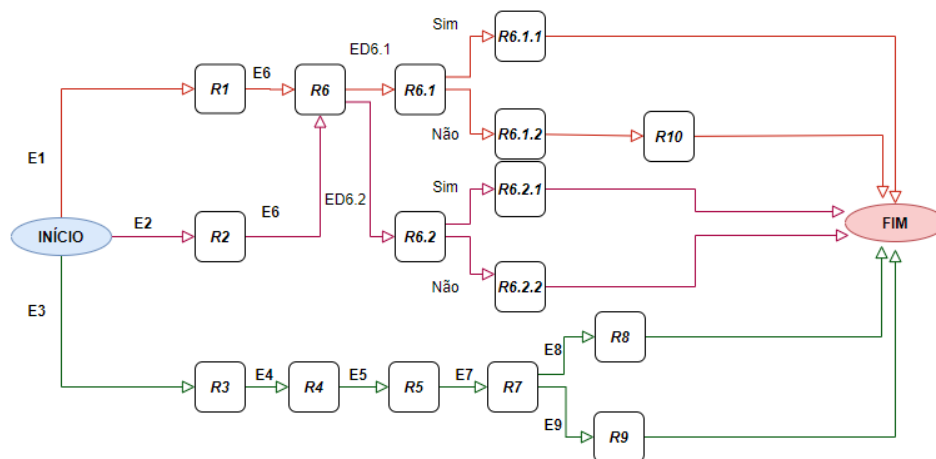


Figura 18 – Fluxos do Modelo CEP

consequências de um determinado evento ou nó na rede de processamento, ou para trás, mostrando todos os antecedentes de um determinado evento ou nó derivado na rede, como foi observado nas figuras 15, 16, 17 e 18.

Logo, como mostrado na tabela 6, foi concluído com sucesso o processo de validação do modelo CEP do sistema de computação em névoa.

Tabela 6 – Validação do Modelo CEP

Problema de Finalização	✓
Evento Não Utilizado	✓
Regra Não Alcançável	✓
Caminho Morto	✓
Processamento Não Determinístico	✓

Nesta seção fizemos a validação do modelo CEP através da análise estática, que verifica a implementação e o fluxo do processamento por meio da análise dos blocos de instruções dos eventos. Para complementar essa análise, na seção a seguir será feita a auditoria para verificar se o processamento está ocorrendo de maneira adequada para a aplicação na qual está sendo utilizada.

7.2.2 Auditoria

O processo de auditoria de um modelo CEP foi feito baseado no *log* de eventos, que é uma lista que retém os eventos brutos e derivados que foram executados em um determinado processamento. A auditoria foi realizada a partir de consultas nesse *log* de eventos. A consulta é semelhante ao processo de análise dinâmica do processo de validação, pois depende da quantidade de informação proveniente da execução de um determinado evento.

Para a auditoria do modelo CEP foram executadas duas consultas principais, baseadas no conceito apresentado em (Etzion; Niblett, 2011):

1. Consultar todos os antecedentes de um determinado evento ou ativação de uma instância EPA (regra);
2. Consultar todas as consequências de um determinado evento ou ativação de uma instância EPA (regra).

Pode-se perceber a partir dessas duas consultas principais que o objetivo da auditoria é verificar os antecedentes e as consequências de um determinado evento processado, ou seja, verificar os antecedentes dos eventos finalizadores e as consequências dos eventos iniciais. Dessa maneira, para o modelo aplicado na arquitetura proposta nesta dissertação é necessário auditar os seguintes eventos:

- **Consulta 1:** EV6, EV8, EV9 e EV10.
- **Consulta 2:** EV1, EV2 e EV3.

As figuras 19, 20 e 21 apresentam os *logs* dos eventos consultados.

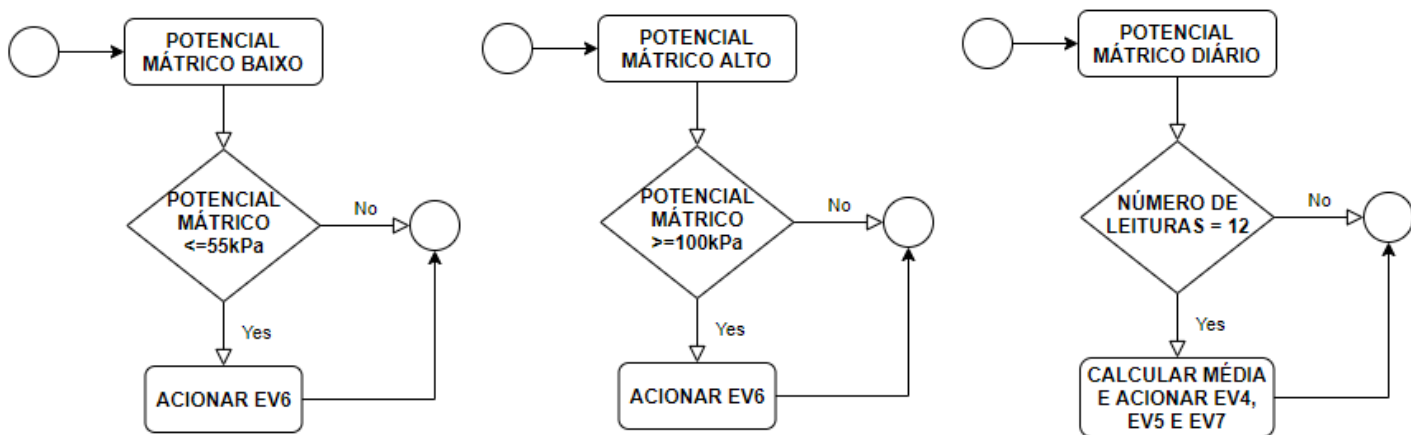


Figura 19 – Logs dos Eventos 1, 2 e 3

A partir dos *logs*, podemos verificar que o processamento de eventos está suprimindo a necessidade da aplicação. Tomando como exemplo o log do Evento 1 ilustrado na Fig. 19. A atuação esperada de acordo com o potencial mátrico baixo é a irrigação, a não ser que tenha ocorrido um mal funcionamento do sensor. Então, seguindo o processamento do evento 1, ele finaliza acionando o evento 6, caso haja a ativação da regra. Observando agora o *log* apresentado na Fig. 20, verificamos que esse evento finaliza exatamente suprimindo a necessidade do acionamento do evento 1: ou finaliza enviando uma mensagem ao usuário informando do mal funcionamento do sensor, ou enviando uma mensagem para acionamento de irrigação manual, ou, por fim, acionando o evento 10. O evento 10, por sua vez, verifica

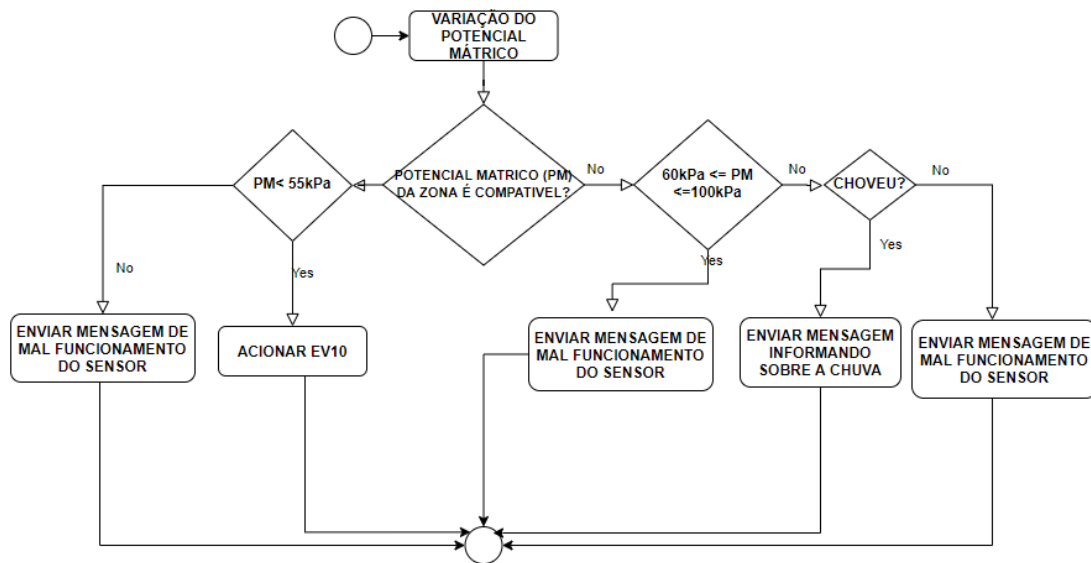


Figura 20 – Log do Evento 6

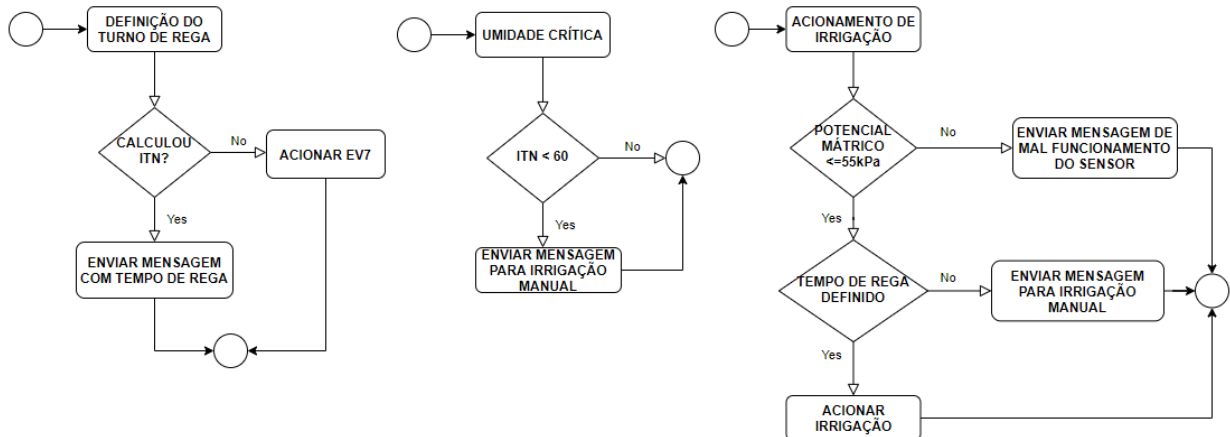


Figura 21 – Logs dos Eventos 8, 9 e 10

mais uma vez a ativação da instancia EPA, testando a regra, e finaliza com o acionamento da irrigação automática ou solicitando que o mesmo seja feito manualmente, como podemos observar no *log* da Figura 21.

Sendo assim, a partir de qualquer consulta executada nos *logs* apresentados, a necessidade da aplicação é satisfeita na finalização do processamento. Partindo do evento 2 (Fig 19) e finalizando no evento 6 (Fig 20) e assim por diante. Dessa forma, a auditoria do modelo CEP utilizado na arquitetura proposta foi satisfeita com sucesso.

8 Conclusão

Nesta pesquisa foi apresentada uma arquitetura de fusão hierárquica de três níveis em conjunto com técnicas de CEP em ambientes de computação em névoa. O sistema é capaz de se adaptar a mudanças físicas no ambiente monitorado, sendo possível os nós sensores adicionados se conectarem a um nó névoa baseando-se no tipo de dado e na distância euclidiana entre eles (autoconfiguração). Para avaliar a proposta foram executadas simulações em cenários distintos de agricultura de precisão, com diferentes densidades de rede e configurações. A avaliação teve como parâmetro de comparação uma rede que usa a camada da névoa apenas como intermediária entre a área monitorada e a nuvem, sem aplicação das técnicas de fusão e CEP. Os experimentos mostraram que a rede obteve resultados satisfatórios, como a redução de 77% no tempo médio de envio de mensagens e 82% de melhoria na vazão média com a aplicação da arquitetura proposta, que pode ser usada para tomada de decisões sobre a área monitorada ou para simples aquisição de conhecimento.

Além disso, foi executada a validação e auditoria do modelo CEP utilizado nas camadas do sensor e da névoa. A validação foi feita por meio da análise estática da rede em relação as métricas de problema de finalização, evento não utilizado, regra (EPA) não alcançável, regra (EPA) que leva a um caminho morto e possível processamento não determinístico. A auditoria foi realizada por meio de consultas nos *logs* das execuções do processamento de eventos. Sendo ambas concluídas com sucesso.

A simulação da proposta nos experimentos realizados resultou na confirmação das hipóteses levantadas no capítulo 1. A utilização da solução arquitetura proposta propiciou uma redução nos dados trafegados na rede e mostrou que a eleição dos nós melhorou o tempo de resposta (Fig. 14) devido a proximidade geográfica entre o nó sensor e o nó névoa que processa seus dados.

Em conjunto com a arquitetura hierárquica, foi utilizado um modelo CEP baseado em dois conjuntos de regras: de monitoramento e de estado crítico. A utilização do mecanismo CEP permitiu a detecção de eventos climáticos (chuva, por exemplo) e ajudou na tomada de decisão (como tempo de rega e momento do início da irrigação). Esse modelo é desacoplado da infraestrutura, permitindo sua aplicação em qualquer outra arquitetura ou para outras finalidades, feitas os devidos ajustes de acordo com a necessidade da aplicação.

Sendo assim, as contribuições dessa pesquisa:

- Arquitetura hierárquica de fusão de dados para computação em névoa com foco em processamento de eventos e tomada de decisões;

- Utilização de técnicas de fusão de dados para reduzir a quantidade de dados transmitidos para a nuvem;
- Uso de lógica de autoconfiguração para eleição de nós da névoa com base no contexto da aplicação e na distância entre os sensores e a névoa;
- Modelagem CEP baseada em regras desacoplado da arquitetura;
- Validação e Auditoria de um modelo CEP desacoplado da arquitetura.

8.1 Trabalhos Futuros

Como trabalhos futuros podem ser realizadas pesquisas em relação a autoconfiguração dos nós quanto as melhores características a serem consideradas para a escolha do nó névoa, como poder de processamento, por exemplo. Outro ponto a ser explorado em trabalhos futuros é o uso do mecanismo CEP orientado a *stream* de dados, que é similar a SQL, ao invés da modelagem orientada a regras. Além disso, seria interessante a realização da implementação do CEP em ambientes como o *flink*, ou algum outro simulador para modelos CEP, para a análise dinâmica da rede de processamento de eventos.

Referências

- Akbar, A. et al. Context-aware stream processing for distributed iot applications. In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. [S.l.: s.n.], 2016. p. 663–668. Citado na página 38.
- Akbar, A. et al. Predictive analytics for complex iot data streams. *IEEE Internet of Things Journal*, PP, p. 1–1, 06 2017. Citado na página 38.
- Akila, V.; Govindasamy, V.; Sandosh, S. Complex event processing over uncertain events: Techniques, challenges, and future directions. In: *2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)*. [S.l.: s.n.], 2016. p. 204–221. Citado na página 36.
- Akila, V.; Govindasamy, V.; Sandosh, S. Complex event processing over uncertain events: Techniques, challenges, and future directions. In: *2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC)*. [S.l.: s.n.], 2016. p. 204–221. ISSN null. Citado na página 45.
- Ali, M. et al. The extensibility framework in microsoft streaminsight. In: . [S.l.: s.n.], 2011. p. 1242–1253. Citado na página 36.
- Alturki, B.; Reiff-Marganiec, S.; Perera, C. A hybrid approach for data analytics for internet of things. In: *Proceedings of the Seventh International Conference on the Internet of Things*. New York, NY, USA: ACM, 2017. (IoT '17), p. 7:1–7:8. ISBN 978-1-4503-5318-2. Disponível em: <<http://doi.acm.org/10.1145/3131542.3131558>>. Citado na página 40.
- Arasu, A. et al. Stream: The stanford stream data manager (demonstration description). In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2003. (SIGMOD '03), p. 665. ISBN 158113634X. Disponível em: <<https://doi.org/10.1145/872757.872854>>. Citado na página 36.
- Benarbia, S.; Alaoui, N.; Bennani, S. Complex event processing distributed architecture for massive open online courses. In: *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*. [S.l.: s.n.], 2018. p. 273–276. ISSN null. Citado na página 47.
- Bierzynski, K.; Escobar, A.; Eberl, M. Cloud, fog and edge: Cooperation for the future? In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.: s.n.], 2017. p. 62–67. Citado na página 25.
- Calheiros, R. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, v. 41, n. 1, p. 23–50, 2011. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.995>>. Citado na página 57.
- Caprolu, M. et al. Edge computing perspectives: Architectures, technologies, and open security issues. In: *2019 IEEE International Conference on Edge Computing (EDGE)*. [S.l.: s.n.], 2019. p. 116–123. ISSN null. Citado na página 25.

- Caprolu, M. et al. Edge computing perspectives: Architectures, technologies, and open security issues. In: *2019 IEEE International Conference on Edge Computing (EDGE)*. [S.l.: s.n.], 2019. p. 116–123. ISSN null. Citado na página 31.
- Carbone, P. et al. Apache flinkTM: Stream and batch processing in a single engine. *IEEE Data Engineering Bulletin*, v. 38, 01 2015. Citado na página 36.
- Coutinho, A.; Carneiro, E.; Greve, F. Computação em névoa: Conceitos, aplicações e desafios. In: _____. [S.l.: s.n.], 2016. p. 266–315. ISBN 2177-4978. Citado na página 31.
- Cremonezi, B. et al. Um sistema multinível de distribuição de identidades em névoas computacionais. In: *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2019. p. 543–555. ISSN 2177-9384. Disponível em: <<https://ojs.sbc.org.br/index.php/sbrc/article/view/7385>>. Citado 2 vezes nas páginas 40 e 42.
- Cugola, G.; Margara, A. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, v. 44, 06 2012. Citado na página 34.
- Dautov, R.; Distefano, S.; Buyya, R. Hierarchical data fusion for smart healthcare. *Journal of Big Data*, v. 6, n. 1, 2019. Disponível em: <<https://www2.scopus.com/inward/record.uri?eid=2-s2.0-85062549653&doi=10.11862fs40537-019-0183-6&partnerID=40&md5=074a943b0014a262d4d2fd1887c1e2fb>>. Citado 3 vezes nas páginas 41, 42 e 44.
- Diao y.; Immerman, N.; Gyllstrom, D. Sase+: An agile language for kleene closure over event streams. 01 2007. Citado na página 36.
- Ding, T.; Gao, Y.; Ma, Y. A semantic preserving cep based approach for business data processing. In: *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. [S.l.: s.n.], 2016. p. 1808–1815. Citado na página 38.
- El-Sayed, H. et al. Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment. *IEEE Access*, v. 6, p. 1706–1717, 2018. Citado 2 vezes nas páginas 27 e 28.
- EsperTech. Esper - complex event processing. 2019. Disponível em: <<http://www.espertech.com/esper/>>. Citado na página 36.
- Etzion, O.; Niblett, P. *Event Processing in Action*. 1a edição. ed. [S.l.]: Manning, 2011. Citado 3 vezes nas páginas 59, 63 e 66.
- Frizzzone, J. *Necessidade de água para irrigação (Lecture Notes)*. [S.l.], 2017. Citado 2 vezes nas páginas 51 e 52.
- Gupta, H. et al. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, v. 47, n. 9, p. 1275–1296, 2017. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509>>. Citado na página 57.
- Kondo, T. et al. Development of the edge computing platform with dynamic modular configuration for an iot platform. In: *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. [S.l.: s.n.], 2019. p. 1–6. Citado 2 vezes nas páginas 41 e 42.

- Lan, L. et al. A universal complex event processing mechanism based on edge computing for internet of things real-time monitoring. *IEEE Access*, v. 7, p. 101865–101878, 2019. Citado 3 vezes nas páginas 28, 39 e 42.
- Lee, J. et al. Development of a continuous complex event processing platform for real-time tactical moving objects. In: *2019 IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2019. p. 6122–6124. Citado na página 38.
- Lera, I.; Guerrero, C.; Juiz, C. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, v. 7, p. 91745–91758, 2019. ISSN 2169-3536. Citado na página 57.
- Mazon-Olivo, B. et al. Rules engine and complex event processor in the context of internet of things for precision agriculture. *Computers and Electronics in Agriculture*, v. 154, p. 347 – 360, 2018. ISSN 0168-1699. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0168169918302436>>. Citado 2 vezes nas páginas 40 e 42.
- Merkel, D.; Cocos, H. Complex event processing on the edge - bringing data consolidation and processing closer to wireless sensor networks. p. 6, 2019. Citado 2 vezes nas páginas 27 e 42.
- Mouradian, C. et al. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, v. 20, n. 1, p. 416–464, Firstquarter 2018. ISSN 2373-745X. Citado na página 25.
- M.Roriz et al. Introdução ao processamento de fluxos de dados: Uma abordagem orientada a eventos complexos. In: *XXV Simpósio Brasileiro de Sistemas Multimídia e Web*. RJ, Brasil: [s.n.], 2019. p. 45–75. Citado 2 vezes nas páginas 34 e 36.
- Nakamura, E.; Loureiro, A.; A.Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys (CSUR)*, v. 39, p. 9, 09 2007. Citado 2 vezes nas páginas 26 e 33.
- Oliveira, S.; Kniess, J. Redução de dados em redes de sensores sem fio baseada em análise de dispersão. In: *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.: s.n.], 2019. p. 1–14. Citado 2 vezes nas páginas 40 e 42.
- Paschke, A. Design patterns for complex event processing. 07 2008. Citado na página 35.
- R.Dautov; S.Distefano. Distributed data fusion for the internet of things. In: *PaCT*. [S.l.: s.n.], 2017. p. 427–432. ISBN 978-3-319-62931-5. Citado 2 vezes nas páginas 41 e 42.
- Shi, W. et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, v. 3, n. 5, p. 637–646, 2016. Citado na página 27.
- Silva, N. *CSmartGreen: Um Framework de Internet das Coisas para Agricultura Inteligente*. Tese (Doutorado) — Universidade Federal do Ceará, Centro de Tecnologia, Curso de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2019. Citado 2 vezes nas páginas 51 e 52.
- Suhothayan, S. et al. Siddhi: A second look at complex event processing architectures. In: . [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 35 e 36.

TORRES, A. B. et al. Multilevel data fusion for the internet of things in smart agriculture. *Computers and Electronics in Agriculture*, v. 171, p. 105309, 2020. ISSN 0168-1699. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0168169919322999>>. Citado 2 vezes nas páginas 41 e 42.

Torres, A. B. B. Fusão de dados multinível para sistemas de internet das coisas em agricultura inteligente. In: . [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 15 e 33.

Vora, J. et al. Faal: Fog computing-based patient monitoring system for ambient assisted living. In: *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*. [S.l.: s.n.], 2017. p. 1–6. ISSN null. Citado 2 vezes nas páginas 41 e 42.

Wang, Y.; K.Cao. A proactive complex event processing method for large-scale transportation internet of things. *International Journal of Distributed Sensor Networks*, v. 2014, p. 1–8, 03 2014. Citado na página 38.

Wei, X.; Wu, L. A new proposed sensor cloud architecture based on fog computing for internet of things. In: *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCoM) and IEEE Smart Data (SmartData)*. [S.l.: s.n.], 2019. p. 615–620. Citado 2 vezes nas páginas 40 e 42.

Zhang, L. et al. Multi-source heterogeneous data fusion. In: *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. [S.l.: s.n.], 2018. p. 47–51. ISSN null. Citado na página 32.

Zhang, Q. et al. Firework: Big data sharing and processing in collaborative edge environment. In: *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. [S.l.: s.n.], 2016. p. 20–25. ISSN null. Citado 2 vezes nas páginas 41 e 42.

Zhou, Q.; Simmhan, Y.; Prasanna, V. Knowledge-infused and consistent complex event processing over real-time and persistent streams. *Future Generation Computer Systems*, v. 76, p. 391 – 406, 2017. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X16304769>>. Citado na página 38.