UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE MATEMÁTICA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FABIANO DE PAULA MARTINS

UM ALGORITMO DE FUSÃO DE DADOS PARA AMBIENTES INTELIGENTES USANDO REDES DE SENSORES SEM FIO

RIO DE JANEIRO

2020

FABIANO DE PAULA MARTINS

UM ALGORITMO DE FUSÃO DE DADOS PARA AMBIENTES INTELIGENTES USANDO REDES DE SENSORES SEM FIO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Claudio Miceli de Farias, D.Sc.
NCE - UFRJ

Co-orientador: Prof. João Antônio Recio da Paixão, D.Sc.
DCC - IM - UFRJ

RIO DE JANEIRO

2020

CIP - Catalogação na Publicação

M386a

Martins, Fabiano de Paula
Um algoritmo de fusão de dados para ambientes inteligentes usando redes de sensores sem fio / Fabiano de Paula Martins. -- Rio de Janeiro, 2020.
71 f.

Orientador: Claudio Miceli de Farias.
Coorientador: João Antônio Recio da Paixão.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2020.

1. IoT. 2. Fusão de dados. 3. Ambientes inteligentes. I. Farias, Claudio Miceli de, orient. II. Paixão, João Antônio Recio da, coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

FABIANO DE PAULA MARTINS

UM ALGORITMO DE FUSÃO DE DADOS PARA AMBIENTES INTELIGENTES USANDO REDES DE SENSORES SEM FIO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 08 de fevereiro de 2020.

BANCA EXAMINADORA:

Prof. Claudio Miceli de Farias, D.Sc.
NCE - UFRJ

Prof. João Antônio Recio da Paixão, D.Sc.
DCC - IM - UFRJ

Profa. Juliana Viana Valério, D.Sc.
DCC - IM - UFRJ

Profa. Silvana Rossetto, D.Sc.
DCC - IM - UFRJ

Dedico à minha família, aos meus
antepassados e àqueles aos quais legarei a
sociedade em que vivo.

AGRADECIMENTOS

Agradeço a Deus por ter me protegido e preservado nessa caminhada toda. Os "Meus pés não resvalaram"[1] e, quando meus calos me incomodavam, a Sua misericórdia e o Seu amor me mostraram que eles me deixaram mais preparado para o caminho. Agradeço aos meus pais Henrique e Teresinha porque, sem dúvida, foram os melhores pais do mundo, fosse nos 'não's e nos 'sim's , me passaram os valores da honestidade e da busca pelo certo.

Agradeço aos meus orientadores João e Cláudio pela paciência, pela amizade e por terem me ensinado a fazer esse projeto. Sem eles este trabalho não seria o mesmo e eu não teria aprendido tanto como aprendi. Por fim, agradeço por terem acreditado em mim e me ajudarem a ter autoconfiança.

Agradeço aos professores do DCC-UFRJ que fizeram parte da minha trajetória: Severino Collier Coutinho, que facilmente entrou para o pódio dos melhores professores que conheci, ensinando-me a amar suas disciplinas por meio de sua didática surpreendente; Juliana Vianna Valério, que me acompanhou desde o início de minhas atividades na iniciação científica, me ensinou cálculo numérico e me recebeu como monitor na mesma disciplina; Valéria Menezes Bastos, que, de portas abertas e pacientemente, ouvia nossos murmúrios e nos ajudava a ressignificar os desesperos da vida acadêmica; João Carlos Pereira da Silva, com quem tive vários debates filosóficos e sempre aceitava uma conversa complicada sobre ética, moral e o sentido das coisas; Silvana Rossetto, que foi uma incrível professora de Computação Concorrente e me ensinou coisas preciosas para este trabalho; e a todos os professores do DCC que me formaram seja pelo conhecimento, pelo exemplo ou pelo esforço.

Agradeço a Jonathan Nogueira Gois, meu amigo desde a adolescência, que me deu várias caronas e vários conselhos para o Fundão a quem agradeço pela honra de me permitir ser seu padrinho do seu casamento com a minha amiga Dayane Rocha Gois. Ao Leonardo Tavares, que me abriu os olhos para um mundo que eu ignorava e me apresentou inúmeras possibilidades. Ao Ciro Monteiro, de cujas conversas sinceras

[1] Salmo 120, 3

sempre vou lembrar das conversas sinceras, principalmente aquelas que ocorreram acompanhadas do Sanduíche de filé-mignon do Restaurante Cervantes. Aos Bruno Alves do Carmo e Charles Barros, grandes amigos que me ajudaram a aprender matemática e a administrar o LC3[2]. Agradeço ao Pe. João Paulo Batista que, em seu sermões, sempre convidou e convida a pensar a vida com os pés no chão, os olhos no céu. Agradeço a todos aqueles que não foram mencionados por falta de espaço e também me ajudaram até aqui!

Agradeço a Lucas Pérez e David Gibbin que me levaram para a Investtools, a melhor empresa para se trabalhar durante a elaboração deste projeto.

Agradeço a Nelson Rodrigues que me ensinou a como não ser um covarde. Por fim, a Luiz Felipe Pondé que fez um dos convites mais complexos dos últimos anos: pensar com a própria cabeça.

[2]Laboratório de Combinatória e Computação Científica

"O que fazemos em vida, ecoa pela eternidade."

Maximus, o gladiador

RESUMO

O uso de redes de sensores sem fio para monitorar ambientes é sempre acompanhado do desafio de maximizar a acurácia e minimizar o consumo de energia das redes de sensores. Esse desafio tem acompanhado o uso dessas redes de sensores por todo o seu desenvolvimento, desde redes de sensores exclusivas dedicadas a uma única aplicação até redes desenvolvidas para serem autônomas e dedicadas a diversas aplicações com os mais variados propósitos. Para servir várias aplicações com pouco consumo de energia, métodos de fusão de dados voltados para redes de sensores têm sido aplicados aos algoritmos de monitoramento. Este trabalho apresenta o *Hercules*, um algoritmo de fusão de dados para redes de sensores capaz de servir várias aplicações. Analisando os picos da curva de frequência, os dados são separados em função dos fenômenos monitorados no ambiente. Os resultados mostram que o *Hercules* consegue ter mais acurácia e consumir menos energia por ser um algoritmo determinístico e sensível ao contexto.

Palavras-chave: IoT. Fusão de Dados. Ambientes Inteligentes.

ABSTRACT

The use of wireless sensor network to monitor environments is always followed by the challenge of maximizing accuracy and minimizing energy consumption. This challenge has followed the use of wireless network sensors for all its development ranging from sensor exclusive networks for only one application until now with autonomous sensor networks to serve many aplications with many kind of aims. To serve many applications with low cost consumption of energy, multisensor data fusion methods were added to monitoring algorithms. This work proposes Hercules as an multisensor data fusion algorithm able to serve many applications. Analizing the frequency curve peaks, the data is separated by the type of phenomenona monitored in the environment. The results show that Hercules can be more accurate and consume less energy because it is a deterministic and context-sensitive algorithm.

Keywords: IoT. Multisensor Data Fusion. Smart Environments.

LISTA DE FIGURAS

Figura 1: Ambos os conjuntos de dados são classificados como plato-cúrticos mono-modais pelo *Hepheastus*. 17
Figura 2: Camadas que compõe uma aplicação IoT. 20
Figura 3: Gráfico que apresenta duas concentrações com dispersões diferentes. 35
Figura 4: Diagrama de etapas do *Hercules* 36
Figura 5: Um exemplo de curva de frequência 37
Figura 6: Um exemplo de curva de frequência em azul, intervalos em verde e a função de médias em laranja. 38
Figura 7: Exemplo do fluxo de trabalho da rede exibindo momentos em que a rede somente coleta dados e outros em que ela reúne os dados e gera os resultados 39
Figura 8: Pilha de protocolos do Contiki 49
Figura 9: Fluxo de dados do *Hercules* 50
Figura 10: Dispersão dos nós no ambiente emulado de 100mX100m 57

LISTA DE TABELAS

Tabela 1: Trabalhos relacionados separados em grupos 33
Tabela 2: Mensagens do *Hercules* 51
Tabela 3: Exemplos de mensagens 51
Tabela 4: Tabela de estados do ambiente monitorado e os status dos dispositivos monitorados. 54
Tabela 5: Tabela de estados do ambiente monitorado 55
Tabela 6: Tabela de consumo durante os testes em Watts. 59
Tabela 7: Tabela de tempo de duração das pilhas AA em minutos 59
Tabela 8: Tabela de consumo de memória dos códigos compilados no Contiki 60
Tabela 9: Parâmetros das variáveis aleatórias em cada estado. 61
Tabela 10: Intervalos de confiança e estimativa da acurácia da aplicação da rede de potência 62
Tabela 11: Intervalos de confiança e estimativa da acurácia da aplicação da bateria 62
Tabela 12: Intervalos de confiança e estimativa da acurácia do experimento . . 62

LISTA DE ABREVIATURAS E SIGLAS

RSSF	Rede de Sensores Sem Fio
LDLS	Lista de Dados Lidos pelos Sensores
OCF(n)	Lista de Ordenadas da Curva de Frequência com n classes
DCF(n)	Lista de Dados da Curva de Frequência com n classes
OFM(m)	Lista de Ordenadas da Função de Médias com m intervalos
DFM(m)	Lista de Dados da Função de Médias com m intervalos
CFM(m)	Lista de Classificação dos m Pontos da Função de Média
LPC	Lista de Pontos de Cisão
LSEI	Lista de Subconjuntos de Eventos Isolados

SUMÁRIO

1 INTRODUÇÃO 14

2 CONCEITOS BÁSICOS 19

2.1 INTERNET DAS COISAS 19

2.2 REDES DE SENSORES SEM FIO 21

2.3 MULTISENSOR DATA FUSION 23

3 TRABALHOS RELACIONADOS 26

4 PROPOSTA 34

4.1 VISÃO GLOBAL DO *HERCULES* 34

4.2 ESTRUTURAS DE DADOS DO *HERCULES* 40

4.3 FUNÇÃO DE MÉDIAS 40

4.4 IDENTIFICAÇÃO DE PONTOS DE CISÃO 42

4.5 CALIBRAÇÃO DOS PARÂMETROS 45

4.5.1 Estruturas de dados do método de calibração 45

4.5.2 Método de calibração 46

5 IMPLEMENTAÇÃO 48

5.1 PLATAFORMA DE DESENVOLVIMENTO 48

5.2 IMPLEMENTAÇÃO DO PROTÓTIPO 49

5.3 PROTOCOLO DE SINCRONIZAÇÃO 50

5.4 PROTOCOLO DE COMUNICAÇÃO 50

6 EXPERIMENTOS 53

6.1 DESCRIÇÃO DO CASO 53

6.2 MÉTRICAS 55

6.3 CONFIGURAÇÃO DE AMBIENTE 56

6.4 MODELO DE ENERGIA 57

6.5 DESCRIÇÃO DOS RESULTADOS 58

6.5.1 Quantidade de energia consumida 58

6.5.2 Quantidade de memória consumida **59**
6.5.3 Teste de Acurácia **60**

7 CONCLUSÃO **63**
7.1 TRABALHOS FUTUROS 64

REFERÊNCIAS **66**

1 INTRODUÇÃO

O crescimento de grandes metrópoles cria a necessidade de sistemas de apoio à tomada de decisão (Chourabi et al., 2012). O crescimento e a concentração populacional em centros urbanos geram uma demanda por serviços como energia elétrica, água encanada, transporte, entre outros (Farias, 2014). Com enormes demandas advindas do tamanho, da diversidade e da atividade econômica da população, sistemas geridos e monitorados unicamente por humanos se tornaram ineficientes para tão grande e diversa demanda de serviços. A otimização da oferta destes recursos requer sistemas autônomos que mantenham a cadeia de suprimentos funcionando, evitam que serviços fiquem indisponíveis e atendam o máximo possível de pessoas simultaneamente (Lambert and Cooper, 2000).

Para atender essa demanda, surgem tecnologias de informação e comunicação destinadas a monitorar de ambientes críticos, como o suprimento de energia de uma cidade, ou aplicadas a outros sistemas que não sejam relacionados ao abastecimento de recursos com "o ganho vital de resolverem problemas ambientais relacionados pela degradação natural" (Farias, 2014, p. 21). Por exemplo, o monitoramento de regiões de mata ao redor das cidades cujo objetivo é identificar focos de incêndio e viabilizar uma resposta mais rápida evitando ou amenizando tragédias (Shaikh et al., 2017). Sendo assim, os sistemas devem ser minimamente invasivos e não afetar o ambiente monitorado usando dispositivos que não afetem o ambiente e colaborem entre si.

A internet das coisas (*Internet of Things - IoT*) é um paradigma que propõe conectar todos os dispositivos de um determinado ambiente e que, usando essa rede de comunicação, seja a infraestrutura de sistemas inteligentes (Kopetz, 2011). A partir da integração desses dispositivos em rede, temos as seguintes características: "identificação autêntica de dispositivos inteligentes, administração autônoma e autorregulação de redes de objetos inteligentes, diagnóstico e manutenção, consciência do contexto e comportamento orientado a objetivos, e intrusão da privacidade" (Farias, 2014, p. 21). Não importa o tamanho, mas o fim da aplicação é que ocorra uma integração entre os dispositivos e os dados que são coletados no ambiente

monitorado.

Esses dispositivos possuem sensores embutidos, como os celulares, ou sensores dedicados, que têm como principal característica a coleta de dados do ambiente, seja temperatura, umidade ou qualquer grandeza que o sensor mensure (Kopetz, 2011). Esses dispositivos podem ser alimentados ou não por bateria e possuem capacidade de comunicação sem fio. Isso tudo reduz os custos e a complexidade de implementação dos sistemas, aumenta a flexibilidade no sensoreamento de ambientes e a resiliência do sistema como um todo. Um exemplo é a falha na cobertura do ambiente, que são áreas do ambiente sem monitoramento, que é simples de resolver reposicionando sensores ou substituindo sensores defeituosos. O ponto negativo desses dispositivos é a dependência que eles criam entre a vida do sistema e a duração das baterias dos dispositivos (Gubbi et al., 2013).

A duração média da bateria é uma das métricas mais relevantes na hora de avaliar um sistema IoT, porque garante o tempo de funcionamento sem que haja uma intervenção no ambiente e isso pode variar dependendo da finalidade do dispositivo. Em IoT, o sistema está distribuído entre os dispositivos que coletam, os que transmitem e os que aplicam algoritmos para tratar e analisar os dados. Um artifício que é essencial para aumentar o tempo de vida de um sistema IoT é usar algoritmo de agregação e/ou fusão de dados nos dispositivos que tratam e analisam os dados.

Um algoritmo de agregação e/ou fusão de dados dentro da rede é usado para reduzir o consumo de energia da rede, previnir transmissões desnecessárias e reduzir o volume de dados transmitidos tornando a rede mais eficiente (Farias, 2014). "As técnicas de fusão de dados combinam dados de diversos sensores, e informação relacionada de banco associados, para alcançar uma acurácia superior e inferências mais específicas do que as obtidas com o uso de um único sensor"(Hall and Llinas, 1997). O resultado pode ser uma síntese de dados ou uma transformação dos dados para um nível mais próximo da camada de decisão. Para IoT, precisamos de algoritmos de fusão que sintetizem os dados para que o volume de dados transmitidos seja reduzido e economize transmissão na rede (Aquino et al., 2016).

Em redes IoT como as dos trabalhos (Wagner et al., 2010; Singhal and Gari-

mella, 2012; Dziengel et al., 2016; de Farias et al., 2012b; Farias et al., 2014), os requisitos da aplicação já eram conhecidos previamente e, com isso, os dispositivos e a rede eram projetados especificamente para os requisitos da aplicação. Nos cenários atuais, os requisitos da aplicação são desconhecidos previamente, assim como qualquer informação sobre os intervalos considerados ou eventos que serão monitorados, porque o objetivo é reaproveitar a mesma infraestrutura para diversas aplicações, em diversos contextos. O resultado é o sensoriamento como um serviço que provê dados às aplicações executadas no ambiente monitorado, fazendo da rede um repositório para ser consumido por diversas aplicações com seus respectivos requisitos (Zaslavsky et al., 2013; Aquino et al., 2016).

Um algoritmo sem requisitos é o *Hepheastus* (Aquino et al., 2016), que, a partir da curtose, assimetria e média, separa os conjuntos de dados em subconjuntos monomodais que se aproximam mais de um monitoramento específico de cada evento que ocorre no ambiente monitorado. O *Hepheastus* é um método que tem por objetivo identificar e diferenciar diversos eventos que estejam ocorrendo num ambiente monitorado. O *Hepheastus*, ao monitorar uma torre de transmissão de energia elétrica, é capaz de identificar que a temperatura da bateria é diferente da rede de potência (Aquino et al., 2016).

Entretanto, o *Hepheastus* faz análise a partir de estatística descritiva (média, curtose e assimetria) e conjuntos diferentes podem apresentar a mesma assimetria e não serem classificados como diferentes. Na figura 1, os dois conjuntos são fortemente simétricos e, pelo *Hepheastus*, não serão mais particionados porque o algoritmo assume que conjuntos fortemente simétricos não devem ser particionados (Sanquetta et al., 2014). Porém o segundo caso apresenta dois picos de frequências que serão considerados como sendo um único pico por serem fortemente simétricos em relação à reta $x = 10$. Nesse caso, o ideal seria particionar esse conjunto como dois conjuntos, cada um contendo os dados referentes aos picos descritos.

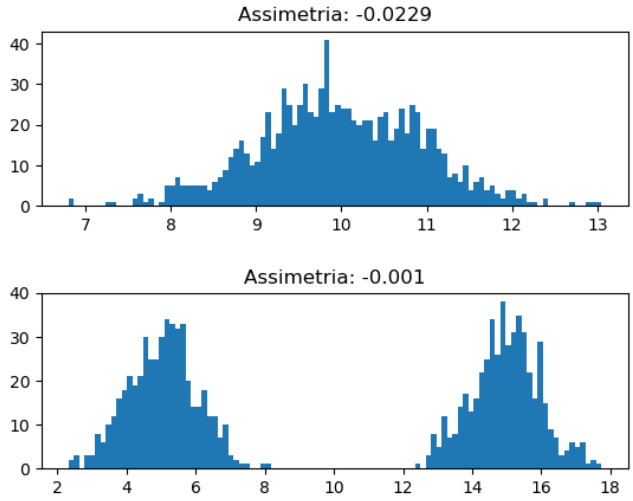

Figura 1: Ambos os conjuntos de dados são classificados como plato-cúrticos monomodais pelo *Hepheastus*.

Nesse trabalho, é apresentado o algoritmo *Hercules* que tem por objetivo principal conseguir identificar, como exemplificado na figura 1, que na primeira curva há somente uma única concentração de frequências e na segunda há duas concentrações de frequências (vamos chamar cada concentração de evento). Neste trabalho, para identificar eventos optou-se por mudar a perspectiva de análise que, ao invés de usar estatística descritiva, usa máximos e mínimos locais. O objetivo dessa abordagem é evitar que as decisões sejam tomadas a partir de somente um número, usando funções matemáticas com mais informações, como intervalos de crescimento e decrescimento, percebendo quais são os máximos e mínimos.

O *Hepheastus* tem um algoritmo que não considera o contexto da aplicação na sua execução. Por outro lado, *Hercules* (o método deste trabalho) considera o contexto da aplicação se adaptando ao conjunto de dados para fazer a análise. O *Hercules* não necessita de pré-requisitos da aplicação, mas seu desempenho está diretamente atrelado ao quão adaptado está ao contexto e quão ajustados são seus parâmetros à configuração do ambiente monitorado. Por isso, *Hercules* não depende dos requisitos das aplicações, mas do ambiente monitorado e suas características.

No capítulo 2, são abordados os conceitos básicos: Internet das Coisas, Redes de

Sensores Sem Fio e Fusão da Informação. Os trabalhos relacionados são comentados no capítulo 3. No capítulo 4, é apresentado o método *Hercules* que motiva este trabalho. As implementações em *Python* e *Contiki/Cooja* são detalhadas no capítulo 5. Os experimentos feitos para testar a acurácia, consumo de energia e consumo de memória são apresentados no capítulo 6.

2 CONCEITOS BÁSICOS

Neste capítulo apresentamos alguns conceitos que servirão de base para os assuntos deste trabalho. Este capítulo é organizado da seguinte forma: (i) seção 2.1 apresenta os conceitos básicos de internet das coisas; (ii) seção 2.2 sobre redes de sensores sem fio e (iii) seção 2.3 apresenta conceitos sobre fusão da informação em redes de sensores.

2.1 INTERNET DAS COISAS

A *Internet das Coisas (IoT - Internet of Things)* (Gubbi et al., 2013) pode ser entendida como um paradigma para o desenvolvimento de aplicações. O termo tem muitas definições diferentes que são pontos de vista sobre o mesmo fenômeno que, por sua vez, engloba diversas áreas da computação e pode ser encarado por diversos aspectos (Atzori et al., 2010). Neste trabalho, a Internet das Coisas é um conjunto de dispositivos, sensores e serviços ligados à internet que fazem leituras de ambientes e geram informações e relatórios usados na tomada de decisão (Da Xu et al., 2014; Al-Fuqaha et al., 2015). Segundo Da Xu et al. (2014), essa definição traz dois grandes desafios: a interoperabilidade desses dispositivos diversos e a arquitetura dessa rede de serviços.

A internet das coisas começou a tomar destaque com o desenvolvimento do *Radio Frequency Identification (RFID)* e depois, com o surgimento das redes de sensores sem fio (RSSF), o interesse cresceu (Atzori et al., 2010). O *Radio Frequency Identification* permitiu identificar diversos dispositivos tais como sensores, celulares e demais objetos inteligentes tornando viável comunicar, interagir e cooperar com baixo custo (Atzori et al., 2010). As redes de sensores sem fios auxiliaram a propriedade *Plug and Play* das aplicações de internet das coisas, permitindo que os objetos sejam intercambiáveis entre diversas redes e aplicações.

A infraestrutura de uma aplicação de internet das coisas se divide em cinco camadas, segundo Al-Fuqaha et al. (2015):

- **Camada dos objetos**: Sensores e outros dispositivos que coletam e pré-processam a informação.
- **Camada de objetos abstratos**: Transferem os dados para a camada superior e possibilitam que a variedade de dispositivos se comunique numa aplicação da internet das coisas. São protocolos de comunicação como: *RFID*, *3G*, *Wi-fi*, *Bluetooth* etc.
- **Camada de administração de serviço**: É uma camada que não considera a origem dos dados e as aplicações dessa camada são voltadas para a análise de dados, a tomada de decisão e o envio de resultado para os usuários da aplicação.
- **Camada de aplicação**: É responsável por prover uma interface *user-friendly* para o cliente e funcionalidades que atendam às suas necessidades.
- **Camada de negócio**: Realiza análise dos resultados providos pela camada de manutenção de serviço, mas usando método de análise de alto custo. Faz uso de *Big Data* e demais técnicas de análise para prover relatórios, gráficos e resultados que auxiliem na tomada de decisão.

Figura 2: Camadas que compõe uma aplicação IoT.

A figura 2 apresenta uma esquematização das camadas. Essa estrutura em camadas impõe três desafios ao desenvolvimento de uma aplicação de internet das coisas (Da Xu et al., 2014): (i) montar a arquitetura orientada a serviços específica

para internet das coisas; (ii) do ponto de vista da rede, conectar todos os diversos dispositivos por diversas tecnologias; (iii) do ponto de vista de serviços, uma linguagem de comunicação que descreva os serviços de forma simples e que simplifique a integração entre eles. E com esses desafios, Da Xu et al. (2014) definem etapas no desenvolvimento de uma aplicação de internet das coisas: (i) desenvolver as plataformas que compõem os serviços; (ii) abstrair as funcionalidades e capacidades de comunicação dos dispositivos conectados em rede; (iii) prover um conjunto de serviços em comum.

Esses passos descrevem o desenvolvimento de uma aplicação que seja livre de contexto, que é um dos objetivos das pesquisas em internet das coisas (Da Xu et al., 2014). Os primeiros sistemas de internet das coisas eram desenvolvidos para cenários específicos. Atualmente, a necessidade de reduzir os custos de implementação e reaproveitamento de infraestrutura impõe o estudo de aplicações de internet das coisas sem contexto, permitindo o seu reaproveitamento em diversos cenários.

2.2 REDES DE SENSORES SEM FIO

As Redes de Sensores sem Fios (*Wireless Sensor Networks - WSN*) são uma área de pesquisa que usa sensores com interfaces de comunicação sem fio e baixo consumo de energia para implementar redes com o objetivo de monitorar ambientes (Akyildiz et al., 2002). "Redes de sensores sem fio ganharam a atenção do mundo nos últimos anos, particularmente com a proliferação da tecnologia de sistemas microeletromecânicos que facilitaram o desenvolvimento de sensores inteligentes" (Yick et al., 2008, p. 1). Redes de sensores sem fio (RSSF) têm mostrado resultados consistentes na redução de custos com infraestrutura de sistemas e na possibilidade de resolver problemas com estruturas mais simples.

As aplicações de *RSSF* envolvem diversas áreas: questões militares, meio ambiente, casas inteligentes, medicina, cidades inteligentes e cadeia de suprimentos. Os dispositivos são sensores que monitoram variáveis do ambiente: temperatura, umidade, posição, luminosidade, pressão, volume de ondas sonoras, velocidade e outros

(Akyildiz et al., 2002). Cada leitura é feita de forma independente por cada sensor e, depois de feitas essas leituras, a rede deve reportar aos sistemas externos algum tipo de resultado ou análise que permita a tomada de decisão, se eles mesmos não tiverem a lógica de tomada de decisão embutida.

Por serem dispositivos independentes, que colaboram entre si, escalar RSSFs é bem simples e barato por consistir apenas em acrescentar mais dispositivos no alcance da rede. Ao adicionar novos sensores, os mesmos são assimilados pelos protocolos de autorregulação (Sohrabi et al., 2000). A simplicidade em adicionar novos sensores faz com que as *RSSFs* tenham alta resiliência à falha de componentes, porque cada sensor funciona alimentado por uma bateria independente (Akyildiz et al., 2002).

Este é um dos critérios de avaliação da eficiência de sistemas em *RSSFs*. Como cada sensor funciona alimentado por uma bateria, o tempo de duração das baterias implica na sobrevivência da rede (Dietrich and Dressler, 2009). A duração da bateria influencia na reposição dos sensores e/ou suas baterias e maximizar esse tempo entre as manutenções é imprescindível para garantir um baixo custo e a viabilidade dos projetos.

Com isso, os algoritmos que são executados nesses sensores têm que considerar em seu design a minimização do custo de processamento e consumo de recurso de rede. "Tempo de vida da rede é considerado como o mais importante parâmetro para determinar o uso de redes de sensores ou os algoritmos a serem usados na rede de sensores"(Dietrich and Dressler, 2009, p. 3). A economia de energia é um fator crucial na eficiência da redes de sensores (Bandyopadhyay and Coyle, 2003).

Por fim, as redes de sensores sem fio proveem o uso de uma infraestrutura versátil e de baixo custo que viabilize uma simplificação ao implementar sistemas de monitoramento. Sua comunicação em rede permite incluir dispositivos dos mais diversos modelos, possibilitando criar aplicações de IoT dos mais diversos níveis de complexidade (Delicato et al., 2003).

2.3 MULTISENSOR DATA FUSION

Multisensor Data Fusion(MDF) é uma área que estuda formas de integrar dados de diversas fontes, sejam elas homogêneas ou não, gerando assim um resultado mais relevante, em algum sentido, que cada fonte de dados isoladamente (Nakamura et al., 2007). Como minimizar consumo de energia é fundamental em *RSSFs* , a MDF oferece formas de acessar os dados coletados pelas *RSSFs* possibilitando ou não reduzir os custos de envio das informações (Khaleghi et al., 2013). Os métodos de MDF são classificados segundo quatro características: (i) nível de abstração da informação, (ii) informação de entradas e informação de saída, (iii) relação entre as fontes de informação, (iv) propósito do método (Nakamura et al., 2007).

Dentro da MDF, a informação tem quatro níveis de abstração da informação: (i) *sinais* são dados coletados pelos sensores que podem ter uma ou várias dimensões, (ii) *pixels* são sinais com informação agregada, (iii) *features* são dados obtidos a partir de sinais e/ou *pixels* e (iv) *símbolos* são a representação de uma decisão (Nakamura et al., 2007). "A MDF lida com praticamente três níveis de abstração: mensuração, caracterização e decisão"(Nakamura et al., 2007, p. 8). Em (Nakamura et al., 2007), encontra-se quatro tipos de métodos de MDF baseados no nível de abstração:

- **Fusão de baixo nível**: Nesse grupo colocamos todos os métodos que trabalham com sinais diretamente. Esses métodos trabalham com dados ruidosos que tratam e expressam melhor o comportamento do ambiente monitorado.
- **Fusão de médio nível**: São os métodos que usam *pixels* e *features* para gerar outras *features*, que serão usadas como entrada de outros métodos de análise e decisão.
- **Fusão de alto nível**: São métodos que combinam símbolos para gerar símbolos mais acurados ou gerar uma visão global do cenário.
- **Fusão de multinível**: Esses métodos combinam dados de diferentes níveis de abstração e geram símbolos ou resultados de qualquer nível de abstração.

Em Dasarathy (1997), são classificados os sistemas de MDF segundo os níveis de abstração das entradas e saídas dos métodos (Nakamura et al., 2007). E se classificam em cinco categorias os métodos de fusão:

- *Entrada Dados - Saída Dados (DAI-DAO)*: os métodos recebem sinais e retornam sinais ou *pixels* que têm como resultado informação mais precisa, acurada e/ou com algum tratamento de ruídos. Exemplo: filtros.
- *Entrada Dados - Saída Feature (DAI-FEO)*: são métodos que usam os sinais e *pixels* para inferir outras *features* (ou propriedades) das entidades (Nakamura et al., 2007). Exemplo: estimar a velocidade, a partir da posição do objeto.
- *Entrada Feature - Saída Feature (FEI-FEO)*: são métodos que refinam e/ou aprimoram *features* ou que extraem novas *features*. Exemplo: estimar a aceleração a partir da velocidade.
- *Entrada Feature - Saída Decisão (FEI-DEO)*: Métodos que recebem *features* e retornam um símbolo indicando um estado ou uma decisão a ser tomada.
- *Entrada Decisão - Saída Decisão (DEI-DEO)*: São métodos que, ao receber um conjunto de símbolos, informam um contexto global ou novas decisões que foram obtidas do conjunto de entrada.

Outro critério para classificar os métodos de MDF é a relação entre as fontes que proveem os dados. As três relações entre as fontes de dados: (i) complementar, (ii) redundante, (iii) cooperativa (Nakamura et al., 2007; Durrant-Whyte, 1990). **Métodos complementares** obtêm informações de fontes independentes ligadas a informação ou entidades distintas. **Métodos redundantes** usam dados equivalentes que provêm de fontes independentes relacionadas a uma mesma informação ou entidade. **Métodos cooperativos** processam dados distintos de fontes independentes que estão relacionados à mesma fonte de informação ou entidade (Nakamura et al., 2007).

Por último, os métodos de fusão também são classificados pelo tipo de operação aplicada na entrada do algoritmo (Nakamura et al., 2007). As classificações segundo

o tipo de operação são: (i) inferência, (ii) estimação, (iii) classificação, (iii) *feature-maps*, (iv) sensores abstratos, (v) agregação e (vi) compressão (Nakamura et al., 2007).

Os métodos de inferência são aqueles que, a partir das variáveis do ambiente, têm como resultado uma decisão, um símbolo. Esses métodos partem de uma situação (um estado) verdadeiro para outro estado verdadeiro. Exemplo: Inferência Bayesiana, Inferência Dempster-Shafer e Lógica Fuzzy (Nakamura et al., 2007).

Métodos de estimação usam *pixels* ou *features* para chegar em *features* que não podem ser obtidas diretamente pelos dados originais. São métodos de nível médio de abstração que podem ser classificados como *Entrada Dados - Saída Feature* ou *Entrada Feature - Saída Feature*. Exemplos: filtro de média móvel, mínimos quadrados, filtros de Kalman (Nakamura et al., 2007).

Métodos *Feature-maps* transformam os conjuntos de *features* em um novo conjunto de *features* que permitem fazer análises diferentes sobre o ambiente diferente. Diferentemente dos métodos de estimação, eles criam conjuntos de *features* e não fazem um acréscimo ao conjunto de entrada. Exemplos: *Ocuppation Grid* (Elfes, 1989), *Network Scan* (Zhao et al., 2002).

Métodos de agregação resolvem as redundâncias geradas pelos sensores e, assim, resumem os dados no resultado (Nakamura et al., 2007). Seu principal objetivo é remover dados que não acrescentam conhecimento e, com isso, economizar gastos com o envio. Funções de agregação são muito utilizadas na linguagem SQL para a geração de relatórios.

Métodos de compressão exploram as correlações entre os dados esparsos e os sensores, impedindo comunicações desnecessárias, exceto quando envolvem disseminação dos dados (Nakamura et al., 2007). Exemplos: *Distributed Source Coding* (Xiong et al., 2004), *Coding by ordering* (Petrovic et al., 2003).

Neste trabalho, a MDF é usada como método de síntese ou agregação, a fim de diminuir os custos de envio de dados para servir de fonte para outras aplicações que vão tomar a decisão (Nakamura et al., 2007).

3 TRABALHOS RELACIONADOS

O desafio de aproveitar a quantidade massiva de dados produzidos pelas rede de sensores e produzir informação com valor agregado tem promovido a publicação de vários trabalhos focados em tratar a grande quantidade de dados produzidos pelas *RSSFs* . Alguns desses trabalhos foram publicados com diferentes técnicas de análises de dados, com características diferentes no resultado produzido. Para cada trabalho relacionado, descrevemos essas características gerais, as similaridades e as diferenças entre cada um deles, além de descrevermos o propósito desse trabalho. Exemplos de trabalhos de propósito semelhante são: Singhal and Garimella (2012), Bicocchi et al. (2012), de Farias et al. (2012b), Farias et al. (2014), Dziengel et al. (2016), Safia et al. (2016), Aquino et al. (2016) e (Kumar and Chaurasiya, 2018).

No primeiro trabalho relacionado, os autores propuseram uma nova função de fusão de sensores baseada em média (Singhal and Garimella, 2012). Os autores reivindicaram que a saída do sensor inclua uma incerteza na leitura (δy), e a observação do sensor seja um intervalo (y - δy, y + δy). A estimativa do intervalo de leitura é feita adicionando dois valores tolerância, pela esquerda e pela direita do valor puro, podendo ser iguais ou não. Esse trabalho relacionado propõe dois algoritmos (Singhal and Garimella, 2012): (i) para o caso em que o intervalo de medida seja o mesmo para todos os sensores; e (ii) outro para o caso em que os intervalos de medida variem de sensor para sensor. Apesar do trabalho propor um sistema de MDF Singhal and Garimella (2012), o procedimento de fusão proposto não considera o cenário de várias aplicações com requisitos diversos. *Hercules* é um algoritmo de MDF que considera a execução num cenário de várias aplicações com requisitos diversos, detectando as massas de dados.

Bicocchi et al. (2012) apresenta um algoritmo para permitir a auto-organização da rede de sensores, uma forma de partição virtual que corresponda a regiões esparsas caracterizadas por padrões de sensoriamento similares. Bicocchi et al. (2012) propôs um algoritmo que permite uma agregação distribuída de dados sensoreados para aproveitar lugares por região, que resulta na modelagem de uma rede de sensores como uma coleção de macrossensores virtuais, cada um associado a uma

região caracterizada do ambiente monitorado. Em cada região, cada sensor físico tem a avaliação local dos dados agregados sobre a região e se é possível atuar como ponto de acesso naquela região. A abordagem de macrossensores de Bicocchi et al. (2012) inclui três principais aspectos: (i) um algoritmo para formação de regiões auto-organizáveis, para dividir de forma otimizada uma rede de sensores em regiões esparsas, cada uma caracterizada pelos padrões específicos do ambiente nos dados sensoreados; (ii) algoritmos de agregação localizada na rede para prover para cada sensor na região com informação agregada sobre o estado global da região; e (iii) soluções inovadoras e peculiares para autoadaptar-se às situações dinâmicas e transições. Essa solução assegura que ambas regiões e informações agregadas sempre vão refletir o atual estado da rede e do ambiente monitorado. O *Hercules* tem dois principais aspectos: (i) um algoritmo de calibração dos parâmetros do algoritmo de agregação e (ii) o algoritmo de agregação de informação. Por outro lado, o algoritmo de calibração do *Hercules* não define um algoritmo de formação de regiões, sendo assim uma escolha de implementação usar ou não.

A abordagem ViMS (*Virtual Macro sensores*) (Bicocchi et al., 2012) propõe que a rede de sensores possa se auto-organize em regiões com padrões de sensoriamentos similares e promova a agregação de dados por regiões, e cada região esparsa monitorada por sensores. Em comparação com o nosso propósito, a abordagem ViMS (Bicocchi et al., 2012) pode ser muito efetiva em suportar vários usuários. Outro aspecto similar é a capacidade de facilitar a coleção de dados em larga escala e reforçar atividades no reconhecimento de fenômenos e situações. Porém, diferentemente da abordagem ViMS (Bicocchi et al., 2012), nosso trabalho propõe um sistema de MDF, enquanto Bicocchi et al. (2012) propõem um algoritmo de auto-organização para particionar a rede em regiões esparsas por similaridade de padrões. Enquanto a abordagem ViMS (Bicocchi et al., 2012) demanda particionar a rede em padrões de similaridade, nosso trabalho não precisa particionar a rede para reconhecer fenômenos e status dos eventos, ela também provê o conhecimento sobre a localidade de vários fenômenos ocupando o lugar da área monitorada. Nós consideramos nosso trabalho uma ferramenta que pode ser usada para particionar a rede, pelo uso do resultado como entrada num algoritmo de particionamento de rede. Os trabalhos

seguintes lidam com o desafio de executar algoritmos de fusão de dados para várias aplicações numa RSSF.

Primeiro, apresentamos o trabalho de Farias et al. (2012b) que modificou o filtro de média móvel (MAF) para calcular os dados sensoriados de forma diferente, de acordo com a importância desse dado para a aplicação. O trabalho de Farias et al. (2012b) apresenta um filtro de média móvel melhorado, sua principal ideia é ponderar o conjunto de dados para expressar os requerimentos (intervalos de dados, taxas de dados e estados) de aplicações diferentes. A desvantagem dessa abordagem é a necessidade do conhecimento sobre os requerimentos da aplicação. Em cenários nos quais o conjunto de aplicações muda rapidamente (como em cidades inteligentes onde as aplicações podem pertencer a qualquer um e ser implantadas a qualquer hora), EMAF fica inviabilizado, necessitando ser constantemente reconfigurado. Como EMAF de Farias et al. (2012b) usa dados não tratados (leituras com pouco nível de abstração), dados de alto nível de abstração (como decisões) não são propriamente manipulados. Entretanto, em trabalhos posteriores, os mesmos autores de EMAF de Farias et al. (2012b) apresentaram métodos de fusão de alto nível de abstração. Em Farias et al. (2014), os autores adaptaram alguns métodos de fusão existentes para executar fusão de informação em dados para várias aplicações nas RSSFs. Eles propuseram os seguintes métodos de fusão de informação: (i) Inferência Bayesiana Aprimorada (IBA), (ii) Inferência Dempster-Shafer Aprimorada (IDSA)e (iii) Média de Tolerância a Falhas Aprimorada (MTFA). A IBA formaliza a combinação de evidências de acordo com as regras de teoria de probabilidade bayesiana para cada aplicação Bayes (1991). A IBA representa a hipótese de que a aplicação "Y"vai ter um comportamento determinado dado o resultado da aplicação "X", por considerar o conjunto de estados de cada aplicação isoladamente. Na IDSA, cada aplicação tem seu próprio conjunto de hipóteses, que representa o comportamento da aplicação. IDSA infere por meio do método de inferência Dempster-Shafer sobre as condições de aplicações isoladas e considera a premissa que ambas as aplicações vão para um estado simultaneamente. MTFA calcula os intervalos de dados de acordo com os requerimentos de cada aplicação, aplicando o tradicional método de intervalo de tolerância a falhas. Então, MTFA produz a combinação de cada intervalo calculado.

Apesar de lidar com informação para várias aplicações, os trabalhos de Farias et al. (2014) e de Farias et al. (2012b) apresentam uma restrição significativa por precisar de uma pré-configuração com os requisitos de cada aplicação na rede. Enquanto EMAF de Farias et al. (2012b) precisa conhecer os requisitos da aplicação para propriamente ponderar o conjunto de dados, IBA, IDSA e MTFA Farias et al. (2014) precisam inferir os estados para cada aplicação para apresentar a decisão que integra várias aplicações. Em cenários no quais o conjunto de aplicações em execução muda constantemente (como as que podem ser implantadas a qualquer tempo em cidades inteligentes), esse trabalho se torna inviável. *Hercules* introduz um novo método de fusão de informação que não depende dos requisitos da aplicação, desde que o ambiente monitorado seja avaliado e usado para calibrar os parâmetros do *Hercules*. Musílek et al. (2015) propõem E-BACH: hierarquia de agrupamentos baseada em entropia para RSSF para desenvolver uma hierarquia de agrupamentos para formar uma RSSF heterogênea com nós operando com taxas de coletas de dados determinadas pela sua entropia. Musílek et al. (2015) propõem um novo método que usa a qualidade dos dados com o objetivo primário da otimização da rede. Baseado no conceito de entropia de dados, o trabalho (Musílek et al., 2015) desenvolve uma rede hierárquica heterogênea de acordo com o potencial ganho de informação (Musílek et al., 2015). Os autores (Musílek et al., 2015) declaram que isso pode ser usado para salvar energia por meio da redução de taxas de amostragem dos nós na localidade com pequena variação de variáveis monitoradas ou alta correlação com outros nós. As medidas de entropia e suas aproximações previamente delineadas podem ser usadas para identificar agrupamentos de sensores Musílek et al. (2015). Dependendo de medidas particulares usadas, resultados do procedimento proposto por Musílek et al. (2015) podem desenvolver uma hierarquia de agrupamentos de acordo com o conteúdo da informação e identificar grupos de sensores que proveem dados similares Musílek et al. (2015). Para esse propósito, medidas de entropia e suas aproximações podem ser tratadas como distâncias (informações distantes) Musílek et al. (2015) entre os dados dos sensores. Em resumo, a contribuição de Musílek et al. (2015) desenvolve um novo algoritmo de agrupamento de RSSFs baseado na entropia dos dados dos sensores. De forma similar ao propósito deste trabalho,

Musílek et al. (2015) propõem um algoritmo de processamento de dados que usa a organização dos dados(entropia) para facilitar a coleta de dados sem se preocupar com os requisitos das aplicações na rede. Porém, diferentemente de nosso propósito, o trabalho não produz uma informação global porque o agrupamento resultante não é adaptável para a coleta de dados independentemente do propósito da aplicação. Ao contrário do propósito de Musílek et al. (2015), nós consideramos o propósito desse trabalho - *Hercules* - como uma ferramenta que não agrupa a rede mas pode ajudar no processo de agrupamento da rede.

Dziengel et al. (2016) propõem um classificador baseado em sistemas de detecção de eventos distribuídos que consiste em dois frameworks: (i) um framework para cálculo, que provê um modelo de classificação e permite um cálculo teórico num grupo de treinamento, e (ii) o framework de detecção de eventos distribuídos que, subsequentemente, aplica ao modelo de classificação para avaliar, filtrar e classificar eventos na rede. Também, Dziengel et al. (2016) calculam a aplicação cerca, que é motivada pela necessidade de um sistema de proteção para regiões abertas e extensas como construções, aeroportos ou locais de acesso não permitido. A cerca equipada com esse sistema de detecção de eventos distribuídos sem fio pode agregar e calcular dados de sensores em vários pontos de medida para classificar e detectar eventos Dziengel et al. (2016). O sistema de vigilância de cerca investigada segue a bordagem orientada pelos dados. Então, geralmente o objetivo é resolver o problema de classificação de distinguir eventos treinados no local de construção cerca com a RSSF (Dziengel et al., 2016). Dziengel et al. (2016) introduzem uma solução de processamento na rede que usa método de compressão de dados baseados em sistema de padrão de reconhecimento. O propósito da abordagem na rede é um sistema de detecção de eventos distribuídos que envolve vários sensores para obter uma visão ampliada de um evento. Em princípio, ele observa eventos de perspectivas diferentes com sensores localizados em posições diferentes na fence para complementar a outra. Um modelo integrado de classificação é usado para definir uma diversidade de descrições de eventos. O processo de detecção de eventos colaborativos junta os dados disponíveis com a rede. De forma semelhante ao nosso trabalho, o trabalho de Dziengel et al. (2016) não apresenta somente um método de cálculo de infor-

mação local, que pode carecer da habilidade de observar eventos em sua totalidade. Também, de fórum semelhante ao nosso trabalho, Dziengel et al. (2016) usam características descritivas para distinguir eventos diferentes. E também, similar ao nosso trabalho, o trabalho de Dziengel et al. (2016) precisa de um conjunto de treinamento para inicializar os classificadores (Classificador de Naive Bayes e classificador de vizinhança K), enquanto *Hercules* precisa calibrar os seus parâmetros que variarão dependendo da configuração da rede e do ambiente monitorado. Safia et al. (2016) propõem um algoritmo distribuído para detectar fenômenos, como incêndios ou derramamento de óleo (ou gases tóxicos), em um ambiente monitorado onde os sensores são dispositivos móveis e o fenômeno é dinâmico (movendo-se, crescendo ou encolhendo). Os autores assumem que o ambiente monitorado não tem um servidor central para coletar e agregar os dados dos sensores. No algoritmo Safia et al. (2016), os sensores organizam-se em grupos nos quais um deles é escolhido como o líder que representa aquele grupo. Os sensores de cada grupo enviam os dados para o sensor líder, que agrega os dados coletados e detecta o fenômeno local (na área abrangida pelos sensores do grupo). Então, baseado na ordem dos identificadores dos líderes, cada líder envia a informação do fenômeno local detectado para o próximo líder, nessa ordem, que agrega a informação e envia para o próximo líder e assim por diante. O último líder na ordem da cadeia agrega a informação de todos os fenômenos detectados para descobrir o fenômeno local. Além disso, o artigo Safia et al. (2016) propõe dois algoritmos de eleição de líderes: baseado nas informações do evento global e baseado nas informações dos fenômenos locais. O trabalho Safia et al. (2016) propõe uma técnica de otimização para reduzir os custos de energia do envio de informações locais. Nesse método, a informação de fenômenos locais enviada entre líderes é resumida por uma representação de um fecho convexo e, assim sendo, reduz a quantidade de dados transmitidos entre líderes.

O trabalho Safia et al. (2016) apresenta alguma similaridade com o propósito do nosso trabalho por ser um algoritmo distribuído de detecção de eventos em um ambiente monitorado. Porém, diferentemente do nosso trabalho, o trabalho Safia et al. (2016) tem as seguintes prerrogativas: (i) para Safia et al. (2016), sensores têm o mesmo poder de processamento, tempo de bateria, armazenamento e alcance de

comunicação na fase inicial; e (ii) sensores têm o conhecimento do "intervalo padrão dos valores". Relativo às duas afirmações acima citadas de Safia et al. (2016), a principal diferença de *Hercules* é que não tem um pré-requisito para os sensores na fase inicial. *Hercules* não precisa especificar as características de seus sensores, somente precisa se adaptar às características do ambiente monitorado e da rede.

Em Aquino et al. (2016) é apresentado um algoritmo distribuído que usa fusão da informação para agregar dados sensoreados usando estatística descritiva para fazer uma análise de picos, identificar mudanças no ambiente monitorado. O *Hepheatus* calcula a média, curtose e assimetria do conjunto de entrada e, a partir dessas medidas, ele decide ou não separar o conjunto em dois usando a média como parâmetro de divisão dos conjuntos. Após isso, ele aplica o mesmo procedimento nos subconjuntos. No nosso trabalho, ao invés de usar no método de fusão a estatística descritiva, usamos cálculo numérico para fazer análises de máximos e mínimos, picos e vales da curva de frequência somente uma vez e não reaplica nos subconjuntos gerados. Com isso, *Hercules* pode fazer uma análise mais voltada para o cenário porque se adapta às curvas de frequências de cada cenário.

Em Kumar and Chaurasiya (2018), é apresentado um algoritmo de fusão da informação que aplica dados da rede a redes neurais e lógica fuzzy para identificar os eventos. O algoritmo proposto por Kumar and Chaurasiya (2018) usa uma rede neural de cinco camadas para apurar informação da rede de sensores sem fio para selecionar o melhor caminho para o envio de dados dos dispositivos de coleta para os dispositivos de análise e fusão de informação. Como o nosso trabalho, ele busca se adequar ao ambiente monitorado para operar melhor. Ao contrário do nosso trabalho, ele somente provê a melhor escolha de caminhos dentro dos *clusters* da rede e o *Hercules* se adapta ao ambiente, mas também trata o volume de dados coletado pelos dispositivos para reduzir os dados que serão enviados para os dispositivos de análise.

Após essa análise, os sistemas podem ser agrupados em quatro conjuntos (Aquino et al., 2016): (i) G1 os algoritmos que funcionam para somente uma única aplicação; (ii) G2 os algoritmos que funcionam para várias aplicações, mas que precisam conhe-

cer os requisitos; (iii) G3 os algoritmos que não precisam conhecer os requisitos das aplicações, mas seu algoritmo somente produz um agrupamento na rede; (iv) G4 os algoritmos que funcionam para várias aplicações, sem conhecer os requisitos e os resultados podem ser usados para tomar decisões; (v) G5 os algoritmos que funcionam para várias aplicações, que se adaptam aos requisitos do ambiente e os resultados podem ser usados para tomar decisões. Na tabela 1 é apresentada a relação.

Grupo	**Trabalho Relacionado**
G1	(Singhal and Garimella, 2012)
	(Dziengel et al., 2016)
G2	(de Farias et al., 2012b)
	(Farias et al., 2014)
	(Safia et al., 2016)
G3	(Bicocchi et al., 2012)
	(Musílek et al., 2015)
G4	(Aquino et al., 2016)
	(Kumar and Chaurasiya, 2018)
G5	*Hercules*

Tabela 1: Trabalhos relacionados separados em grupos

4 PROPOSTA

Neste capítulo é apresentado *Hercules*, um algoritmo de fusão de dados que analisa a curva de frequência das leituras coletadas pelos sensores para separar as massas de dados em conjuntos unimodais. *Hercules* é um algoritmo que se baseia na análise de funções de frequência. Este capítulo será dividido nas seguintes sessões: na sessão 4.1 é descrita uma visão global do *Hercules* , na sessão 4.2 são apresentadas todas as estruturas de dados que são usadas no *Hercules*, na sessão 4.3 é descrita a construção da função de média, na sessão 4.4 é descrita a seleção de pontos para separação em conjuntos unimodais e na sessão 4.5 é descrito um algoritmo para otimizar os parâmetros do *Hercules* antes de seu uso para monitoramento.

4.1 VISÃO GLOBAL DO *HERCULES*

O *Hercules* é um algoritmo que analisa a curva de frequência das leituras coletadas pelos sensores, usando os máximos e mínimos de funções para separar as leituras em conjuntos unimodais. Constrói-se uma curva de frequência usando os dados coletados pelos sensores e usando um número arbitrário de classes. Essa curva é um conjunto de pontos (x, y), em que x é ponto central da classe e y é frequência de dados deste mesmo intervalo da classe. Após construir a curva de frequência, constrói-se uma função de médias que consiste em dividir a curva de frequência em intervalos e obter o ponto médio ponderado pelas frequências e a média simples das frequências desse intervalo. Esse procedimento, usando número de intervalos adequado, remove ruídos da curva e mantém a curvatura original. Com essas duas características, usa-se a análise dos máximos e mínimos da função de médias para encontrar as fronteiras entre as massas de dados.

O *Hercules* tem como resultado particionar o conjunto original em subconjuntos unimodais, de forma que cada um represente um único fenômeno dentro do ambiente monitorado. O processo de particionamento consiste em escolher um valor X e separar o conjunto original em todos os valores menores ou iguais a X e todos os valores maiores de X, garantindo que os conjuntos gerados sejam unimodais, dos

quais serão calculadas métricas que serão enviadas para a aplicação de análise de cenário. Esses subconjuntos unimodais, contendo somente os dados relacionados a um único evento, permitem caracterizar os eventos que estão acontecendo no ambiente monitorado.

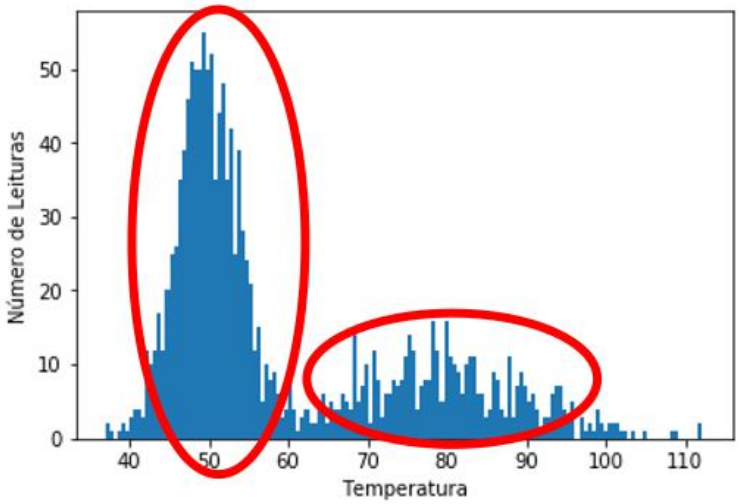

Figura 3: Gráfico que apresenta duas concentrações com dispersões diferentes.

Na figura (3) é apresentado o histograma da curva de frequência das leituras de temperatura coletadas no ambiente monitorado. No intervalo $[40, 60]$, as frequências crescem conforme se aproximam do valor 50, e decrescem quanto mais perto dos limites do intervalo. Ou seja, a maior parte das leituras advindas dos sensores estão dentro desse intervalo e existe um pico de frequência próximo ao valor 50. Massas de dados são esses intervalos em que um crescimento de frequências é seguido de um decrescimento. Em $[40, 60]$ e $[60, 100]$ nota-se duas massas de dados com intervalos de tamanho diferentes e valores máximos que se distinguem.

Em cada caso, os ruídos que são decorrentes da amostra coletada são máximos locais que se destacam, mas não são o centro da tendência. Os ruídos atrapalham ao contabilizar os máximos das amostras, porque do ponto de vista local são máximos, mas do ponto de vista global são somente um pico isolado que não representa as massas de dados. Para impedir que esses ruídos atrapalhem a análise, a função de médias é construída fazendo, assim, uma versão sem ruídos da curva de frequências.

O *Hercules* foi projetado para, a partir da função de médias, inferir quantas

massas de dados, intervalos em que as frequências de leituras se concentram, e encontrar os melhores pontos para particionar essas massas em subconjuntos. Os pontos principais desse processo são: (i) a construção da **curva de frequências**; (ii) a construção da **função de médias**; (iii) a classificação dos pontos da função de médias para encontrar os pontos de cisão(pontos mínimos da função); (iv) o uso dos **pontos de cisão** para particionar o conjunto das leituras coletadas em subconjuntos com uma única massa de dados; (v) e para cada subconjunto gerar métricas que sejam o resultado da agregação (a média, por exemplo). Esses subconjuntos representam fenômenos distintos no ambiente monitorado. Neste trabalho, é usado como exemplo o monitoramento de temperatura da rede de potência e da bateria de uma torre de transmissão. O objetivo é separar o conjunto de leituras advindos da rede e da bateria em dois conjuntos de dados: da bateria e da rede de potência, quando esses não tiverem a mesma temperatura. Desses fenômenos isolados, geram-se métricas como média, variância e curtose que são enviadas para a aplicação de análise.

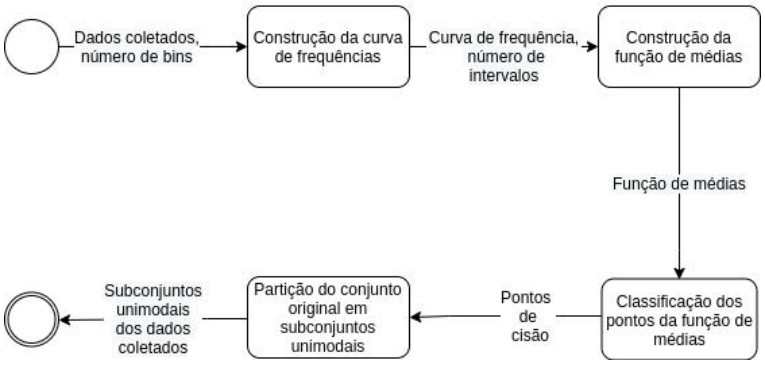

Figura 4: Diagrama de etapas do *Hercules*

Na figura 4, apresenta-se a relação entre as etapas do *Hercules*. O algoritmo tem duas constantes arbitrárias: **o número de classes** da curva de frequência e **o número de intervalos** da função de média. As *classes da curva de frequência* são intervalos de tamanho fixo entre os valores extremos da amostra. O **número de classes** é o parâmetro que ajusta a curva de frequência à granularidade dos dados. Cada classe tem a sua frequência associada ao número de valores que estão entre os limites do intervalo. A curva de frequência é necessária para analisar e formalizar

as massas de dados independentemente de serem esparsos ou não.

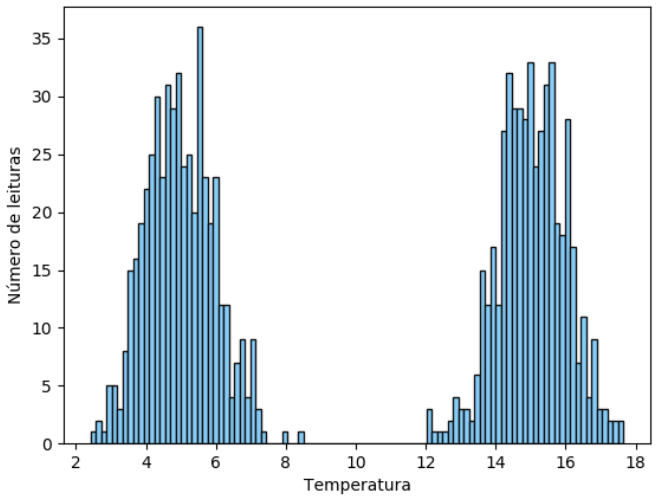

Figura 5: Um exemplo de curva de frequência

O **número de intervalos da função de média** é o parâmetro que informa o nível de detalhamento que será usado ao criar a função de médias. A relação entre a função de médias e curva de frequência é que intervalos de tamanho fixo na curva de frequência são transformados em pontos na função de médias. Quanto mais intervalos, mais semelhante à curva de frequência será a função de média. Se o número de intervalos for igual ao número de classes da curva de frequência, teremos uma cópia da curva de frequência. O principal objetivo é que, por meio da média, os ruídos da curva de frequências não interfiram. Por isso, é preciso pré-visualizar os dados que serão monitorados para assim realizar a melhor escolha de número de intervalos. A função de médias tem o objetivo de impedir que os ruídos atrapalhem a análise e que seja possível selecionar os pontos de cisão de forma eficiente.

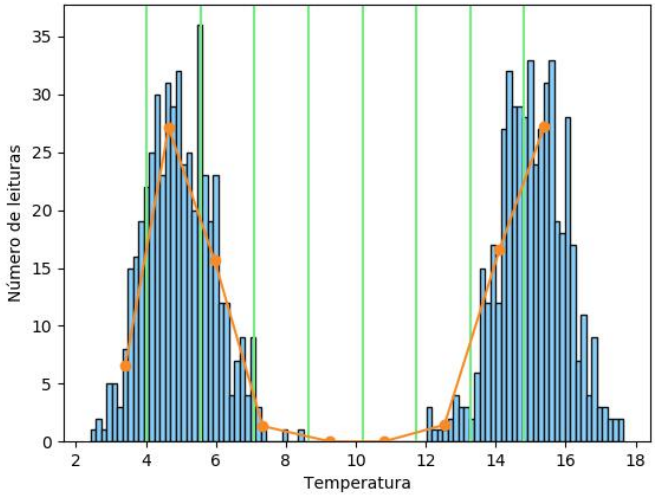

Figura 6: Um exemplo de curva de frequência em azul, intervalos em verde e a função de médias em laranja.

Os **pontos de cisão** são pontos escolhidos para particionar o conjunto de dados original em subconjuntos unimodais. Eles são escolhidos após a classificação dos pontos da função de médias por intervalo. Cada subconjunto unimodal representa um único evento no ambiente monitorado. Para entender o que significa um evento, define como:

Cada subconjunto unimodal é a massa de dados que representa um evento. Um evento pode estar representar um único ou um conjunto de fenômenos que estão acontecendo no ambiente monitorado e gerando dados que serão capturados pelos sensores.

Os dados são coletados em intervalos de tempo que chamamos de janelas. Ao final de cada janela se executa o processo de fusão de dados(em uma janela de fusão). Na figura 7 exemplifica-se como funciona o fluxo de trabalho da rede.

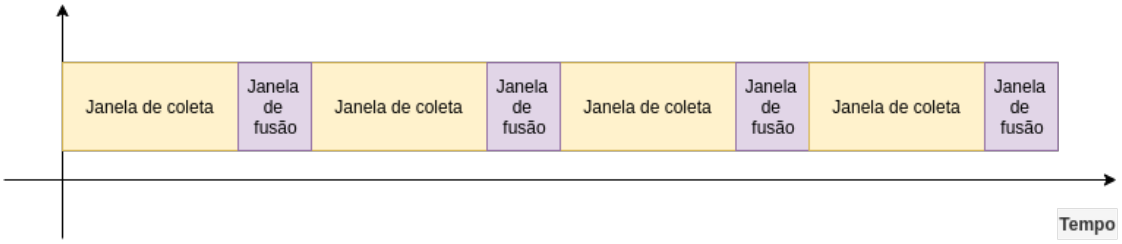

Figura 7: Exemplo do fluxo de trabalho da rede exibindo momentos em que a rede somente coleta dados e outros em que ela reúne os dados e gera os resultados

A primeira etapa do processo de fusão é a construção da curva de frequência. Essa curva de frequência vai aglutinar dados de diversas fontes que podem ser dados homogêneos ou não. Dados homogêneos, que são da mesma grandeza física, são o caso base do estudo. É possível trabalhar com dados de diversas grandezas físicas, mas isso não será abordado no trabalho. Na figura (3), temos um exemplo de histograma homogêneo que usa a leitura de dados de temperatura de diversas fontes.

Após a construção da função de médias, classifica-se cada ponto em três atributos: picos (máximos locais), vales (mínimos locais) e passagens (pontos que não são nem máximos, nem mínimos). Após a classificação, os vales são pontos em que se realizará a cisão ou divisão do conjunto de dados original. Os vales, que representam os intervalos de menor concentração de dados, são os pontos que serão usados para dividir o conjunto original em conjuntos unimodais.

Todas essas etapas fazem parte do procedimento principal e acontecem dentro da janela de fusão. Antes de efetivamente iniciar o fluxo completo de monitoramento e análise, devemos fazer uma pré-análise dos dados que serão monitorados e, assim, fazer a melhor escolha dos parâmetros (número de classes e número de intervalos). Esse procedimento é chamado de *calibração do método*.

4.2 ESTRUTURAS DE DADOS DO *HERCULES*

As estruturas de dados usadas no *Hercules* são: LLS (Lista de Leituras dos Sensores), $OCF(n)$ (Lista de Ordenadas da Curva de Frequência), $DCF(n)$ (Lista de Dados da Curva de Frequência), $OFM(m)$ (Lista de Ordenadas da Função de Médias com m intervalos), $DFM(m)$ (Lista de Dados da Função de Médias com m intervalos), $CFM(m)$ (Lista de Classificação dos m Pontos da Função de Média), LPC (Lista de Pontos de Cisão), $CSEI$ (Lista de Conjunto de Subconjunto de Eventos Isolados).

O LLS armazena os dados recebidos de todos sensores os dados chegam no formato de uma cadeia de caracteres com duas casas decimais e são convertidas para um número na memória. O $OCF(n)$ armazena as frequências das classes da curva de frequência, que são construídas tendo como base os dados armazenados em LLS. O $DCF(n)$ armazena os pontos médios das classes da curva de frequência, que são construídas tendo como base os dados armazenados em LLS. O $DFM(m)$ tem os pontos médios de todos m intervalos em que a curva de frequência foi dividida, que são obtidos de $DCF(n)$. O $OFM(m)$ lista as médias de todos m intervalos em que a curva de frequência foi dividida, que são obtidos de $DCF(n)$. O $CFM(m)$ armazena as classificações dos pontos da função de médias, que são inicializadas todas em zero. O LPC armazena todos os pontos de cisão, que são os pontos com classificação de vale. $CSEI$ é a estrutura que armazena os subconjuntos gerados pela divisão ou cisão do conjunto original, usando os pontos de cisão armazenados em CPC.

4.3 FUNÇÃO DE MÉDIAS

A função de médias é construída usando a curva de frequência. Construir a função de média consiste em particionar a curva de frequência em intervalos e calcular a média simples da frequência nesse intervalo e a média ponderada pela frequência para cada intervalo. Seja $H = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$, a curva de frequência de N classes gerada a partir dos dados da amostra S. A função das médias de K

intervalos $\chi = [(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2), \ldots, (\bar{x}_K, \bar{y}_K)]$ tem seus pares ordenados definidos como:

$$\bar{x}_k = \frac{\sum_{i=kn}^{(k+1)n} y_i x_i}{\sum_{i=kn}^{(k+1)n} y_i}, \forall k \in \{0, 1, 2, \ldots, K-1\} \tag{4.1}$$

$$\bar{y}_k = \frac{\sum_{i=kn}^{(k+1)n} y_i}{n}, \forall k \in \{0, 1, 2, \ldots, K-1\} \tag{4.2}$$

em que, $n = \frac{N}{K}$ é o tamanho dos intervalos em que a curva de frequência foi dividida. A curva de frequência tem um tamanho inteiro N, por isso todas as escolhas de K devem ser divisores de N para garantir um n inteiro ou $n = \lfloor n \rfloor$. A média do intervalo foi escolhida por representar um centro de gravidade (DeGroot and Schervish, 2012) do intervalo selecionado, mas nada impede que a mediana ou a moda sejam usadas no lugar da média.

Cada ponto na função de médias representa um intervalo na curva de frequência. Se houver uma sequência ascendente (ou decrescente), isso significa que, em média, a curva de frequência esteja crescendo (ou decrescendo) conforme aparece na figura (6). A função de médias indica as tendências de crescimento e de decrescimento da curva de frequências sem que os ruídos interfiram. Dependendo da amostragem, uma curva de frequências com muitas classes tem um comportamento mais irregular com muitos máximos locais que não são relevantes para a análise da curva de frequência como um todo. Por isso a função de médias usa as médias dos intervalos como uma forma de suavizar a função e de seus ruídos(máximos locais irrelevantes) não interferirem na análise.

Os máximos locais da função de médias são os centros dos eventos, porque o máximo local na curva de frequência implica que todos os intervalos ao redor têm picos inferiores, ou seja, a concentração de frequência nesses intervalos é menor (DeGroot and Schervish, 2012). Por outro lado, os mínimos locais da função de média representam os intervalos com poucas frequências entre dois picos. O vale

simboliza que a frequência de dados está diminuindo e, depois desse ponto, ela retoma o crescimento. Essa conjuntura significa que temos duas massas de dados distintas: a esquerda do vale cuja frequência diminui e a direita do vale que está começando logo após o vale.

Algoritmo 1: Algoritmo da função de médias

Entrada: Ordenadas da curva de frequência: OCF,
Dados da curva de frequência: DCF,
Número de intervalos: K

Saída: Ordenadas da Função de médias: OFM,
Dados da Função de médias: DFM

1 **para** $index \in \{0, 1, 2, 3, \cdots, K-2, K-1\}$ **faça**
2 $\quad OFM(index) \leftarrow media(OCF, index * n, (index + 1) * n)$
3 $\quad DFM(index) \leftarrow mediaPonderada(DCF, OCF, index * n, (index + 1) * n)$
4 **fim**

4.4 IDENTIFICAÇÃO DE PONTOS DE CISÃO

Os pontos da função de média são os representantes de intervalos da curva de frequência. Com eles, pode-se inferir quais são as regiões com a maior frequência de leituras e como identificar as fronteiras entre duas massas de dados. Para isso, definimos os máximos e mínimos na função de médias:

Definição 1. *Ponto de máximo local ou pico é o elemento do conjunto X de índice i para qual vale a seguinte condição:*

$$X(i-1) < X(i) \quad e \quad X(i) > X(i+1). \tag{4.3}$$

Definição 2. *Ponto de mínimo local ou vale é o elemento do conjunto X de índice i para qual vale a seguinte condição:*

$$X(i-1) > X(i) \quad e \quad X(i) < X(i+1). \tag{4.4}$$

Pontos que não se encaixam nessas duas definições são classificados como *passagem*. Os pontos que estão nos extremos do vetor não satisfazem a condição e são classificados como passagens. Mesmo que sejam classificados como picos ou vales, por satisfazer parte das condições, isso não vai influenciar o resultado. Em determinados casos poderia até comprometê-lo. Caso um extremo seja definido como um pico, isso será irrelevante porque picos não interferem no resultado final e são ignorados após a classificação. Por outro lado, caso um extremo seja definido como vale, criará uma cisão que vai dividir em dois conjuntos, um com a maior parte dos dados e outro com uma sobra que não representa um conjunto em si. Por não satisfazerem as definições de picos e vales, os extremos devem ser classificados como passagens e também porque, definidos assim, não influenciam a análise.

Os picos indicam a presença de, pelo menos, uma massa de dados, onde, cada uma é um evento monitorado que deve ser isolado num conjunto unimodal. Quando um ponto é classificado como vale, ele informa que no conjunto de dados é um conjunto com mais de uma massa de dados, tendo mais de um evento sendo monitorado. O vale é o melhor ponto para isolar duas massas de dados por indicar o ponto médio mais baixo entre esses dois picos.

Todos os pontos começam classificados como passagem, ou como 0, no CFM. Ao percorrer o OFM, o algoritmo analisa se a função está crescendo ou decrescendo. A classificação dos pontos anteriores pode mudar dependendo do ponto atual.

Quando a função está crescendo (linha **??**), verifica-se a classificação do ponto anterior. Se o ponto anterior for um vale (linha 3) ou uma passagem, esse ponto atual é um pico (linha 4) e sem alterar os pontos anteriores. Caso o ponto anterior seja um pico (linha 5), classifica-se o ponto anterior como uma passagem (linha 6), porque se encontrou um ponto menor que o anterior e esse ponto maior se transforma no pico (linha 7).

Durante o decrescimento da função (linha 8) temos um comportamento quase simétrico. Se o ponto anterior for um pico (linha 11) ou uma passagem, esse ponto é um vale (linha 12). No caso do ponto anterior ser um vale (linha 13), o ponto anterior é reclassificado como uma passagem e o ponto atual é classificado como um

Algoritmo 2: Classificar os pontos da função de médias

Entrada: Lista de frequências da função de médias: OFM,
Número de intervalos da função de média: K

Saída: Classificação dos pontos da função de médias: CFM

```
1  para index ∈ {1, 2, 3, ··· , K − 2, K − 1} faça
2      senão se OFM(index − 1) <= OFM(index) então
3          se CFM(index − 1) < 1 então
4              CFM(index) ← 1
5          senão se CFM(index − 1) = 1 então
6              CFM(index − 1) ← 0
7              CFM(index) ← 1
8      se OFM(index − 1) > OFM(index) então
9          se CFM(index − 1) > −1 então
10             CFM(index) ← −1
11         senão se CFM(index − 1) == −1 então
12             CFM(index − 1) ← 0
13             CFM(index) ← −1
14     index ← index + 1
15 fim
16 CFM(0) ← 0
17 CFM(K − 1) ← 0
```

novo vale.

Após iterar em todos os pontos da função, temos que lidar com problemas que a classificação dos extremos possa gerar. Caso os extremos sejam classificados como vales, geram conjuntos inúteis que não representam um evento. Caso os extremos sejam classificados como o pico, não serão relevantes para as próximas etapas. Para evitar os casos de extremos que sejam classificados como vales ou picos, devem ser classificados automaticamente como passagens (linhas 16 e 17).

Após classificar os pontos da função de médias e selecionar todos os vales e

armazená-los no CPC, o próximo passo é particionar o CLS usando os pontos do CPC e armazenar em $CSEI$. Após tudo isso, o resultado é a criação de vários conjuntos unimodais em $CSEI$, em que cada um representa um único evento. A partir do $CSEI$ é necessário gerar métricas, que variam com os interesses das aplicações que processam essas métricas, sobre os conjuntos unimodais, que serão enviadas para a aplicação de análise de dados.

4.5 CALIBRAÇÃO DOS PARÂMETROS

Hercules considera a granularidade, dispersão e esparsidade por seus dois parâmetros (número de classes da curva de frequência e número de intervalos da função de médias). Em cada caso que o *Hercules* for aplicado como algoritmo de fusão, deve ser feita uma escolha adequada aos dados que serão analisados e sempre temos uma única escolha que é a melhor. Para isso, levam-se em consideração os seguintes fatores: (i) acurácia, (ii) o consumo de memória e (iii) o tempo de execução.

Os parâmetros influenciam diretamente na construção de estruturas de dados internas ao método: o número de classes da curva de frequência e o número de intervalos da função de médias, com isso interferem no consumo de memória e tempo de tempo. Não obstante, é preciso assegurar a melhor acurácia possível. O número de classes está informando como o método deve tratar a granularidade e a dispersão dos dados e o número de intervalos informa o quanto de informação deve ser coletada da curva de frequência.

4.5.1 Estruturas de dados do método de calibração

As estruturas de dados usadas no método de calibração do *Hercules* são: CDT (Conjunto de Dados para Teste), LC (Lista de Classes que a curva de frequência pode ter), $LIPC$ (Lista de Intervalos Por Classes da função de médias), FO (Função Otimizadora), NOC (Número Ótimo de Classes) e NOI (Número Ótimo de Intervalos).

O LC é a estrutura de dados que armazena todas as possíveis quantidades de

classes que a curva de frequência pode admitir. Seus limites inferiores e superiores são 1 e o número total de dados lidos, mas esses limites nunca são úteis e somente servem como limitadores. A $LIPC$ é a estrutura que armazena a quantidade de intervalos que podem ser assumidos por classe, mas na prática para a classe i são armazenados seus divisores. A FO é a estrutura de dados bidimensional que armazena a acurácia de cada iteração sendo indexada pelo número i de classes para a curva de frequência e o número j de intervalos para a função de médias. O NOC é a estrutura de dados que armazena a quantidade de classes ótima que foi encontrada e NOI é a estrutura que armazena quantidade ótima de intervalos para a função de média que foi encontrada.

4.5.2 Método de calibração

O método de calibração é um estudo prévio do ambiente monitorado e/ou a coleta de dados do mesmo. Com dados coletados ou randomicamente gerados por um modelo, usa-se esses dados como um grupo de controle para encontrar os melhores parâmetros para otimizar a execução do *Hercules*. A comparação entre cada parâmetro listado em LC e $LIPC$ é feita pelo cálculo da acurácia. A acurácia é o percentual de verdadeiros positivos somados com falsos positivos no grupo de teste, mas essa medida pode ser variável dependendo do seu critério. O resultado do processo é o ponto ótimo, definido como:

Definição 3. *O ponto ótimo é o ponto que procura maximizar a acurácia, minimizar o número de classes da curva de frequência e minimizar o número de intervalos da função de médias.*

A minimização dos parâmetros se deve à importância de minimizar o tamanho das estruturas de dados dentro do sensor. É uma restrição que informa dentro dos pontos com máxima acurácia qual é o mais econômico. O procedimento é listar quais as possibilidades de classes da curva de frequência e depois quantos intervalos seriam possíveis considerando a quantidade de classes da curva de frequência. Após a construção dessa lista, iterando sobre os parâmetros listados e calculando a acurácia do método sobre o conjunto de dados, temos a função objetivo para ser otimizada.

Como o domínio é pequeno, foi usada a força bruta como algoritmo de busca do ponto ideal.

Algoritmo 3: Algoritmo de calibração de parâmetros

Entrada: Conjunto de dados de teste: CDT,
Lista de Classes: LC,
Lista de intervalos por Classe: LIPC

Saída: Número ótimo de classes: NOC,
Número ótimo de intervalos: NOI

```
1 FO ← estrutura de dados bidimensional para i ∈ LC faça
2 |   para  j ∈ LICP(i) faça
3 |   |   FO[i][j] ← acurácia ao aplicar o Hercules  no CDT com i classes e j
  |   |   intervalos
4 |   fim
5 fim
6 (NOC, NOI) ← buscar o ponto ótimo em FO
```

A calibração deve ser feita necessariamente antes de implementar o algoritmo para um ambiente monitorado, visando saber quais são os parâmetros de execução. O método de calibração também deve ser aplicado em momentos em que se altera quaisquer elementos do ambiente monitorado. Isso envolve fatores do ambiente como a organização da rede de sensores e os elementos do próprio ambiente. Se o número ou a disposição dos sensores forem alterados a ponto de afetar a granularidade dos dados e/ou a sua dispersão, é preciso recalibrar e achar um novo par de parâmetro que otimize os métodos. Se novos fenômenos ou eventos forem adicionados no ambiente monitorado, deve se refazer a calibração porque não é garantido que os parâmetros continuem ótimos. Por fim, o método de calibração deve ser aplicado sempre que o contexto do ambiente monitorado for alterado e isso influenciará os parâmetros ótimos que otimizam a execução do *Hercules*.

5 IMPLEMENTAÇÃO

Este capítulo tem o objetivo de descrever a implementação do *Hercules* nas plataformas de experimentos, dividido nas seguintes seções: 5.1 plataforma de desenvolvimento, 5.2 implementação de protótipo, 5.3 protocolo de sincronização, 5.4 protocolo de comunicação.

5.1 PLATAFORMA DE DESENVOLVIMENTO

O algoritmo do *Hercules* foi desenvolvido usando a linguagem de programação C e a plataforma *Contiki*. *Contiki* é um sistema operacional que foi desenvolvido para dar suporte ao desenvolvimento de IoT (*Internet Of Things*). O *Contiki* usa o emulador *COOJA* para testar e emular os códigos desenvolvidos para IoT. Esse ambiente é desenvolvido para emular dispositivos que têm $8KB$ de memória *RAM* e $16MHz$ de *clock* processamento como o *Zolertia Z1* que foi usado nos experimentos. Isso é possível pela compilação do *Contiki* para plataformas nativas como bibliotecas compartilhadas e carregadas como bibliotecas no Java usando *Java Native Interfaces* (JNI) (Strübe, 2016).

A figura 8 apresenta a pilha de protocolos de rede do *Contiki*. A pilha tem quatro grandes camadas: (i) a camada Radio que é responsável pelo envio de fato; (ii) a camada RDC(*Radio Duty Cycling*) que é responsável por poupar energia durante os ciclos de inatividade; (iii) a camada MAC(*Medium Access Control*) que é responsável por retransmitir pacotes para a camada RDC em caso de colisão de pacotes; (iv) a camada de Rede que envolve não somente os protocolos de comunicação em rede, mas as camadas de aplicação e roteamento.

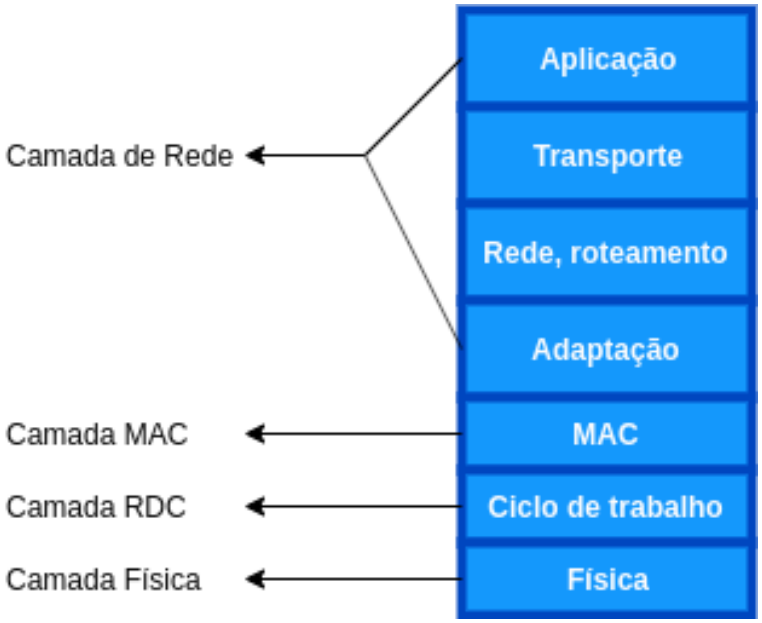

Figura 8: Pilha de protocolos do Contiki

5.2 IMPLEMENTAÇÃO DO PROTÓTIPO

Hercules é implementado como o procedimento principal do nó fusor. Durante uma janela de coleta, o nó fusor recebe de diversos nós amostras coletadas no ambiente monitorado e armazena esses dados num buffer que fica disponível para ser preenchido durante toda a janela de coleta de dados. Ao terminar uma janela de coleta de dados, inicia-se uma janela de fusão de dados e sua primeira ação é bloquear qualquer acesso ao buffer de dados coletados. Tendo bloqueado o acesso, o *Hercules* é executado e seu resultado é enviado para o nó sorvedouro.

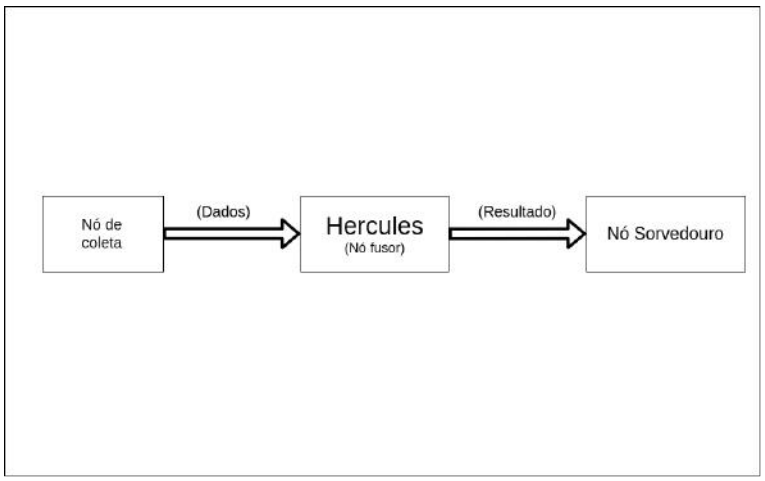

Figura 9: Fluxo de dados do *Hercules*

5.3 PROTOCOLO DE SINCRONIZAÇÃO

A sincronização é feita usando janelas de sincronização. O algoritmo tem duas janelas: (i) janela de coleta é o período que os nós coletores vão enviar dados coletados para o nó fusor; (ii) a janela de fusão é o período em que os dados na janela de coleta anterior são processados e um resultado é enviado para as aplicações servidas. Durante a janela de fusão, o nó fusor ativa uma trava (*lock*) que impede que qualquer dado que seja enviado dos sensores que coletam seja adicionado ao vetor de conjuntos analisados (Pacheco, 2011; Herlihy and Shavit, 2011).

5.4 PROTOCOLO DE COMUNICAÇÃO

Essa seção descreve o protocolo de comunicação é usado na emulação do *Hercules*. A comunicação deve permitir que o *Hercules* seja capaz de executar todas as suas funções dentro da RSSF. As funções que serão executadas pelo *Hercules* são: (i) enviar os dados coletados dos nós coletores para o nó fusor; (ii) executar; (iii) enviar os dados resultantes para o nó sorvedouro. A tabela 2 descreve as mensagens do *Hercules* com suas Fontes, Destinos e Identificadores.

Mensagem	Identificador	Fonte	Destino
DADOS_COLETADOS	c	Nó coletor	Nó fusor
INFORMACAO_LOCAL	H	Nó Fusor	Nó sorvedouro

Tabela 2: Mensagens do *Hercules*

A DADOS_COLETADOS é a mensagem enviada pelos nós coletores com as leituras que foram feitas no ambiente monitorado e enviadas para o nó fusor. A DADOS_COLETADOS contém os seguintes elementos: (i) o identificador da mensagem "c"; (ii) o número k de dados coletados pelo nó coletor; e (iii) k dados que foram coletados pelo nó durante a janela de fusão.

A INFORMACAO_LOCAL é a mensagem enviada pelo nó fusor para o nó sorvedouro a fim de prover os dados resultantes da execução do *Hercules* . A INFORMACAO_LOCAL contém os seguintes elementos: (i) o identificador da mensagem "H"; (ii) o número s de dados enviados na mensagem; e (iii) s dados que foram o resultado da execução do *Hercules*. Essa mensagem em específico pode mudar dependendo da aplicação de consumo. Nessa implementação, optou-se por enviar somente a média dos conjuntos identificados pelo *Hercules*. Na seção, sugere-se que um dos trabalhos futuros seja indicar quais as melhores métricas que devem ser enviadas para abranger a maior quantidade de métodos de análise e decisão.

Na tabela 3, há dois exemplos de mensagens, uma para cada tipo de mensagem: DADOS_COLETADOS e INFORMACAO_LOCAL. Os campos das mensagens são separados por ":". O exemplo de mensagem de DADOS_COLETADOS tem os seguintes valores: (i) seu identificador "c"; (ii) seguido do número 5 que informa o número de leituras coletadas; e (iii) "19,20,22,21,18" que são os cinco dados coletados separados por ",".

Mensagem	Exemplos
DADOS_COLETADOS	"c:5:19,20,22,21,18"
INFORMACAO_LOCAL	"H:2:21,55"

Tabela 3: Exemplos de mensagens

O exemplo de INFORMACAO_LOCAL tem os seguintes valores: (i) seu identificador "H"; (ii) seguido do número 2 que informa o número de conjuntos identificados pelo *Hercules*; e (iii) "21,55"que são médias do *Hercules* separados por ",".

6 EXPERIMENTOS

Neste capítulo, apresenta-se um estudo de aplicações em *Smart Grid* (Denholm et al., 2005) e esse modelo é usado para experimentar o *Hercules* num contexto de várias aplicações. O objetivo desse experimento é mostrar que o *Hercules* pode ser executado como um método de fusão de informação em um ambiente monitorado. Este capítulo se divide nas seguintes seções: (i) a seção 6.1 que descreve o estudo de caso em *Smart Grid*; (ii) na seção 6.3 descrevemos os ambientes de testes usados; (iii) na seção 6.2, as métricas usadas para avaliar os resultados são descritas; (iv) na seção 6.4 é descrito o modelo de energia dos sensores; e (v) na seção 6.5 são apresentados os resultados obtidos nos experimentos.

6.1 DESCRIÇÃO DO CASO

Seguindo o cenário usado em Aquino et al. (2016), o *Hercules* foi aplicado ao contexto de aplicações em *Smart Grid* Denholm et al. (2005). Esse cenário apresenta duas aplicações que fazem o monitoramento de uma torre que compõe um *Smart Grid* e seus dois componentes principais: a rede de potência e a bateria. A rede de potência são os cabos e todos os dispositivos que são responsáveis pela transmissão de energia elétrica em alta voltagem. A bateria fica na base da torre e serve como um dispositivo de apoio à rede de potência.

Cada uma das aplicações monitora a temperatura de cada componente, sendo uma aplicação para a rede de potência e outra específica para a bateria. As duas aplicações consomem o mesmo tipo de informação (a temperatura), mas cada um tem o seu intervalo respectivo. Para que a rede de potência não tenha danos causados pelo superaquecimento, a temperatura deve estar no intervalo de $40°C$ a $65°C$. No caso de temperatura entre $65°C$ e $75°C$, temos a indicação de dano em potencial para rede potência. A bateria tem a sua temperatura normal de operação no intervalo de $40°C$ a $144°C$ e acima de $144°C$ indica perigo de dano na bateria. Esses valores vão mudar de acordo com o modelo de temperatura apresentado em Schlapfer and Mancarella (2011). Temperatura no intervalo de $65°C$ a $75°C$ indica um

comportamento normal para a bateria, mas indica um comportamento de perigo de dano na rede de potência e é esse o ponto principal desse estudo de caso.

Os intervalos de operação de cada aplicação permitem criar uma lista de estados do ambiente monitorados, a partir dos estados de cada aplicação. As duas aplicações têm status de funcionamento normal e um status em que o dispositivo está em perigo de dano, assim podemos criar quatro estados do ambiente monitorado: C1 em que tanto a aplicação da rede potência quanto a aplicação da bateria estão em funcionamento normal; C2 em que a aplicação de rede potência corre perigo de dano e a bateria permanece em funcionamento normal; C3 em que a aplicação de rede potência tem funcionamento normal e a bateria corre perigo de dano; e por fim, C4 em que ambas as aplicações indicam que os dispositivos correm perigo de dano. Essa relação é resumida na tabela 4.

Estados	**Rede de Potência**	& bateria
C1	Normal	Normal
C2	Normal	Em Perigo
C3	Em perigo	Normal
C4	Em perigo	Em perigo

Tabela 4: Tabela de estados do ambiente monitorado e os status dos dispositivos monitorados.

No estado C1, a rede de potência está na condição normal e isso gera leituras de temperaturas entre $40°C$ e $65°C$, enquanto a bateria está na condição normal e tem leituras de $40°C$ a $144°C$. No estado C2, a rede de potência está em perigo de dano com leituras de temperatura entre $65°C$ e $75°C$ e a bateria está na condição normal e tem leituras entre $40°C$ e $144°C$. No estado C3, a rede de potência está na condição normal e isso gera leituras de temperaturas entre $40°C$ e $65°C$ e a bateria está em perigo de dano e tem leituras acima de $144°C$. Por fim, no estado C4, a rede de potência está em perigo de dano com leituras de temperatura entre $65°C$ e $75°C$ e a bateria está em perigo de dano e tem leituras acima de $144°C$. Os intervalos de cada estado foram resumidos na tabela 5.

Estados	**Intervalos de Funcionamento**
C1	$40°C$ a $65°C$ na rede de potência; $40°C$ a $144°C$ na bateria.
C2	$65°C$ a $75°C$ na rede de potência; $40°C$ a $144°C$ na bateria.
C3	$40°C$ a $65°C$ na rede de potência; Acima de $144°C$ na bateria.
C4	$65°C$ a $75°C$ na rede de potência; Acima de $144°C$ na bateria.

Tabela 5: Tabela de estados do ambiente monitorado

6.2 MÉTRICAS

Para executar os experimentos e estimar os resultados, usamos as seguintes métricas: (i) a **quantidade de energia consumida**; (ii) o **número de bytes ocupados na memória flash**; (iii) o **número de bytes livres na memória RAM**; (iv) e a **acurácia do método**.

A **quantidade de energia consumida** é o total de energia consumida pelo nó fusor da rede. O **número de bytes ocupados na memória flash** é a quantidade de bytes que o programa ocupa ao ser armazenado na memória flash do nó fusor. O **número de bytes livres na memória RAM** é quantos bytes livres são deixados ao carregar o programa na memória do nó fusor durante a execução do procedimento de fusão.

A **acurácia** (ACC) é o percentual de acertos da execução de uma bateria de testes. Uma bateria de teste é um conjunto de 100 conjuntos de entrada para serem executados quando se sabe o resultado final. Sabendo o resultado esperado ao executar um único teste, esse mesmo resultado pode ser classificado de quatro formas: Falsos Negativos (FN) que indicam a detecção correta de que o estado é de risco (a bateria não é considerada em um ambiente de risco quando é um ambiente

de risco); Falsos Positivos (FP) que indicam a detecção incorreta de que o estado é de risco (a bateria é considerada em um ambiente de risco quando não está num ambiente de risco); Verdadeiros Positivos (VP) que indicam a detecção correta de que o estado não é de risco (a bateria não é considerada em um ambiente de risco quando não é um ambiente de risco) e Verdadeiros Negativos (VN) que indicam a detecção correta de que o estado é de risco (a bateria é considerada num ambiente de risco quando está num ambiente de risco). A acurácia é a soma de *Verdadeiros Positivos* e *Falsos Negativos* sobre o total de testes, como podemos ver em:

$$ACC = \frac{VP + FN}{VP + FP + VN + FN}. \tag{6.1}$$

6.3 CONFIGURAÇÃO DE AMBIENTE

Para os testes do *Hercules* foram usados dois ambientes de execução em comparação com o *EMAF* e o *Hepheastus*: (i) usando uma implementação do método em *Python*; (ii) e outra no sistema operacional *Contiki*. A implementação do método em *Python* teve o intuito de testar a acurácia dele com os outros métodos e a implementação no sistema *Contiki* para testar o consumo da memória e da energia.

O ambiente *Python* é uma máquina Intel i5 de 64 bits com 4GB de memória. No sistema *Contiki*, emulamos um ambiente com 21 nós (20 para a coleta de dados e outro para o método de fusão). Todos os nós, independentemente de serem nós fusores ou coletores, são um sensor *Zolertia Z1* com um microcontrolador *MSP430F2617*, $8Kb$ de memória RAM, $92Kb$ de memória flash e um processador RISC de 16 bits e $16MHz$. Na figura 10, temos a distribuição dos nós no ambiente emulado.

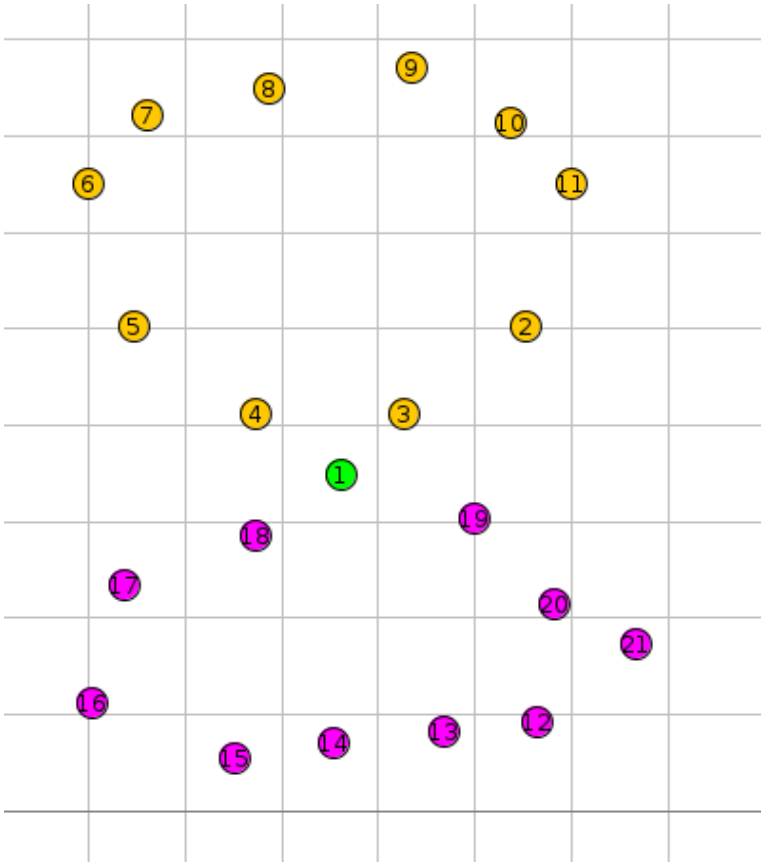

Figura 10: Dispersão dos nós no ambiente emulado de 100mX100m

6.4 MODELO DE ENERGIA

Nesta seção, vamos apresentar o modelo de consumo de energia que usamos para calcular o consumo de energia do *Hercules*. O *Contiki/Cooja* é usado como plataforma de emulação do ambiente da rede de sensores e, no desenvolvimento dos testes, foi utilizada a biblioteca *Energest* e o plugin *Powertrace*. O *Energest* é uma bilbioteca que provê uma estimativa de energia consumida, com baixo custo, que é usada em dispositivos *IoT* com restrição de recursos. O *Powertrace* é um plugin do *Contiki* que facilita o desenvolvimento e o monitoramento do consumo de energia de cada sensor da rede durante a emulação. Com o uso de ambos, temos um modelo de energia que vamos usar nos nossos testes do *Hercules*.

Cada dispositivo tem duas unidades principais: unidade de processamento, que é responsável pela execução do método e acesso aos sensores, e unidade de comuni-

cação, que é responsável por enviar e receber pacotes de mensagem. Cada unidade pode estar em dois estados de consumo de energia: ativo, em que a unidade tem um consumo constante de energia, inativo, em que o consumo de energia é bem menor do que o estado ativo e considerado desprezível (sendo considerado como zero). Usando o modelo, consideram-se quatro estados do dispositivo: *CPU* em que o dispositivo tem a unidade de processamento ativa; *LPM* (*Low-Power Mode*) em que o dispositivo tem modo de baixo consumo tanto para processamento quanto para o envio de pacotes; *TRANSMIT* em que a unidade de comunicação está ativa enviando dados para a rede; e *LISTEN* em que a unidade de comunicação está ativa recebendo dados da rede.

Com o estado *LPM* , o consumo de energia é desprezível e considerado 0 (zero). O consumo total de energia é a soma das energias consumidas nos períodos em que o dispositivo ficou nos outros três estados *CPU*, *TRANSMIT*, *LISTEN*. Para fins de análise, podemos afirmar que consumo de energia é a soma do consumo da unidade de processamento mais o da unidade de comunicação.

6.5 DESCRIÇÃO DOS RESULTADOS

Nesta seçã, é descrito cada conjunto de testes sobre o *Hercules*. Os testes que serão descritos a seguir são: a quantidade de energia consumida na seção 6.5.1, o número de bytes ocupados na memória flash e o número de bytes livres na memória RAM na seção 6.5.2, e, a acurácia do método na seção 6.5.3.

6.5.1 Quantidade de energia consumida

Nesta seção, são apresentados os resultados dos experimentos que estimaram o consumo de energia do *Hercules* em função do tempo em que o algoritmo foi executado. Nessa rede foi emulado o comportamento de três tipos de sensores: (i) os dispositivos que seriam responsáveis por coletar dados da rede de potência e enviar para o dispositivo fusor; (ii) os dispositivos que coletariam dados da bateria e enviariam para o dispositivo fusor; e (iii) os dispositivos que seriam responsáveis

por receber os dados, executar o algoritmo de fusão e enviar para as aplicações de decisão. Nesse experimento foram comparados os resultados do *Hercules* com a implementação do *Hepheastus* (Aquino et al., 2016) no mesmo ambiente; para facilitar a coleta de dados, usamos o plugin do *Powertrace*.

Foram executadas 60 simulações (30 para cada algoritmo), cada simulação durava 60 minutos e as janelas de coleta duravam um minuto. Após cada janela de coleta, foi executado o método de fusão (*Hercules* ou *Hepheastus*) nos dados coletados e, usando o *Powertrace*, coletamos o consumo de energia do nó fusor a cada um minuto. Depois da execução de cada simulação, estimamos a média e o intervalo de confiança do consumo de energia de cada algoritmo.

Algoritmo	**Média**	**Intervalo de confiança**
Hepheastus	1,627234 Watts	+/- 0,003682
Hercules	1,358005 Watts	+/- 0,010421

Tabela 6: Tabela de consumo durante os testes em Watts.

O *Zolertia Z1*, dispositivo escolhido como sensor no experimento, pode ser alimentado por duas baterias do tipo AA. Foram usadas como exemplo duas baterias *Duracell Ultra Power AA MX1500*, como alimentação do sensor. Na tabela 7, apresentamos o tempo de duração dos sensores com a alimentação das baterias *Duracell Ultra Power AA MX1500*.

Algoritmo	**Duração (em minutos)**
Hepheastus	287,60 (4 horas e 47 minutos)
Hercules	344,62 (5 horas e 44 minutos)

Tabela 7: Tabela de tempo de duração das pilhas AA em minutos

6.5.2 Quantidade de memória consumida

Nesta seção, são apresentados os resultados dos experimentos que calcularam o consumo de memória flash e memória *RAM* do método *Hercules* em comparação com

o *Hepheastus* (Aquino et al., 2016). Nesse experimento, comparamos a quantidade de *bytes* livres na memória *flash* e a quantidade de *bytes* livres na memória *RAM*. O percentual de *bytes* livres na memória *flash* foi calculado usando o arquivo de saída do compilador do *COOJA* e medida a quantidade de memória *flash* restante. Para calcular o percentual de *bytes* livres na memória *RAM* foi usado o mesmo procedimento, mas desconsiderando estruturas de tamanho variável. O dispositivo escolhido para executar o experimento é um *Zolertia Z1* e esse sensor dispõe de $8KB$ *RAM* (*Random Access Memory* - Memória de acesso randômico) e de $92KB$ de memória flash. Na tabela 8, é apresentado o consumo de memória de ambos os algoritmos.

Algoritmo	**bytes da RAM livres**	**bytes em uso na Flash**
Hepheastus	1670 bytes	50396 bytes
Hercules	1934 bytes	46292 bytes

Tabela 8: Tabela de consumo de memória dos códigos compilados no Contiki

6.5.3 Teste de Acurácia

Esta seção compara os resultados da acurácia dos experimentos simulados do *Hercules* em comparação com *Hepheastus* (Aquino et al., 2016) e ao MAF (de Farias et al., 2012a). Esse teste foi executado na platforma *Jupyter Notebook*, que usa a linguagem *Python* Martelli et al. (2005). O objetivo era testar a capacidade de acerto do método usando conjunto de testes amostras aleatórias baseadas em modelos do ambiente apresentado no estudo de caso na seção 6.1.

Cada estado do ambiente monitorado é modelado como variável aleatória. Cada aplicação tem uma variável aleatória que aproxima o comportamento do objeto monitorado. As temperaturas da bateria e da rede de potência terão uma distribuição normal, essa foi uma escolha arbitrária feita neste trabalho para os testes. O *Hercules* não tem uma restrição sobre a origem ou o modelo dos dados que recebe como entrada. Os parâmetros para as variáveis normais podem ser conferidos na tabela abaixo.

Estado	Rede de Potência		Bateria	
	Média	Desvio Padrão	Média	Desvio Padrão
C1	50,0	4,0	92,0	18,0
C2	85,0	2,0	92,0	18,0
C3	50,0	4,0	155,0	2,0
C4	85,0	2,0	155,0	2,0

Tabela 9: Parâmetros das variáveis aleatórias em cada estado.

Foram gerados 100 conjuntos de conjuntos amostrais, usando em cada conjunto de conjuntos uma semente de geração de números aleatórios diferentes. Em cada conjunto há 100 conjuntos amostrais gerados a partir da semente e com 100 amostras para cada um dos casos. Foram comparados os três métodos: (i) *MAF* (de Farias et al., 2012a); (ii) *Hepheastus* (Aquino et al., 2016); e (iii) *Hercules*. A acurácia foi calculada em dois níveis: o nível de interpretação do estado do ambiente e o nível de interpretação do estado da aplicação. O nível de interpretação do ambiente está relacionado a qual estado que o conjunto amostral representa e como foi avaliado pelo método. A avaliação do resultado considera, para cada aplicação, qual seria o resultado esperado e qual foi o valor avaliado pelo método. Após rodar o método em todos os lotes e tendo estimado a acurácia em todos os conjuntos de conjuntos, foi calculado o intervalo de confiança em cada caso. O resultado da acurácia está na tabela 12, o resultado de acurácia da aplicação da rede de potência está na tabela 10 e o resultado da acurácia da aplicação da bateria está na tabela 11.

	MAF		Hepheastus		Hercules	
Caso:	Acurácia	Intervalo	Acurácia	Intervalo	Acurácia	Intervalo
C1	50.0 %	+/-0.0 %	99.9579 %	+/-0.0018 %	99.9158 %	+/-0.0027 %
C2	50.0 %	+/-0.0 %	99.9728 %	+/-0.0017 %	99.9505 %	+/-0.0021 %
C3	50.0 %	+/-0.0 %	99.9604 %	+/-0.0019 %	99.9183 %	+/-0.0029 %
C4	50.0 %	+/-0.0 %	99.9653 %	+/-0.0018 %	99.9406 %	+/-0.0021 %

Tabela 10: Intervalos de confiança e estimativa da acurácia da aplicação da rede de potência

	MAF		Hepheastus		Hercules	
Caso:	Acurácia	Intervalo	Acurácia	Intervalo	Acurácia	Intervalo
C1	50.0 %	+/-0.0 %	89.7723 %	+/-0.0286 %	99.4431 %	+/-0.0069 %
C2	50.0 %	+/-0.0 %	89.5767 %	+/-0.0228 %	99.4827 %	+/-0.0067 %
C3	50.0 %	+/-0.0 %	89.7104 %	+/-0.0333 %	99.4332 %	+/-0.0072 %
C4	50.0 %	+/-0.0 %	89.7327 %	+/-0.0281 %	99.4901 %	+/-0.0073 %

Tabela 11: Intervalos de confiança e estimativa da acurácia da aplicação da bateria

	MAF		Hepheastus		Hercules	
Caso:	Acurácia	Intervalo	Acurácia	Intervalo	Acurácia	Intervalo
C1	75.0 %	+/-0.0 %	89.7723 %	+/-0.0289 %	99.4257 %	+/-0.0069 %
C2	25.0 %	+/-0.0 %	89.5693 %	+/-0.0228 %	99.4505 %	+/-0.0066 %
C3	25.0 %	+/-0.0 %	99.9728 %	+/-0.0015 %	99.9827 %	+/-0.0012 %
C4	75.0 %	+/-0.0 %	99.9802 %	+/-0.0015 %	99.9604 %	+/-0.0018 %

Tabela 12: Intervalos de confiança e estimativa da acurácia do experimento

7 CONCLUSÃO

Este trabalho apresenta um algoritmo que não depende dos requisitos do ambiente monitorado e nem das aplicações que consomem os dados. Assim como *Hepheastus* Aquino et al. (2016), o algoritmo transforma o ambiente monitorado em um repositório de dados a ser consumido pelas aplicações. O *Hercules* aceita várias aplicações, porque ele se adapta para o ambiente monitorado e não para as aplicações. Por ser sensível ao contexto, *Hercules* tem um procedimento de calibração de parâmetros que otimizem a análise dos dados coletados. Após a coleta, constrói uma curva de frequência dos dados. A partir da curva de frequência, constrói uma função de médias que suaviza a curva de frequência dando mais destaque aos pontos com maior concentração de frequências do que aos pontos máximos, que são consequências de erro na amostragem. Com os mínimos locais da função de médias, o conjunto de dados original é particionado em conjuntos unimodais que representam um evento isolado no ambiente monitorado, sendo possível, assim, gerar qualquer tipo de métrica sobre esses conjuntos.

Sendo adaptado para o ambiente monitorado, qualquer mudança no ambiente monitorado ou na rede de sensores exige uma recalibração. Ao calibrar parâmetros, condiciona-se o método a uma determinada quantidade de dados coletados pela rede e a entropia dos eventos interagindo entre si. Caso essas duas variáveis mudem, é necessário uma recalibração do método para assegurar a acurácia e a representação do ambiente monitorado.

Neste trabalho, o *Hercules* teve sua acurácia testada usando duas aplicações para SmartGrid (Monitoramento de sobrecarga de redes de potência - Monitoramento da OLPM e da bateria Aquino et al. (2016)). O experimento, como foi usado em Aquino et al. (2016), usa para a acurácia as duas aplicações de SmartGrid que indicam o *Hercules* como sendo capaz de tratar com várias aplicações sem conhecer previamente os requisitos. Nosso experimento mostrou que *Hercules* tem uma acurácia maior que o *Hepheastus* Aquino et al. (2016). Executar um experimento com várias aplicações é necessário para medir a acurácia do *Hercules* em diferenciar vários fenômenos.

Hercules se adapta à curva de frequência dos dados, o que permite que ele identifique com muito mais clareza os eventos que acontecem no ambiente monitorado. Este trabalho define um método de calibração simples, mas ele é fundamental para o potencial de *Hercules* ser bem aproveitado.

As principais contribuições do trabalho são: (i) *Hercules* é orientado pelo ambiente monitorado, sendo somente necessário calibrar simples parâmetros para a adaptação e execução do método; (ii) *Hercules* considera a quantidade de dados sensoreados e a sua concentração na curva de frequência; (iii) *Hercules* é dividido em dois procedimentos principais: método de calibração dos parâmetros e o procedimento de análise dos dados.

7.1 TRABALHOS FUTUROS

Nesta seção, apresentamos os trabalhos futuros do *Hercules*. Esses trabalhos representam desafios que serão descritos nesta seção. Como trabalhos futuros, temos os seguintes pontos: (i) aprimorar o método de calibração; (ii) definir uma saída genérica que atenda os diversos métodos de análise de dados.

O atual procedimento de calibração tem a premissa de maximizar a acurácia do método de forma a minimizar os parâmetros. Isso assegura que os parâmetros escolhidos estejam interpretando o formato do ambiente e minimizando o consumo de memória do sensor de fusão. Porém o método não foi pensado para um sensor, mas para um dispositivo com poder computacional para utilizar a força bruta, testando todas as combinações possíveis, e estimar acurácia em cada uma para selecionar o ponto ótimo. Uma contribuição para o trabalho seria uma conclusão de quais combinações de parâmetros podem ser ignoradas levando em conta fatores como a entropia e a quantidade de dados. Isso permitindo uma inicialização e reconfiguração mais rápida, possibilitando que o método seja usado em ambientes que mudam mais constantemente.

Após a execução de *Hercules*, definimos como saída do algoritmo os conjuntos unimodais que são subconjuntos do conjunto de dados coletados. Porém nunca

vamos enviar esses conjuntos unimodais como resultado para reduzir o consumo de energia na transmissão de dados. E assim, fica a critério da implementação quais seriam as métricas para a síntese dos dados. Um trabalho futuro seria definir as melhores métricas para atender à maioria dos métodos de decisão que as aplicações usam. Isso permitirá uma melhor integração entre as aplicações de decisão com o *Hercules*, mas também levará em consideração os algoritmos que fazem previsão de cenários sobre o ambiente monitorado a partir de dados previamente coletados.

REFERÊNCIAS

I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002. ISSN 1389-1286. doi: https://doi.org/10.1016/S1389-1286(01)00302-4. URL http://www.sciencedirect.com/science/article/pii/S1389128601003024.

Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

G. Aquino, L. Pirmez, C. M. de Farias, F. C. Delicato, and P. F. Pires. Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 59–66, July 2016.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

Seema Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 3, pages 1713–1723 vol.3, March 2003. doi: 10.1109/INFCOM.2003.1209194.

Th Bayes. An essay towards solving a problem in the doctrine of chances. 1763. *MD computing: computers in medical practice*, 8(3):157, 1991.

Nicola Bicocchi, Marco Mamei, and Franco Zambonelli. Self-organizing virtual macro sensors. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1):2, 2012.

H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl. Understanding smart cities: An integrative framework. In

2012 45th Hawaii International Conference on System Sciences, pages 2289–2297, Jan 2012. doi: 10.1109/HICSS.2012.615.

Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

Belur V Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997.

C. M. de Farias, L. Pirmez, F. C. Delicato, I. L. Dos Santos, and A. Y. Zomaya. Information fusion techniques applied to shared sensor and actuator networks. In *37th Annual IEEE Conference on Local Computer Networks*, pages 188–191, Oct 2012a. doi: 10.1109/LCN.2012.6423603.

Claudio M de Farias, Luci Pirmez, Flavia Coimbra Delicato, Igor L Dos Santos, and Albert Y Zomaya. Information fusion techniques applied to shared sensor and actuator networks. In *37th Annual IEEE Conference on Local Computer Networks*, pages 188–191. IEEE, 2012b.

Morris H DeGroot and Mark J Schervish. *Probability and statistics*. Pearson Education, 2012.

F. C. Delicato, P. F. Pires, L. Pinnez, L. Fernando, and L. F. R. da Costa. A flexible web service based architecture for wireless sensor networks. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 730–735, May 2003. doi: 10.1109/ICDCSW.2003.1203639.

Paul Denholm, Gerald L Kulcinski, and Tracey Holloway. Emissions and energy efficiency assessment of baseload wind energy systems. *Environmental science & technology*, 39(6):1903–1911, 2005.

Isabel Dietrich and Falko Dressler. On the lifetime of wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(1):5, 2009.

Hugh F Durrant-Whyte. Sensor models and multisensor integration. In *Autonomous robot vehicles*, pages 73–89. Springer, 1990.

Norman Dziengel, Martin Seiffert, Marco Ziegert, Stephan Adler, Stefan Pfeiffer, and Jochen Schiller. Deployment and evaluation of a fully applicable distributed event detection system in wireless sensor networks. *Ad Hoc Networks*, 37:160–182, 2016.

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Claudio Farias. *A FRAMEWORK FOR DEVELOPING SMART SPACE APPLICATIONS USING SHARED SENSOR NETWORKS*. PhD thesis, Universidade Federal do Rio de Janeiro, 2014.

Claudio Farias, Luci Pirmez, Flávia Delicato, Luiz Carmo, Wei Li, Albert Y Zomaya, and José N de Souza. Multisensor data fusion in shared sensor and actuator networks. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2014.

Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2013.01.010. URL http://www.sciencedirect.com/science/article/pii/S0167739X13000241. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

D. L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, Jan 1997. ISSN 0018-9219. doi: 10.1109/5.554205.

Maurice Herlihy and Nir Shavit. *The art of multiprocessor programming*. Morgan Kaufmann, 2011.

Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1):28 – 44, 2013. ISSN 1566-2535. doi: https://doi.org/10.1016/

j.inffus.2011.08.001. URL http://www.sciencedirect.com/science/article/pii/S1566253511000558.

Hermann Kopetz. *Internet of Things*, pages 307–323. Springer US, Boston, MA, 2011. ISBN 978-1-4419-8237-7. doi: 10.1007/978-1-4419-8237-7_13. URL https://doi.org/10.1007/978-1-4419-8237-7_13.

Shishupal Kumar and Vijay Kumar Chaurasiya. A multisensor data fusion strategy for path selection in internet-of-things oriented wireless sensor network (wsn). *Concurrency and Computation: Practice and Experience*, 30(18):e4477, 2018.

Douglas M Lambert and Martha C Cooper. Issues in supply chain management. *Industrial Marketing Management*, 29(1):65 – 83, 2000. ISSN 0019-8501. doi: https://doi.org/10.1016/S0019-8501(99)00113-3. URL http://www.sciencedirect.com/science/article/pii/S0019850199001133.

Alex Martelli, Anna Ravenscroft, and David Ascher. *Python cookbook*. "O'Reilly Media, Inc.", 2005.

P. Musílek, P. Krömer, and T. Barton. E-bach: Entropy-based clustering hierarchy for wireless sensor networks. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pages 231–232, Dec 2015. doi: 10.1109/WI-IAT.2015.88.

Eduardo F Nakamura, Antonio AF Loureiro, and Alejandro C Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys (CSUR)*, 39(3):9, 2007.

Peter Pacheco. *An introduction to parallel programming*. Elsevier, 2011.

Dragan Petrovic, Rahul C Shah, Kannan Ramchandran, and Jan Rabaey. Data funneling: Routing with aggregation and compression for wireless sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pages 156–162. IEEE, 2003.

Amany Abu Safia, Zaher Al Aghbari, and Ibrahim Kamel. Phenomena detection in mobile wireless sensor networks. *Journal of Network and Systems Management*, 24(1):92–115, 2016.

Carlos Roberto Sanquetta, Alexandre Behling, Ana Paula Dalla Corte, Sylvio Pellico Netto, Aurelio Lourenço Rodrigues, and Augusto Arlindo Simon. A model based on environmental factors for diameter distribution in black wattle in brazil. *PloS one*, 9(6):e100093, 2014.

Markus Schlapfer and Pierluigi Mancarella. Probabilistic modeling and simulation of transmission line temperatures under fluctuating power flows. *IEEE Transactions on Power Delivery*, 26(4):2235–2243, 2011.

F. K. Shaikh, S. Zeadally, and E. Exposito. Enabling technologies for green internet of things. *IEEE Systems Journal*, 11(2):983–994, June 2017. ISSN 1932-8184. doi: 10.1109/JSYST.2015.2415194.

Deepti Singhal and Rama Murthy Garimella. Simple median based information fusion in wireless sensor network. In *2012 International Conference on Computer Communication and Informatics*, pages 1–7. IEEE, 2012.

Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. Protocols for self-organization of a wireless sensor network. *IEEE personal communications*, 7 (5):16–27, 2000.

Moritz 'Morty' Strübe. An Introduction to Cooja. `https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja`, 2016. [Online; accessed 08-July-2019].

Andreas Wagner, Sebastian Speiser, and Andreas Harth. Semantic web technologies for a smart energy grid: Requirements and challenges. In *In proceedings of 9th International Semantic Web Conference (ISWC2010)*, pages 33–37. Citeseer, 2010.

Zixiang Xiong, Angelos D Liveris, and Samuel Cheng. Distributed source coding for sensor networks. *IEEE signal processing magazine*, 21(5):80–94, 2004.

Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

Arkady Zaslavsky, Charith Perera, and Dimitrios Georgakopoulos. Sensing as a service and big data. *arXiv preprint arXiv:1301.0159*, 2013.

Yonggang Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scans for monitoring wireless sensor networks. 2002.