UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Marcelino Mendes da Silva Neto

Um Algoritmo Distribuído de Eleição de Líder para a Internet das Coisas Móveis

São Luís
2018

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcelino Mendes da Silva Neto

Um Algoritmo Distribuído de Eleição de Líder para a Internet das Coisas Móveis

São Luís
2018

Marcelino Mendes da Silva Neto

Um Algoritmo Distribuído de Eleição de Líder para a Internet das Coisas Móveis

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de MESTRE em Ciência da Computação.

Orientador: Rafael Fernandes Lopes

Doutor em Engenharia Elétrica – UFMA

São Luís

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Silva Neto, Marcelino Mendes da.
Um Algoritmo Distribuído de Eleição de Líder para a Internet das Coisas Móveis / Marcelino Mendes da Silva Neto. - 2018.
75 f.

Orientador(a): Rafael Fernandes Lopes.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís - MA, 2018.

1. Eleição de Líder. 2. Internet das Coisas Móveis. 3. Middleware. I. Lopes, Rafael Fernandes. II. Título.

Marcelino Mendes da Silva Neto

Um Algoritmo Distribuído de Eleição de Líder para a Internet das Coisas Móveis

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Marcelino Mendes da Silva Neto e aprovada pela comissão examinadora.

Aprovada em 30 de Agosto de 2018

BANCA EXAMINADORA

Rafael Fernandes Lopes (orientador)

Doutor em Engenharia Elétrica – UFMA

Ariel Soares Teles

Doutor em Engenharia Elétrica – IFMA

Hélder Pereira Borges

Doutor em Ciência da Computação – IFMA

Aos meus pais, meus irmãos, meus amigos e meus professores.

Resumo

A Internet da Coisas (*Internet of Things* (IoT)) pode ser definida como a interação de tecnologias de diferentes áreas, tais como: computação ubíqua, protocolos, tecnologias de comunicação e dispositivos com sensores e/ou atuadores embarcados, chamados de objetos inteligentes. A interconexão de milhares de objetos endereçáveis, heterogêneos, e com conectividade de rede, permite aos mesmos coletarem e compartilharem dados contribuindo para melhorar a vida das pessoas. Em alguns casos, os objetos possuem restrições de memória e processamento e conectividade com redes de médio e longo alcance, precisando assim de um *gateway* local, para coletar, processar e encaminhar essas informações através da Internet para as aplicações consumidoras.

O *Mobile Hub* (M-Hub) é um middleware que possibilita a coleta, processamento e distribuição dos dados de uma grande quantidade de objetos inteligentes. O M-Hub é executado em dispositivos móveis, transformando-os em gateways da rede IoT. Ele representa uma entidade autônoma, capaz de detectar um conjunto de objetos disponíveis na vizinhança e monitorá-los, independentemente de outros M-Hubs em sua proximidade. Consequentemente, o mesmo conjunto de objetos é monitorado por vários M-Hubs, levando assim ao desperdício de recursos de comunicação, processamento e energia.

Neste contexto, esta dissertação apresenta um algoritmo de eleição de líder para o *middleware* M-Hub, de forma a permitir a comunicação e negociação entre diferentes dispositivos móveis, determinando o melhor *gateway* da *Internet of Mobile things* (IoMT) disponível no ambiente para cada objeto descoberto oportunisticamente. Este trabalho também apresenta um estudo de caso, experimentos relacionados ao tempo de detecção e recuperação de falhas do nó e, finalmente, a complexidade do algoritmo proposto baseado na troca de mensagens.

Palavras-chave: Internet das Coisas Móveis, Middleware, Eleição de líder.

Abstract

Internet of Things (IoT) can be defined as the interaction of technologies from different areas, such as: ubiquitous computing, protocols, communication technologies, and devices with embedded sensors and/or actuators, called smart objects. The interconnection of thousands of heterogeneous, addressable objects with network connectivity enables them to collect and share data, helping to improve people's lives. In some cases, objects have memory and processing restrictions and connectivity to medium- and long-range networks, thus requiring a local gateway to collect, process and forward this information over the Internet to consumer applications.

The M-Hub is a middleware that enables gathering, processing and distribution of data from a large number of smart objects. The M-Hub runs on mobile devices, turning them into IoT gateways. It represents an autonomous entity, able to detect a set of objects available in its neighborhood and monitoring them independently of other M-Hubs in proximity. Hence, the same set of objects is monitored by several M-Hubs, so leading to the wastage of resources of communication, processing, and energy.

In this context, this thesis presents a leader election algorithm for M-Hub middleware to allow communication and negotiation between different mobile devices, determining the best IoMT gateway available in the environment for each object discovered opportunistically. This work also presents a case study, experiments related to the time of detection and recovery of failure of the node and, finally, the complexity of the proposed algorithm based on the message passing.

Keywords: Internet of Mobile Things, Middleware, Leader Election.

Agradecimentos

Ao bom Deus em primeiro lugar, pela minha vida e por todas as maravilhas.

Ao meu orientador, o Prof. Rafael Lopes, pelo exemplo, apoio, compreensão, orientação. Agradeço ao Prof. Ariel Teles (IFMA) pelo apoio, dedicação e pela orientação dos artigos e escrita da dissertação mesmo sem ser do corpo docente da PPGCC, seu esforço foi muito importante para a conclusão deste trabalho.

Aos meus familiares pelo apoio que me deram nesta trajetória. Eles me fazem querer ser alguém melhor todos os dias.

Ao membros do LSDi-UFMA, com os quais compartilhei alegrias, tristezas, e conhecimento. Registro a importância de Adeilson Marques, Anderson Soares, Allinger Medeiros, Alysson Cirilo, Danne Makleyston, Fernando Benedito, Jhonsef Rabelo, Rodolfo Alves e Tércio Santana para a conclusão desta pesquisa.

Aos amigos José Denes, Jonnison Lima, Antonino Calisto que desde a época da graduação sempre estiveram presentes nos momentos difíceis. E ao novos amigos Moises Laurence e Thiago Pinheiro que o mestrado me proporcionou.

A UFMA e ao PPGCC pela estrutura dada a execução deste trabalho. Agradeço também ao chefe do PPGCC, ao coordenador e aos professores.

"Quando não souberes para onde ir, olha para trás e saiba pelo menos de onde vens."

Provérbio Africano

Lista de Figuras

Lista de Tabelas

Lista de Siglas

M-Hub *Mobile Hub*.

IoT *Internet of Things*.

IoMT *Internet of Mobile things*.

M-Objs *Mobile Objects*.

RFID *Radio-Frequency IDentification*.

RSSF *Redes de Sensores Sem Fio*.

SIG *Bluetooth Special Interest Group*.

BLE *Bluetooth Low Energy*.

L2CAP *Logical Link Control and Adaption Protocol*.

GATT *Generic Attribute Profile*.

ATT *Attribute Protocol*.

SMP *Security Manager Protocol*.

GAP *General Access Profile*.

RSSI *Received Signal Strength Indicator*.

UAVs *Unmanned Aerial Vehicle*.

WPAN *Wireless Personal Area Network*.

SDDL *Scalable Data Distribution Layer*.

CDDL *Context Data Distribution Layer*.

ID *Identificadores*.

S2PA *Short-Range Sensor, Presence and Actuation Service*.

FIFO *First In, First Out*.

P2P *Peer-to-Peer*.

UUID *Universally Unique Identifier*.

CPU *Central Processing Unit*.

JSON *JavaScript Object Notation*.

CEP *Complex Event Processingn*.

MR-UDP *Mobile Reliable UDP*.

MQTT *Message Queuing Telemetry Transport*.

Sumário

1 Introdução

A Internet da Coisas (IoT) pode ser definida como a interação de tecnologias de diferentes áreas, tais como: computação ubíqua, ciência de contexto, protocolos, tecnologias de comunicação e dispositivos com sensores e/ou atuadores embarcados, chamados de objetos inteligentes [4]. A interconexão de milhares de objetos endereçáveis, heterogêneos, e com conectividade de rede, permite aos mesmos coletarem e compartilharem dados a respeito do ambiente onde se encontram. Estes objetos podem ser utilizados para fornecer uma variedade de serviços em diversos setores, tais como gerenciamento de transporte público e privado, tratamento de água, gerenciamento e consumo de eletricidade, espaços públicos para entretenimento e atividades sociais como parques e teatros.

Em alguns casos, os objetos inteligentes possuem restrições de memória, processamento e conectividade com redes de médio e longo alcance. Eles não implementam uma pilha de protocolo TCP/IP, consequentemente, eles não conseguem acessar a Internet por meio de seus próprios recursos. Além disso, de acordo com Francis daCosta [11], o IPv6[1] não resolve todos os problemas da IoT, porque o gerenciamento, endereçamento e roteamento são seus maiores desafios. Os protocolos baseados em IP não são suportados pela grande maioria dos Objetos Inteligentes, assim como seus serviços não são adequados para a maioria dos aplicações da IoT. A razão é que os protocolos baseados em IP foram projetados intrinsecamente para ciclos de trabalho intensivo, grandes fluxos de dados e confiabilidade, enquanto na IoT a comunicação envolve mensagens pequenas, mas frequentes, onde cada mensagem individualmente não é importante, mas os fluxos de dados correspondentes transportam as informações relevantes.

Contudo, tais objetos inteligentes podem transmitir seus dados a um *gateway* local, por meio de tecnologias de comunicação de curto alcance (e.g., *Bluetooth*, *ZigBee*). Desse modo, ao receber os dados, o *gateway* poderá processar e encaminhar essas informações através da Internet para as aplicações consumidoras via tecnologias de comunicação sem fio (e.g., Wi-fi e 3G/4G).

[1] Versão mais atual do Protocolo de Internet (`http://ipv6.br/`)

Uma extensão da IoT chamada Internet das Coisas Móveis (*Internet of Mobile Things* - IoMT) [39] propõe cenários nos quais os objetos podem ser movidos ou podem se mover (i.e., chamados de Objetos Móveis - *Mobile Objects* (M-Objs)) com grande facilidade no ambiente e, ainda assim, permanecerem acessíveis e controláveis de forma remota. Neste cenário, dispositivos convencionais como *tablets* e *smartphones* podem atuar como *gateways*, promovendo a descoberta e a conexão oportunista com objetos em sua proximidade.

O *middleware Mobile-Hub* (M-Hub) [48], concebido no âmbito do projeto *ContextNet* [19], fornece um arcabouço capaz de realizar a coleta, processamento, análise e a visualização remota dos dados de uma grande quantidade de objetos, permitindo o desenvolvimento de diferentes tipos de serviços para a IoMT. O M-Hub é executado em dispositivos móveis (e.g., *tablets*, *smartphones*) transformando-os em *gateways* da IoMT. Ele descobre oportunisticamente os objetos inteligentes próximos à medida que se move, tal descoberta é realizada por meio da tecnologia *bluetooth*.

O M-Hub representa uma entidade autônoma, capaz de detectar a presença de um conjunto de objetos disponíveis na sua vizinhança e monitorá-los, independentemente de outros M-Hubs em sua proximidade. Entretanto, esta condição pode levar ao desperdício de recursos de comunicação, processamento e energia, visto que o mesmo conjunto de objetos será monitorado por múltiplos M-Hubs. Assim, os mesmos objetos poderão ter os mesmos dados coletados, processados e transmitidos desnecessariamente. Neste contexto, este trabalho de pesquisa apresenta um algoritmo de eleição de líder para o *middleware* M-Hub, de forma a permitir a comunicação e negociação entre diferentes dispositivos móveis, determinando assim o melhor *gateway* da IoMT disponível no ambiente cada objeto descoberto oportunisticamente.

1.1 Objetivos

A pesquisa à qual se refere esta dissertação de mestrado tem como objetivo geral desenvolver um algoritmo de eleição de líder para o *middlaware* M-Hub com o intuito de determinar o melhor *gateway* da IoMT para realizar os serviços de coleta, processamento e distribuição de dados para cada um dos objetos descobertos oportunisticamente a medida que se move pelo ambiente.

Com o intuito de demonstrar a viabilidade da proposta, este objetivo é concretizado a partir dos seguintes objetivos específicos:

1. Levantar o estado da arte referente a algoritmos de eleição de líder e ao *middleware* M-Hub.
2. Projetar e implementar funcionalidades da solução proposta, de acordo com a arquitetura do M-Hub.
3. Desenvolver estudos de caso e experimentos que permitam avaliar e aperfeiçoar a solução proposta.

1.2 Estrutura da Dissertação

Esta dissertação está organizada como segue:

- O Capítulo 2 apresenta os conceito sobre algoritmos de eleição de líder, da IoT/IoMT e por fim sobre o *middleware* M-Hub. O entendimento dos conceitos e do funcionamento dos componentes do M-Hub é fundamental para a compreensão da solução proposta neste trabalho.
- O Capítulo 3 apresenta os trabalhos relacionados ao problema abordado nesta pesquisa de mestrado.
- O Capítulo 4 aborda a solução proposta, apresentando as motivações e os requisitos para seu desenvolvimento. Este capítulo descreve ainda funcionalidades, em especial aquelas responsáveis pela troca periódica de mensagens e a função de eleição de líder.
- O Capítulo 5 apresenta um estudo de caso, experimentos relacionados ao tempo de detecção e recuperação de falhas do nó líder e por fim, o custo do algoritmo proposto.
- O Capítulo 6 apresenta as conclusões obtidas a partir desta pesquisa e apresenta trabalhos futuros que podem ser desenvolvidos a partir deste esforço inicial.

2 Fundamentação Teórica

Visando melhorar a compreensão sobre os aspectos relativos a esta pesquisa, o presente capítulo apresenta uma breve fundamentação teórica dos temas relativos ao contexto no qual este trabalho está inserido. São apresentados conceitos, características, aplicações e desafios relacionados a IoT/IoMT e sobre o *Middleware* M-Hub (Mobile-Hub). Por fim, é feita uma breve introdução aos algoritmos de eleição de líder em sistemas distribuídos.

2.1 Internet das Coisas

A Internet da Coisas (*Internet of Things - IoT*) pode ser definida como a interação de tecnologias de diferentes áreas, tais como: computação ubíqua, ciência de contexto, protocolos e tecnologias de comunicação e dispositivos com sensores e/ou atuadores embarcados, chamados de objetos inteligentes [4]. Esta integração visa gerar um sistema dinâmico global, em que milhares de dispositivos endereçáveis e heterogêneos são interligados por redes de comunicação, e são capazes de trocar dados uns com os outros sem a necessidade de intervenção humana, na maioria das ocasiões, por meio de comunicação Maquina-a-Maquina (*Machine-to-Machine* (M2M)) [2]. Esses objetos podem ser utilizados em diversos tipos de ambientes, fornecendo uma variedade de serviços em diversos setores, tais como gerenciamento de transporte público e privado, tratamento de água, gerenciamento e consumo de eletricidade, espaços públicos para entretenimento, atividades sociais como parques e teatros, industria, medicina, agricultura, dentre outros campos [3, 29]. A arquitetura geral de sistemas IoT pode ser vista na Figura 2.1.

O termo IoT foi criado em 1999, por Kevin Ashton [1]. Ele usou para explicar como os objetos da vida cotidiana estarão equipados com microcontroladores, transceptores e pilhas de protocolos que os tornarão capazes de se comunicar entre si e com os usuários. Estes dispositivos têm como principais características as capacidades de sensoriamento, atuação e comunicação.

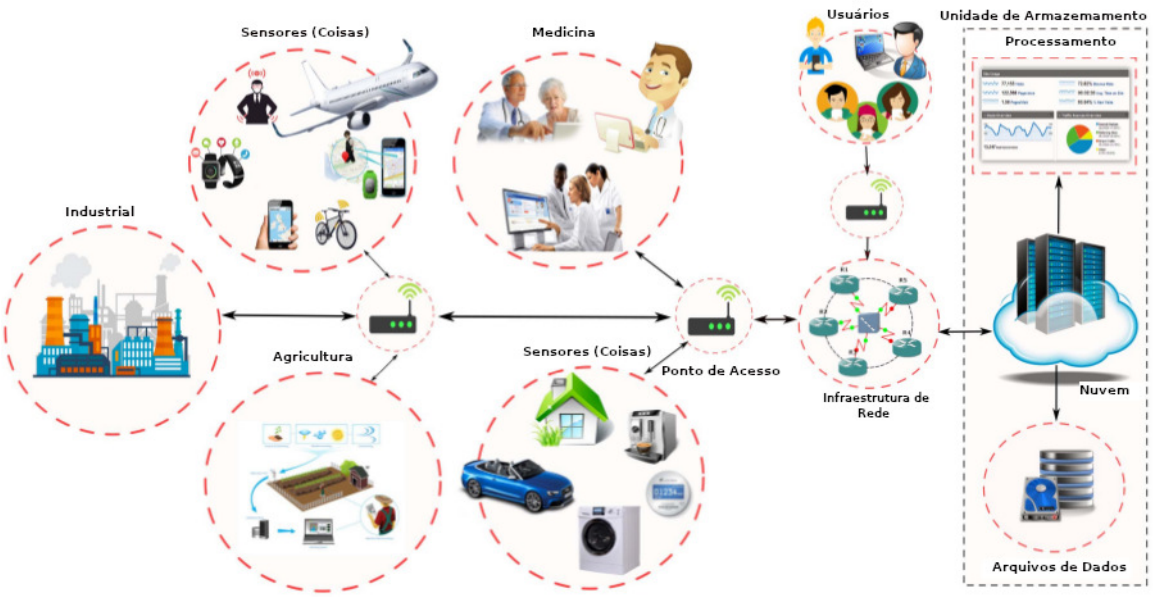

Figura 2.1: Arquitetura da IoT. Fonte: L. Farhan et al. [20]

A ideia básica é a presença generalizada em torno de uma variedade de coisas ou objetos - como *tags* de identificação de radiofrequência (*Radio-Frequency IDentification* (RFID)), sensores, atuadores, entre outros. Através de esquemas de endereçamento únicos, esses dispositivos são capazes de interagir uns com os outros e cooperar com seus vizinhos para alcançar metas comuns [4]. Os objetos inteligentes, como pode ser visto na Figura 2.2, possuem as seguintes características [38]:

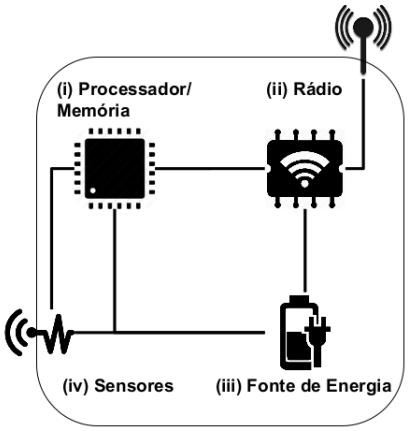

Figura 2.2: Objeto inteligente.

- Objetos que possuem um identificadores exclusivos;
- Ter um conjunto mínimo de funcionalidades de comunicação, como a capacidade de ser descoberto e aceitar mensagens recebidas e responder-las;

- Estão associados a pelo menos um nome e um endereço. O nome é uma descrição legível por humanos e pode ser usado para fins de raciocínio. O endereço é uma *string* legível pela máquina que pode ser usada para se comunicar com o objeto;

- Possuir algumas capacidades básicas de computação. Isso pode variar da capacidade de combinar uma mensagem recebida para um determinado objeto (como em RFID passivos), para a capacidade de realizar cálculos complexos, incluindo descoberta de serviços e tarefas de gerenciamento de rede;

- Pode possuir meios para detectar fenômenos físicos (por exemplo, temperatura, luz, nível de radiação eletromagnética) ou realizar ações com efeito sobre a realidade física (atuadores).

2.1.1 Domínio de aplicações da IoT

Existem vários domínios de aplicação que serão afetados pela emergente Internet das coisas. As aplicações podem ser classificadas com base no tipo de disponibilidade de rede, cobertura, escala, heterogeneidade, envolvimento do usuário e impacto [28]. Pode se categorizar as aplicações em quatro domínios: (1) Pessoal e Residencial; (2) Empresarial; (3) Utilitários; e (4) Móvel.

- **Pessoal e Residencial**: a utilização de dispositivos para realizar o gerenciamento de casas, por exemplo, automação residencial. A utilização de dispositivos que captem dados pertinentes à saúde das pessoas para acompanhamento médico e prevenção de acidentes. Estes sensores podem ser usados para reunir as informações de assistência médica e transmitir para centros médicos remotos [42, 45]. A IoT também foi desenvolvida em muitas áreas de infraestrutura: como casas inteligentes (*smart houses*), cidades inteligentes (*smart cities*), monitoramento ambiental e residencial. O termo "*smart cities*" foi proposto como um ecossistema ciber-físico com sensores e novos serviços em toda a cidade [34]. Atualmente, mais da metade da população mundial reside em megacidades [41]. A transformação de megacidades em cidades inteligentes [1] [2] alimentada pela IoT permite a interconexão de diversas áreas em uma escala inédita;

[1] `http://www.smartsantander.eu/`

[2] `https://amsterdamsmartcity.com/projects`

- **Empresarial**: tecnologias da IoT que empregadas em ambientes empresariais com o intuito de monitorar o ambiente para permitir um controle sobre o número de funcionários em um determinado local e a utilização de *Redes de Sensores Sem Fio* (RSSF) para atuar na segurança, são exemplos de implantação da IoT em meio empresarial;

- **Utilitários**: em geral, são utilizados os recursos da IoT para aperfeiçoar algum serviço já existentes, como: *smart grid* e *smart metering* [53]. Através do monitoramento contínuo de cada ponto de eletricidade é possível alcançar um consumo eficiente de energia dentro de uma casa e usando informações de consumo de energia para otimizar a forma como a eletricidade é consumida. Essa informação na escala da cidade é usada para manter o equilíbrio de carga dentro da rede, garantindo alta qualidade de serviço;

- **Móvel**: o emprego de dispositivos que monitorem e gerenciem a mobilidade urbana é um exemplo desse domínio de aplicação. Dispositivos com comunicação de curta distância podem ser utilizados para realizar leituras durante a movimentação dos objetos pela cidade e identificar possíveis congestionamentos. É possível ainda, a utilização desse domínio em logísticas de transporte onde é necessário fazer um acompanhamento em tempo real dos veículos em trânsito [33].

2.1.2 Comunicação dos Dispositivos da IoT

Em ambientes IoT existem uma vasta gama de dispositivos, com aplicações de grande e pequeno porte na indústria, *smart cities*, hospitais, dentre outros. Essas aplicações necessitam de uma tecnologia de comunicação apropriada. Dentre os dispositivos que utilizam este padrão pode-se incluir os dispositivos como, *smartphones*, impressoras, câmeras digitais, alguns periférico como teclados, mouses, *joysticks*, dispositivos embarcados, como travas elétricas, aparelhos de som automotivo e qualquer dispositivo que contenha um chip *Bluetooth*. O *Bluetooth* opera entre as faixas 2.400 a 2.485 MHz dividida em diferentes canais. Embora esta faixa de frequência não seja licenciada, existe o grupo que regulamenta a utilização da

tecnologia, chamado *Bluetooth Special Interest Group* (SIG) [3], que e formado por mais de 20 mil empresas e membros.

Ao longo dos anos, várias versões desta tecnologia de transmissão foram desenvolvidas. Em cada uma delas, itens como a segurança, a velocidade de transmissão, facilidade de conexão, economia de energia, alcance, entre outros, foram melhorados. A Tabela 2.1 apresenta a evolução do *Bluetooth* até a versão 3.0.

Tabela 2.1: Evolução das versões do *Bluetooth* [35]

Ano	Versões - Principais Características	Dispositivos
1999	Bluetooth 1.0 - Apresentava problemas de comunicação entre os dispositivos	Computadores, Telefones Celular
2001	Bluetooth 1.1 - Representa o estabelecimento do Bluetooth como padrão IEEE 802.15. Problemas encontrados na versao anterior foram resolvidos.	Impressora, Laptop, Mouse, Headset,
2003	Bluetooth 1.2 - Conexões mais rápidas, proteção contra interferências mais eficiente.	MP3 Player, Equipamentos Médicos, GPS, Câmera Digital
2004	Bluetooth 2.0 - Diminuição do consumo de energia e aumento na velocidade de transmissão de dados para 3 Mbps. dispositivos.	Fone Estéreo, Relógio, Radio-Relógio
2007	Bluetooth 2.1 - Permite uma melhor seleção dos dispositivos antes de estabelecer conexão, melhoria nos procedimentos de segurança e melhor gerenciamento do consumo de energia.	Televisão, Porta-Retrato
2009	Bluetooth 3.0 - altas taxas de velocidade de transmissão de dados. Dispositivos compatíveis podem atingir a taxa de transmissão de 24 Mbps.	Eletrodomésticos

O *Bluetooth* foi projetado para funcionar em curto alcance geralmente atingindo alguns metros. O alcance efetivo depende de vários fatores como propagação, interferência, atenuação, reflexão do sinal, características da antena e potência de transmissão. Os dispositivos *Bluetooth Classic* são divididos em três

[3] `https://www.bluetooth.com/`

classes, que especificam sua potência máxima permitida, o que, por sua vez, afeta significativamente o alcance [13]. As diferentes classes, limites de potência e alcance estão listadas na Tabela 2.2.

Tabela 2.2: *Bluetooth Classic* alcance máximo.

Classe	Alcance Máximo	Potência Máxima de Transmissão
Classe 1	100 m	20 dBm
Classe 2	10 m	4 dBm
Classe 3	1 m	0 dBm

Em 2010 o SIG publicou a especificação 4.0 que introduziu o *Bluetooth Low Energy* (BLE) também conhecido como *Bluetooth Smart*. Uma de suas características mais marcantes é o baixo consumo de energia, fato que garantiu o desenvolvimento de novos dispositivos que mesmo alimentados por bateria possuem anos de autonomia. A tecnologia tornou-se forte candidata a protagonizar a disseminação da IoT, uma vez que as características de comunicação bastante limitados em relação ao *Bluetooth* tradicional, chamado *Bluetooth Classic*, eram insuficientes para este cenário [16].

O *Bluetooth LE* adiciona um novo modo de funcionamento em comparação com o *Bluetooth* tradicional. Ao usar o *Bluetooth LE* para determinadas aplicações, não é mais necessário fazer um pareamento para realizar troca dados. O *Bluetooth LE* suporta quatro modos diferentes: *Broadcaster*, Observador, Central e Periférico. No modo *Broadcaster* os dados podem ser enviados nos canais de anúncio sem estabelecer uma conexão. Um modo complementar ao *Broadcaster* é o papel de Observador que recebe os dados transmitidos.

A função Central é semelhante à função do mestre no *Bluetooth Classic*. Contudo, segundo a especificação do *Bluetooth Smart*, um dispositivo de função Central, teoricamente, pode suportar infinitas conexões simultâneas de diferentes dispositivos periféricos. Esta é uma melhoria da especificação em comparação com o *Bluetooth Classic*, que suporta no máximo 7 conexões [13].

Um dispositivo periférico é tipicamente um dispositivo simples que pode manipular apenas uma conexão ativa por vez, diferentemente de um dispositivo central. Um dispositivo pode suportar uma única, várias ou todas as funções especificadas, mas apenas uma pode estar ativa por vez.

A pilha de protocolos do *Bluetooth LE* é composta de duas partes, uma denominada controlador e a outra *host*. O controlador representa a camada física. Basicamente é um pequeno chip com um rádio de funcionamento bastante simples [27]. Já o *host* consiste no software e camadas de nível mais alto que executam os seguintes serviços:

- ***Logical Link Control and Adaption Protocol* (L2CAP)**: responsável pela multiplexação dos dados entre as camadas mais altas e a mais baixa, chamada *Control Link*. O L2CAP do BLE funciona de maneira mais eficiente que no *bluetooth Classic*, pois não fornece retransmissão ou controle de fluxo, nem segmentação ou remontagem, visto que as camadas mais altas permitem apenas o envio de pacotes que possuam o tamanho que se enquadre no limite máximo da carga útil do L2CAP;

- ***Generic Attribute Profile* (GATT)**: permite ao cliente e servidor trocarem informações de maneira estruturada, isto é, define como os dados devem ser organizados e enviados pelas aplicações. O cliente GATT envia um requerimento ao servidor que por sua vez responde a requisições do cliente. Estas respostas incluem dados referentes aos atributos. O cliente inicia a requisição pois em um primeiro momento não conhece os atributos do servidor;

- ***Attribute Protocol* (ATT)**: usado para enviar atributos entre dois dispositivos que estão se comunicando. Tais atributos são um tipo de estrutura de dados que o GATT utiliza para enviar e receber dados;

- ***Security Manager Protocol* (SMP)**: o procedimento para estabelecer a conexão entre dispositivos é realizado através de uma série de etapas que requerem diversas trocas de chaves de criptografia. O SMP é o responsável por realizar estas trocas. Além disso, oferece funções de segurança às outras camadas de maneira que a troca de informações seja sempre segura através de um link criptografado;

- ***General Access Profile* (GAP)**:define os procedimentos genéricos relacionados à descoberta de dispositivos *Bluetooth* e aos aspectos de gerenciamento dos links da conexão dos dispositivos *Bluetooth* (procedimentos de modo de conexão).

Também define procedimentos relacionados ao uso de diferentes níveis de segurança.

O alcance de transmissão do *Bluetooth LE* é semelhante ao do *Bluetooth Classic* dependendo da potência de transmissão, bem como diferentes interferências do ambiente. Embora o *Bluetooth LE* tenha sido projetado principalmente para baixo consumo de energia, a energia de transmissão de até +10dBm é suportada. Normalmente, os dispositivos *Bluetooth LE* operam com 0 dBm ou menos. Usando a potência de transmissão máxima de +10 dBm obtém-se um alcance teórico de >300 m, enquanto que o poder de transmissão de 0 dBm ,mais comumente usado, tem um alcance teórico de cerca de 50 m [23].

O indicador de intensidade do sinal recebido (*Received Signal Strength Indicator* (RSSI)) é uma indicação da intensidade do sinal experimentada pelo receptor de um dispositivo *Bluetooth*. É um valor inteiro de 8 bits, e no caso do BLE varia entre -127 e 20 dBm. Quanto maior o valor, maior a intensidade do sinal recebido. O valor do RSSI pode ser recuperado durante o *scan* de dispositivos, bem como durante a conexão com outro dispositivo. Uma diferença importante entre o *Bluetooth Classic* e o BLE relacionado a esse parâmetro é como o RSSI pode ser obtido. No *Bluetooth Classic*, duas formas eram possíveis. As primeiras versões do *Bluetooth* só permitiam recuperar o parâmetro durante uma conexão ativa entre dois dispositivos. Especificações posteriores (1.2 e superiores) permitiram recuperar o parâmetro durante a fase de "*scan*“. No BLE foi melhorado, e o RSSI é obtido, passivamente ao receber mensagens de anúncios. A possibilidade de recuperá-lo durante uma conexão ativa foi preservada. E por meio do RSSI é possível obter aproximadamente a distância do dispositivo [47,52].

2.1.3 Internet das Coisas Moveis

Os dispositivos IoT são, em sua maioria, objetos estáticos. No entanto dispositivos mais modernos podem se locomover de forma autônoma ou serem movidos de forma quase irrestrita, ao mesmo tempo em que são capazes de comunicar-se entre si e com a infraestrutura de redes disponível, representando assim, uma extensão da Iot chamada de Internet das Coisas Moveis (IoMT).

No cenário IoMT as coisas conectáveis (ou Objetos) podem ser movidas ou podem se mover com grande facilidade, e ainda assim devem ser acessíveis remotamente e controláveis. Objetos móveis (M-OBJs) podem ter tamanho, finalidade e complexidade muito diferentes - podem variar de veículos terrestres de qualquer tipo (carros, ônibus, etc.), robôs domésticos ou industriais móveis, robôs aéreos (*Unmanned Aerial Vehicle* (UAVs)) e *tags* de identificação.

Segundo Nahrstedt [40] a diferença entre IoT e IoMT é que, ao considerar a mobilidade das coisas, mudanças importantes ocorrem em termos de: (a) contexto, por exemplo, onde o dispositivo móvel está localizado, e com quem ele está agora, (b) acesso à Internet e conectividade, por exemplo, caso o dispositivo móvel esteja conectado e, quando, qual a largura de banda ou os esquemas de segurança disponíveis, (c) disponibilidade de energia, por exemplo, onde o dispositivo móvel pode ser recarregado novamente, quanto de energia o aplicativo móvel precisa? (d) segurança e privacidade, por exemplo, qual o tipo de infraestrutura de segurança que o dispositivo móvel encontra ao se deslocar entre diferentes locais e quais informações os provedores de serviços têm sobre o usuário.

Desse modo, o cenário da IoMT apresenta vários desafios na concepção e implementação de sistemas de coleta de dados. Esses desafios incluem:

- Gerenciar um conjunto de objetos;
- Gerenciamento de energia dos dispositivos moveis;
- Desenvolvimento de aplicações para sensorear/atuar no ambiente;
- Privacidade dos dados coletados;
- Compreender a mobilidade e o contexto das pessoas.

Essencialmente, o paradigma IoMT considera qualquer situação em que a posição relativa e a velocidade entre os M-OBJs e os *gateways* são variáveis e pode mudar a qualquer momento, e onde M-OBJs podem ser alcançados através de *gateways* diferentes e até mesmo múltiplos ao longo do tempo. Portanto, a IoMT engloba todas as situações em que: (i) os M-OBJs estão permanentemente associados a um lugar e o *gateway* se move através dos locais para interagir oportunisticamente com os M-OBJs; (ii) os M-OBJs estão ligados a elementos móveis, enquanto um *gateway* está ligado a

um local; (iii) um ou mais M-OBJs podem estar em co-movimentação com um *gateway* móvel, durante um certo período de tempo [48].

2.2 Middleware Mobile-Hub (M-Hub)

Em alguns casos, os objetos possuem restrições de memória, processamento e conectividade com redes de médio e longo alcance. Consequentemente, eles não conseguem acessar a Internet por meio de seus próprios recursos. Contudo, são capazes de transmitir seus dados a um *gateway* local, por meio de tecnologias de comunicação de curto alcance (*Bluetooth Classic*, BLE). Desse modo, ao receber os dados, o *gateway* poderá processar e encaminhar essas informações através da Internet para as aplicações consumidoras via tecnologias de comunicação sem fio (e.g., Wi-fi e 3G/4G).

Considerando que os dispositivos móveis, telefones e *tablets* estão se tornando cada vez mais onipresentes, acessíveis e poderosos. Tais dispositivos tornam-se os candidatos naturais para serem os nós de propagação, ou seja, os *gateways* que poderão realizar a coleta, processamento e distribuição dos dados do objetos da IoT/IoMT para as aplicações consumidoras através da Internet.

O M-Hub [48] é um serviço de middleware, concebido no âmbito do projeto *ContextNet* [19], que executa em dispositivos pessoais móveis, responsável pela descoberta, conexão e aquisição de dados de baixo nível junto a objetos inteligentes (e.g. sensores, atuadores, relógios, robôs, veículos) próximos e acessíveis a partir de tecnologias *Wireless Personal Area Network* (WPAN) de curto alcance, como *Bluetooth Classic* e BLE, ou que estejam embutidos no próprio dispositivo móvel. Portanto, o M-Hub é um *gateway*, servindo de ponte entre os objetos inteligentes e a Internet.

O M-Hub descobre oportunisticamente os objetos inteligentes (móveis ou estacionários) à medida que se move pelos ambientes, e permite a comunicação remota, através de uma camada de distribuição de dados: o *middleware Scalable Data Distribution Layer* (SDDL)[4] [14] ou o *middleware Context Data Distribution Layer* (CDDL)[5] [26].

[4]http://www.lac.inf.puc-rio.br/dokuwiki/doku.php
[5]http://www.lsdi.ufma.br/projetos/cddl/doku.php

O M-Hub fornece as seguintes funcionalidades:

1. **Descoberta de sensores próximos**: o M-Hub, periodicamente, procura objetos inteligentes nas proximidades que estejam anunciando seus *Identificadores* (ID) e capacidades. Estas informações sobre objetos inteligentes alcançáveis são mantidas no banco de dados do M-Hub;

2. **Conexão com sensores**: dependendo do protocolo WPAN usado, o M-Hub pode precisar estabelecer um link de comunicação com o sensor, sobre o qual ele irá emitir solicitações síncronas ou assíncronas para receber dados periodicamente;

3. **Protocolo de transcodificação**: Os pacotes de dados recebidos dos sensores podem ter diferentes formatos e codificações. Assim, o M-Hub transcodifica e serializa os dados, antes de transmití-los. A transcodificação de dados é altamente dependente do tipo, marca e fabricante do sensor;

4. **Caching de dados**: a fim de otimizar a transmissão para a nuvem através da Internet móvel, o M-Hub pode agrupar várias amostras de dados obtidas a partir de vários sensores próximos antes de enviá-los ao gateway em uma única rajada. Para fazer isso, ele armazena em cache as amostras de dados enviadas pelos sensores;

5. **Configuração e Controle dos sensores**: dependendo do tipo de sensor, o M-Hub pode, eventualmente ou periodicamente, enviar comandos, definições de parâmetros ou requisições de consultas de dados de através da WPAN para os sensores conectados;

6. **Pré-processamento de dados do sensor**: antes de enviar os dados pode ser necessário aplicar uma função de pré-processamento (e.g. transcodificação, formatação, agregação, filtragem, ou comparação com leituras anteriores etc.). Esse pré-processamento é feito no M-Hub;

7. **Carregamento dinâmico de módulos do sensor**: uma vez que não é possível ter módulos internos para todos os sensores que podem estar disponíveis à medida que o usuário se move, o M-Hub oferece um mecanismo de implantação em tempo de execução e gerenciamento do ciclo de vida de módulos específicos de sensores;

8. **Processamento Ciente de Energia**: através de um componente de gerenciamento de energia, o M-Hub monitora o nível da bateria do dispositivo móvel e dispara ações adaptativas que ajustam os comportamento dos seus serviços de acordo com a disponibilidade de energia do dispositivo móvel.

Em relação às WPANs, o M-Hub oferece suporte a conexão com objetos inteligentes via *Bluetooth Classic* (2.0 e 3.0) e *Bluetooth Low Energy* (BLE 4.0). Apesar de seu maior consumo de energia, o *Bluetooth Classic* ainda é utilizado por uma ampla variedade de dispositivos empregados na área da saúde, tais como o Zephyr BioHarness 3 e Zephyr HxM. A BLE está emergindo como uma tecnologia muito promissora. Isso porque ela é mais eficiente em relação ao consumo de energia, e também permite uma descoberta mais rápida de dispositivos periféricos. Além disso, a BLE suporta cerca de 2.500 conexões simultâneas. Mas a razão mais importante para a escolha da BLE é o fato de ela está disponível em um número crescente de modelos de smartphones (e.g. Android, iOS e BlackBerry) e está sendo incorporada em uma crescente variedade de dispositivos. Por ter uma arquitetura aberta e independente de tecnologia, o M-Hub pode ser estendido para dar suporte a novas WPANs.

A arquitetura do M-Hub, ilustrada na Figura 2.3, é composta por diversos serviços e gerenciadores locais [25, 48, 50], todos executando em *background* e em paralelo com os aplicativos do usuário.

- ***Short-Range Sensor, Presence and Actuation Service* (S2PA)**: responsável pela descoberta, conexão e comunicação com os M-Objs. Este componente define uma API que fornece uma abstração para a comunicação com *smart objects* que utilizam tecnologias de curto alcance (e.g., *Bluetooth Low Energy* e *Bluetooth Classic*). Para a realização da troca de dados com os *smart objects* é necessária a utilização do *driver* do objeto que deverá ser fornecido pelo seu fabricante;

- ***LocationService*** é responsável por registrar a posição atual do M-Hub e anexá-la a qualquer mensagem que será transmitida;

- ***Connection Service***: foi implementado sobre a ClientLib [6], uma biblioteca utilizada para estabelecer a comunicação, via *Mobile Reliable UDP* (MR-UDP), com um *gateway* SDDL;

[6] `http://wiki.lac.inf.puc-rio.br/doku.php?id=download`

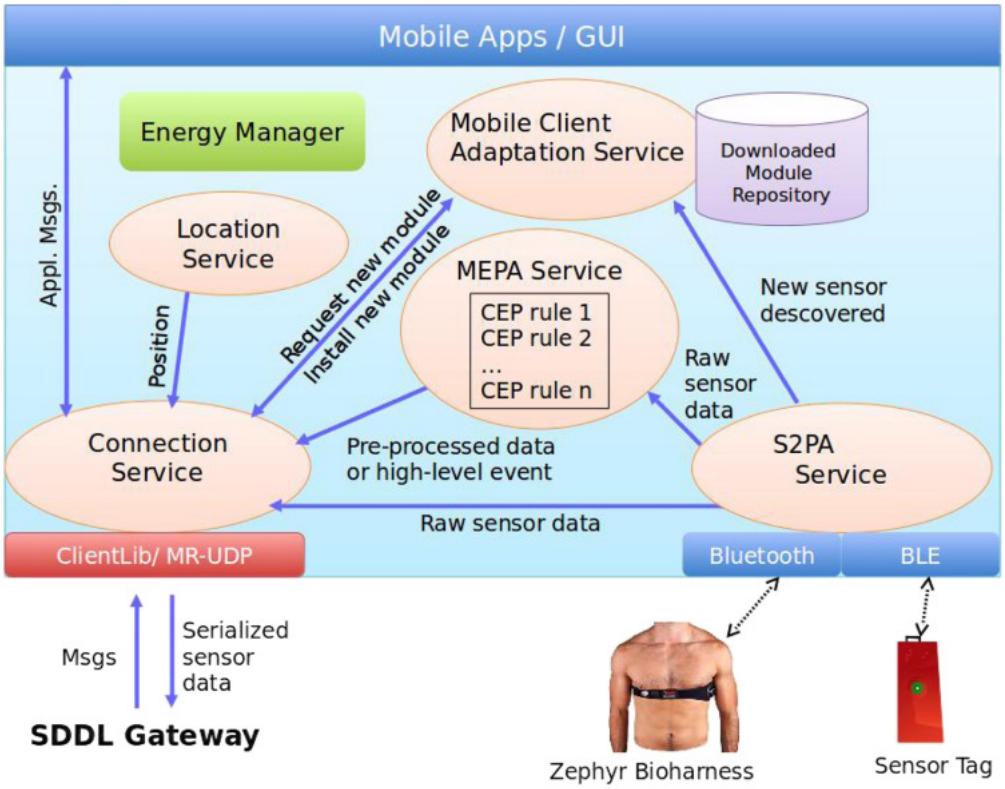

Figura 2.3: M-Hub: seus principais componentes. Fonte: [48]

- ***MEPA Service***: permite uma análise contínua sobre os fluxos de dados lidos dos *smart objects* em busca de identificar padrões definidos pela aplicação móvel;
- A periodicidade e a duração das ações realizadas pelo *LocationService*, *S2PAService* e *ConnectionService* são influenciadas pelo nível de energia dos dispositivos (baixo, médio, alto). Isso será definido pelo *Energy Manager*, que coleta amostras do nível da bateria do dispositivo e verifica se ele está conectado a uma fonte de alimentação.

O M-Hub foi projetado e desenvolvido por pesquisadores do *Laboratory For Advanced Collaboration* (LAC)[7] da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), no âmbito do projeto ContextNet. O processo de desenvolvimento do M-Hub contou com a colaboração de pesquisadores do Laboratório de Sistemas Distribuídos Inteligentes (LSDi)[8] da Universidade Federal do Maranhão (UFMA).

O M-Hub representa uma entidade autônoma, capaz de detectar a presença de um conjunto de objetos disponíveis na sua vizinhança e monitorá-los,

[7] http://www.inf.puc-rio.br/blog/laboratorio/lac/
[8] http://www.lsdi.ufma.br/

independentemente de outros M-Hubs em sua proximidade. Entretanto, a presença de múltiplos M-Hubs nas proximidades de diversos M-Objs pode levar ao desperdício de recursos de comunicação, processamento e energia dos M-Hubs, visto que o mesmo conjunto de objetos será monitorado por múltiplos M-Hubs. Desse modo, torna-se necessário o desenvolvimento de um mecanismo para realizar a negociação entre os M-Hubs para definir o melhor *gateway* da IoMT para os objetos descobertos oportunisticamente à medida em que se move pelo ambiente.

Para projetar um algoritmo que permita aos M-Hubs decidir como distribuir o monitoramento entre os diversos M-Objs disponíveis no ambiente, foi realizado o estudo de alguns algoritmos eletivos aplicados no contexto de sistemas distribuídos. Esses algoritmos são apresentados na seção a seguir.

2.3 Algoritmos de Eleição de Líder

Diversos algoritmos distribuídos exigem a existência de um nó central que desempenhe o papel de coordenador ou líder para realizar alguma função especial durante a execução de um processo. Em alguns nós distribuídos, é essencial a utilização de algoritmos de eleição de líder para determinar qual nó assumirá esta função, sendo que, muitas vezes, não importa qual nó seja o novo líder, desde que alguém seja eleito. E é necessário que todos os nós estejam de acordo com a escolha do novo líder e que novas eleições possam ser realizadas para eleger substitutos de um líder.

A eleição do líder [6] é um problema importante para construir sistemas distribuídos tolerantes a falhas, e é uma das partes mais críticas de qualquer sistema distribuído. Este é um problema desafiador em um sistema distribuído porque os dados são distribuídos entre diferentes nós que são geograficamente separados. Para manter a coordenação entre os nós, um líder deve ser selecionado.

Considerando um arranjo com N nós, podem ser realizadas no máximo N eleições concorrentemente, sendo que cada nó pode iniciar no máximo uma eleição por vez. Independente disso, um requisito importante é de que somente um líder seja eleito para um processo ou para um conjunto de processos. A qualquer momento, um nó pode ser considerado um participante ou um não-participante, ou seja, no primeiro

caso, significa que o nó está participando da eleição, enquanto que no segundo caso o nó em questão não está.

Os requisitos para se determinar o nó líder, assim como a tolerância a falhas empregadas pelo algoritmo dependem do cenário de aplicação do algoritmo. Por exemplo, pode-se selecionar o novo líder por meio do ID do processo, por meio das capacidades computacionais do nó, assim como outros parâmetros específicos para cada cenário como os apresentados na Tabela 2.3.

Tabela 2.3: Exemplos de parâmetros para eleição de líder.

Parâmetro	Definição
P1	Estado da Bateria: carregando, descarregando ...
P2	Uso da Bateria: porcentagem de uso da bateria.
P3	Processador: atividade dos núcleos
P4	Distância em relação aos objetos
P5	Distância em relação aos outros nós

Durante um processo de eleição de líder, os nós trocam esses parâmetros por meio de mensagens. As mensagens trocadas durante uma eleição integram as informações importantes (parâmetros) que o algoritmo de eleição de líder precisará para determinar se um nó tem a capacidade de se tornar Líder. Dependendo do cenário, apenas um parâmetro pode ser utilizado ou a combinação de um conjunto de parâmetros.

Serão apresentados alguns algoritmos para eleição de líder empregados em diferentes cenários. Primeiramente serão apresentados dois algoritmos clássicos de eleição de líder: o primeiro deles realiza o processo de eleição supondo que a comunicação entre os processos é realizada ao redor de um anel; o segundo não exige a disposição em anel, mas exige que cada processo conheça informações sobre os identificadores de outros processos. Além dos algoritmos para eleição de líder discutidos nessa seção, serão apresentados uma série de outros algoritmos para cenários em que os processos são organizados em topologias arbitrárias e dinâmicas.

2.3.1 Eleição de líder em anel

Essa solução para eleição de líder em sistemas distribuídos exige que os nós estejam organizados em um anel lógico [22], onde cada nó sabe quem é seu sucessor. As mensagens são enviadas no sentido horário através de um canal de comunicação que liga cada nó ao seu vizinho. O algoritmo parte do princípio que os nós não estão sujeitos a falhas durante a execução do algoritmo de eleição, e que o sistema é assíncrono.

Inicialmente, no processo de eleição de líder em anel, todos os nós são marcados como não-participantes. Quando um nó identificar a ausência de um líder, ele inicia o processo de eleição primeiramente marcando-se como participante e em seguida enviando uma mensagem de eleição contendo seu identificador para seu vizinho no anel.

Ao receber uma mensagem de eleição, um nó deve comparar o identificador contido na mensagem com o seu próprio. Se o identificador recebido for maior, o nó deve passar adiante a mensagem para seu vizinho. Se por outro lado, o identificador contido na mensagem for menor que o identificador do nó em questão e o mesmo ainda não for um participante, então este deverá substituir o conteúdo da mensagem pelo seu próprio identificador antes de enviá-la para seu vizinho. Porém, caso o identificador do receptor seja maior que o contido na mensagem e este já for um participante, nada deve ser feito, uma vez que seu identificador já está circulando no anel.

Caso o identificador contido na mensagem seja igual ao do nó receptor, significa que o identificador deu uma volta completa no anel, vencendo todos os demais, e sendo portanto o maior identificador dentre todos os nós. Resta a esse processo se anunciar como coordenador para os demais. Inicialmente ele marca-se como não-participante e envia para seu vizinho uma mensagem do tipo coordenador contendo seu identificador. Ao receber uma mensagem de coordenador, o receptor marca-se como não participante, guarda o identificador do novo coordenador e envia a mensagem para seu vizinho. Ao completar uma volta no anel, a mensagem de coordenador é removida e os nós voltam a trabalhar normalmente, considerando o processo vencedor da eleição como o novo líder.

Em relação à complexidade do algoritmo, o pior caso de execução acontece quando o nó que inicia a eleição é o vizinho seguinte (no sentido horário) do nó com

o maior identificador. Nesse caso, serão necessárias N – 1 mensagens até atingir o nó que realmente será o futuro líder, o qual só então colocará seu identificador na mensagem que será enviada N vezes até completar o circuito. Por fim, a mensagem de coordenador será enviada N vezes para informar o resultado da eleição aos demais nós do anel, totalizando 3(N - 1) mensagens.

No melhor caso, o nó com o maior identificador inicia a execução, de forma que seu identificador seja enviado N vezes até completar uma volta no anel. Outras N mensagens serão necessárias para propagar o resultado da eleição, totalizando 2N mensagens.

2.3.2 Algoritmo "valentão" (*Bully algorithm*)

O algoritmo "valentão" para eleição de líder diferencia-se da solução anterior pela sua capacidade de tolerar falhas nos nós durante a execução da eleição (porém, a comunicação em rede deve ser confiável). Além disso, esse algoritmo faz uso de *timeouts* para detectar possíveis falhas nos nós participantes, exigindo, portanto, que o sistema seja síncrono.

Essa solução utiliza três tipos de mensagens diferentes: a mensagem de eleição é utilizada por um nó para indicar um processo de eleição; a mensagem OK é utilizada para responder a uma mensagem de eleição; por último, uma mensagem do tipo coordenador é utilizada para que um nó se anuncie como o novo líder.

Uma eleição é iniciada quando um nó verifica que o atual coordenador não está mais respondendo (através da utilização de *timeouts*). Diversos nós podem perceber, ao mesmo tempo, que o coordenador falhou, iniciando assim uma ou mais eleições concorrentes.

Se o nó que iniciar a eleição apresentar o maior identificador dentre todos os nós, este pode simplesmente se anunciar para os outros como sendo o novo líder. Por outro lado, se o nó não possuir o maior identificador, este deve enviar uma mensagem de eleição para todos os outros que possuírem identificador maior que o seu e aguardar pelas mensagens de resposta. Caso o nó em questão não receba nenhuma resposta dentro de um determinado intervalo de tempo, este vence a eleição e pode considerar-se o novo líder, bastando apenas enviar mensagens de coordenador para todos os

demais nós que possuírem identificador menor que o seu próprio identificador. De outra forma, se algum nó com identificador superior responder a mensagem de eleição com uma mensagem de OK, o nó que iniciou a eleição pode se retirar da disputa pela liderança, pois a tarefa será completada por algum nó com identificador superior.

Ao receber uma mensagem de eleição, o receptor responde com uma mensagem de OK, informando ao nó com identificador mais baixo a existência de pelo menos um superior. Além disso, cabe a esse nó com identificador mais alto iniciar uma nova eleição (se ainda não iniciou), considerando apenas os nós que possuam identificador maior que o seu. Quando um nó recebe uma mensagem de coordenador, o procedimento a ser tomado é apenas guardar o identificador recebido na mensagem e tratar o nó correspondente como o novo líder.

Um nó que é incluído (ou reincluído) ao conjunto de nos ativos é responsável por iniciar uma nova eleição. Caso esse nó possua o maior de todos os identificadores, ele vencerá a eleição, anunciará seu identificador para todos os demais nós e assumirá o papel de líder, justificando o nome dado ao algoritmo (“valentão”).

Em relação à complexidade, o pior caso de execução corresponde à situação na qual o nó com o identificador mais baixo detectar a ausência de um líder e, a partir daí, N – 1 eleições serão iniciadas, sempre com os nós com identificadores mais baixos enviando mensagens para os nós com identificadores mais altos. Nesse caso, o algoritmo exige O(n^2) mensagens.

O melhor caso de execução acontece quando o nó com o segundo identificador mais alto detectar que o líder atual não está mais respondendo. Caberá a esse nó eleger-se como o novo líder e enviar um total de N – 2 mensagens para todos os demais nós que possuírem o identificador menor que o seu para informar o resultado da eleição. A seguir serão classificados alguns algoritmos de eleição de líder de acordo com os pressupostos e métodos utilizados.

2.4 Taxonomia dos Algoritmos de Eleição

Os autores em [24] examinam diferentes algoritmos para a eleição do líder. Em relação aos pressupostos e métodos utilizados, os autores classificaram os

algoritmos em: eleição de líder em salto único, múltiplos saltos, hierárquicos e eleição de N líderes.

2.4.1 Eleição de Líder em salto único

O primeiro conjunto de algoritmos de eleição do líder em redes sem fio é para redes de salto único. Isso significa que todos os nós se comunicam entre si diretamente e sem intermediários. Em outras palavras, cada nó envia uma mensagem, todos os nós a recebem. No entanto, esses algoritmos necessitam de uma infraestrutura de rede (e.g., um *Hotspot* fixo ou móvel, um servidor remoto, um *broker MQTT* [9] [10] para permitir uma comunicação entre as nós.

Esses algoritmos dividem o tempo em intervalos de tempo iguais (t) para que cada nó tenha oportunidade de transmitir suas mensagens. Os algoritmos de salto único podem ser utilizados em cenários onde, (1) o número de nós (n) é conhecido; (2) o número de nós é desconhecido, mas há um limite de tempo (t) para a troca de mensagens; (3) o número de nós e o limite de tempo é desconhecido.

2.4.2 Eleição de Líder em saltos múltiplos

Na subseção 2.4.1, foram apresentados protocolos baseados em único salto. Esses protocolos usam *Broadcast* para transmissão de dados em redes sem fio. Em algoritmos de saltos múltiplos, para enviar uma mensagem para um nó em particular, podem haver nós intermediários para reencaminhar as mensagens. Em outras palavras, para enviar uma mensagem, pode haver um nó intermediário, dessa forma, saltos múltiplos são necessários para realizar a trocar mensagens entre os nós. No entanto, tais algoritmos necessitam de algoritmos de roteamento para transmitir mensagens entre os nós.

Os algoritmos deste grupo contêm as seguintes pre-requisitos. Todos os nós possuem um identificador exclusivo. Todos os *links* são Bidirecionais e *First In, First Out* (FIFO). Os nós têm *buffer* suficiente para realizar troca de mensagens. A topologia da rede pode mudar com o tempo. Não há limite do número de nós na rede. Os

[9] Message Queuing Telemetry Transport

[10] `http://mqtt.org/`

protocolos de eleição de líderes são classificados em duas categorias, *Random Election* e *Extreme-Finding*.

- *Random Election*: Esses algoritmos elegem o líder aleatoriamente, ou seja, o nó que detecta a alteração da topologia, anuncia-se como o líder. Os algoritmos contidos nessa classe têm como característica o uso de *Broadcast* para realizar a troca de mensagens entre os nós.
- *Extreme-Finding*: Os algoritmos baseados em *Random Election* elegem o líder aleatoriamente, entretanto em muitas ocasiões a eleição do líder deve usar como base em parâmetros correspondentes ao nó, como poder computacional ou energia restante. Dessa forma, os algoritmos desse grupo buscam eleger o nó com os melhores parâmetros. Por exemplo, os algoritmos *Leader Election Algorithm for Ad* [51] e *Candidate Base LEAA algorithm* [43] utilizam como base o algoritmo de termino de Dijkstra and Sholten para encontrar o nó com os melhores parâmetros.

2.4.3 Eleição de Líder Hierárquico

Este grupo de algoritmos divide a rede *Ad-hoc* em *clusters* e cada *cluster* possui um Líder. O líder de um grupo de líderes é denominado *cluster Head*. Os nós internos do *cluster* estão relacionados uns com os outros no que diz respeito à situação imprevisível da estrutura da rede. Em particular, a falta do *cluster Head* causa perda de comunicação no *cluster* ou mesmo em toda a rede. Então, quando se perde um *cluster head*, precisa-se de uma nova eleição. Como exemplos de algoritmos de líder hierárquico, tem-se *A hierarchical leader election protocol for mobile ad hoc networks* [12], *Design and implementation of a leader election algorithm in hierarchy mobile ad hoc network* [54].

2.4.4 Eleição de K Líderes

Este grupo de algoritmos é aplicado em cenários em que é necessário manter mais de um nó como candidato a se tornar líder para um determinado processo: Líder e sub-líder, ou quando há um número n de processos diferentes, que necessitam de

K Líderes. Pode-se fazer uma alusão às eleições políticas, em que é possível eleger candidatos para diferentes cargos.

Nos algoritmos de K líderes, para definir os melhores líderes são empregadas características relacionadas aos nós e os processos que estão em disputa, assim como as características e requisitos do cenário em que o algoritmo está sendo empregado. Como exemple tem-se *Top k-leader election in wireless ad hoc networks* [44].

2.5 Conclusão

Neste capítulo foram apresentados conceitos relacionados à Internet das Coisas (IoT), seus componentes básicos, as "coisas" ou objetos inteligentes. Tais objetos possuem identificadores únicos, capacidade de processamento e comunicação geralmente limitadas, e detectores (sensores) de contexto do ambiente e/ou são capazes de realizar ações sobre o ambiente (atuadores). Também foram descritos os domínios de aplicação da IoT, como: pessoal e residencial, empresarial, utilitários e móvel. Foi apresentada ainda a Internet das Coisas Móveis (IoMT), uma extensão da IoT que considera que os objetos podem ser movidos ou podem se mover autonomicamente pelo ambiente. Descreveu-se os meios de comunicação entre objetos e/ou dispositivos com ênfase no *Bluetooth*, perpassando pelo *Bluetooth Classic* até o *Bluetooth Low Energy*, descrevendo suas características e suas respectivas limitações.

Devido às restrições de memória, processamento e conectividade é necessário um *gateway* para coletar os dados de contexto dos objetos. Desse modo, foi descrito o funcionamento e a arquitetura do *middleware* M-Hub, que foi concebido no âmbito do projeto *ContextNet*. Ele fornece um arcabouço para realizar coleta, processamento das informações dos objetos e encaminhamento para as aplicações consumidoras.

Foram apresentados conceitos e características dos algoritmos de eleição de líder, descrevendo assim sua importância em um sistema distribuído. Foram descritos algoritmos empregados em diversos cenários, passando pelos algoritmos clássicos até os algoritmos mais recentes, de acordo com sua taxonomia.

3 Trabalhos Relacionados

Muita pesquisa tem sido desenvolvida nas áreas de IoT e de redes de sensores sem fio para fornecer abordagens para balanceamento de carga. Neste capítulo, descrevemos alguns trabalhos relacionados que desenvolveram algoritmos de eleição para sistemas distribuídos móveis, explorando as características dos principais algoritmos para eleição de líder apresentados na literatura. Foram descritos diversos parâmetros utilizados para determinar qual o melhor líder nos cenários explorados em cada trabalho.

No entanto, apesar da variedade de funcionalidades que serão apresentadas pelos algoritmos analisados, nenhum deles foi concebido tendo por objetivo a disponibilização de funcionalidades que permitam a eleição de K líderes para o cenários de IoMT.

3.1 *Load Balancing in P2P Smarthone Based Distributed IOT Systems* [15]

O autor apresenta um esquema de balanceamento de carga *Peer-to-Peer* (P2P) para distribuir tarefas em *clusters* de *smartphones* em cenários específicos. O autor apresenta o LoadIoT, um esquema de balanceamento de carga para IoT. O LoadIoT fornece uma maneira de diminuir o número de mensagens trocadas para aquisição de dados e assim poupar energia em dispositivos móveis.

Um *Smart Head* é eleito para cada serviço e tarefa identificada pelo sistema. A atribuição de tarefas depende do resultado do algoritmo de balanceamento de carga. O algoritmo proposto utiliza uma lista de parâmetros de entrada, como: estado da bateria (carregando, descarregando, crítico), energia residual (porcentagem de energia restante), carga do processador, número de tarefas em execução. Contudo o autor usa basicamente o estado da bateria do dispositivo (parâmetro X_0), como pode ser visto na Tabela 3.1. Em caso de empate o algoritmo utilizará o próximo parâmetro como critério para o desempate.

Tabela 3.1: Lista de parâmetros.

Parâmetro	Definição
P1	Estado da Bateria: carregando, descarregando ...
P2	Uso da Bateria: porcentagem de uso da bateria.
P3	Processador: atividade dos núcleos
P4	Atraso de Resposta
P5	Número de Tarefas em Execução

Uma eleição acontece quando um nó deseja acessar algum serviço e não há nenhum *Smart Head* para aquele serviço. O autor utiliza dois modos de balanceamento de carga: o modo proativo, que aciona o processo de eleição em um intervalo de tempo predefinido, neste caso, a cada 5 minutos. E o modo reativo, que inicia o processo de eleição, somente, quando um *Smart Head* é necessário. Uma nova eleição acontece em dois casos: quando a bateria restante do Smart Head chegou ao limite ou o serviço o qual ele era responsável foi finalizado. O autor concentra seu estudo apenas no balanceamento de carga e não no posicionamento, ou seja, nos cenários de testes todos os nós são estacionários.

3.2 *Leader Election in Opportunistic Networks* [18]

Os autores introduziram duas abordagens para eleger um líder em redes oportunistas, capaz de coletar, processar e distribuir os dados coletados e transmiti-los para as aplicações consumidoras. A primeira abordagem implica que cada nó dissemine uma candidatura, que é usada pelos nós receptores para calcular uma pontuação. O nó com maior pontuação torna-se o líder. Para definir a pontuação, os autores usam uma série de características como: confiança, centralidade, probabilidade de contato e latência.

O valor de confiança de um nó é baseado no número de entregas de mensagens bem sucedidas. O líder deve ser facilmente acessível aos nós em seu grupo e a centralidade é a medida dessa propriedade. A probabilidade utilizada na computação de pontuação de líder é baseada no histórico de contatos [9] com base em observações feitas por Ciobanu et al. [8] na qual os dispositivos, como *smartphones*, são

transportados por seres humanos, possibilitando assim determinar alguns padrões de mobilidade. E partir desses padrões é possível calcular a probabilidade de contato dos nós. A latência é um tempo entre o momento em que a candidatura foi gerada e o tempo que a mensagem foi recebida pelo nó. Esse valor é utilizado para garantir que as solicitações dos nós aos líderes cheguem o mais rápido possível, e os nós podem contar com uma resposta relativamente rápida do líder.

A segunda abordagem é bem similar a primeira, mas não é calculada a latência, tornado-a mais direta e simples. Os autores definem os líderes levando em consideração a posição relativa do nó em relação ao outros nós conectados a ele. Contudo, objetivo proposto neste trabalho de dissertação está interessada em definir o líder de acordo com posição do nó em relação ao objetos encontrados oportunisticamente no ambiente.

3.3 *Eventual Leader Election despite Crash-Recovery and Omission Failures* [21]

Os autores introduziram um algoritmo de eleição de líder para sistemas parcialmente síncronos com base na contabilidade das falhas sofridas em termos de computabilidade (recuperação de falhas) e comunicação (omissões) com o objetivo de selecionar um único processo que deverá coordenar as ações de um conjunto de processos. Uma punição é calculada para cada candidato com base em seus contadores de falhas e desconexões. O líder é o nó com a menor punição ou, no caso de empate, com o menor identificador de processo. Desse modo, é possível evitar que nós com altas taxas de falhas se tornem líderes, ou seja, a probabilidade de um nó se tornar líder é inversamente proporcional ao número de falhas.

Entretanto, em cenários onde os nós possuem diferentes graus de mobilidade, os nós com pouca mobilidade ou estacionários serão privilegiados em uma eleição, pois tem menos chances de serem punidos devido as falhas de comunicação. Essa solução diferencia-se do objetivo proposto neste trabalho de dissertação pois em cenários da IoMT em que há muita mobilidade é interessante permitir que qualquer nó seja elegível a se tornar líder.

3.4 *Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks* [51]

Os autores apresentaram um algoritmo de eleição de líder baseado no algoritmo de detecção de término de Dijkstra-Scholten [17]. O algoritmo suporta mobilidade dos nós e, consequentemente, topologias aleatórias, incluindo particionamento e agrupamento de um nó ou conjunto de nós.

O algoritmo suporta eleições múltiplas e concorrentes. Cada nó pode iniciar apenas uma eleição por vez. A mensagem de eleição contem o identificador do nó e o valor do nó. O valor é incrementado toda vez que um nó inicia um eleição. Um nó, ao receber uma mensagem de eleição verifica se o valor do nó que iniciou outra eleição é superior ao seu. Em caso positivo, ele se retira da disputa pela eleição e define o nó como líder. Por exemplo, na Figura 3.1 (a), o nó G envia uma mensagem de eleição com índice de computação 3 para D e para A. Ao receber esta mensagem , o nó A para de participar de sua computação atual ("2,B"), configura seu índice de computação como "3, D", conforme mostrado na Figura 2 (b), e propaga a mensagem de Eleição recebida para os nós B e C.

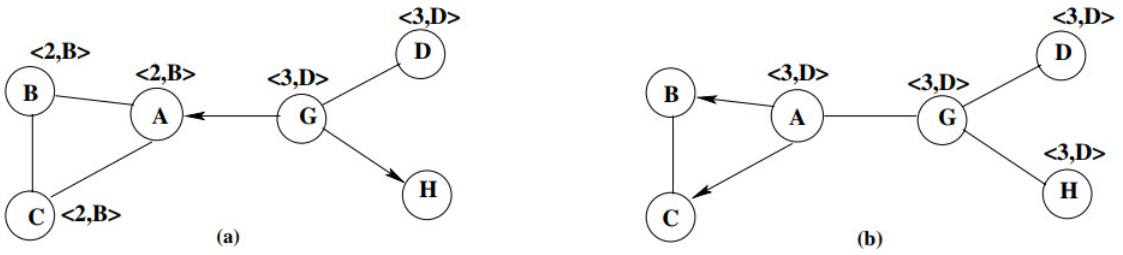

Figura 3.1: Controle de múltiplas eleições.

Embora, os nós possam ser móveis antes e depois da eleição, presume-se que a topologia da rede permaneça estática durante o período de eleição, diferente da solução que é proposta nesse trabalho, a qual permite mobilidade dentro da cobertura da rede em que estão os M-Hubs e M-Objs.

3.5 Um algoritmo distribuído para eleição de líderes de clusters semânticos em redes de sensores sem fio [30]

Neste trabalho os autores propõem um algoritmo para eleição de líderes de *clusters* semânticos de forma distribuída à base de um motor de inferência nebulosa (*Fuzzy*). A ideia geral da clusterização é organizar a rede em grupos (*clusters*), compostos por nós comuns e por um nó líder, ou *Cluster-Head*, o qual coleta os dados sensoriados pelos nós comuns e os encaminha até o sorvedouro através de uma comunicação multissalto. Como, em geral, os nós internos aos clusters estão mais próximos do seu líder do que do sorvedouro, a rede economiza energia. Este trabalho tem como objetivo prolongar o tempo de vida de nós sensores em uma RSSF, alimentados por baterias, de modo a contornar problemas na qualidade de entrega de dados da rede.

Este trabalho estende o SEMANTK [46] para aumentar a sobrevida dos líderes semânticos através de uma eleição eficiente. Enquanto no SEMANTK a eleição dos líderes semânticos leva em consideração apenas a quantidade de vizinhos semânticos dentro de um *cluster* físico, o algoritmo proposto pelos autores executa uma eleição com base no RSSI e energia residual usando um motor de inferência nebulosa.

O sistema nebuloso proposto combina as variáveis linguísticas de entrada "Energia" e "RSSI" para auxiliar na tomada de decisão relativa à eleição mais adequada do líder semântico. Como neste trabalho não há mobilidade dos nós sensores, suas distâncias não mudam, o que favorece a eleição de candidatos mais próximos ao sorvedouro, em detrimento do valor da energia residual dos nós. Para contornar essa situação, atribuiu-se pesos aos valores da entrada linguística Energia, de modo, que a distância elevada (que corresponde a um RSSI baixo) não exclua a possibilidade de eleição de um dado dispositivo.

A base do motor de inferência nebulosa conta com nove regras que são compostas por dois antecedentes e um consequente, respectivamente: "Energia", "RSSI" e "Chance" (Figura 3.2). A saída deste sistema nebuloso é representada pela variável linguística Chance, que representa a probabilidade de um candidato ser eleito como líder de *cluster* semântico. Caso dois candidatos tenham a mesma chance, será eleito aquele que possuir a maior energia residual.

Regra	*Energia*	*Distancia*	*Chance*
1	Baixa	Perto	Pouco Pequena
2	Baixa	Média	Pequena
3	Baixa	Longe	Muito Pequena
4	Média	Perto	Muito Média
5	Média	Média	Média
6	Média	Longe	Pouco Média
7	Alta	Perto	Muito Grande
8	Alta	Média	Grande
9	Alta	Longe	Pouco Grande

Figura 3.2: Base de regras utilizada.

Com a estratégia de eleição proposta pelos autores, foi possível obter uma maior rotatividade de líderes nos cenários de teste. Os níveis de energia dos líderes tendem a cair de maneira equilibrada devido à associação de um peso à variável "Energia", possibilitando que os dispositivos mais distantes possuam chances equivalentes às dos dispositivos mais próximos ao sorvedouro. Isso mostra que a ponderação das variáveis de entrada ("Energia" e "RSSI") possibilitou um comportamento equilibrado independente das distâncias e localizações dos dispositivos nos cenários observados.

3.6 Discussão

A tabela 3.2 compara os diferentes mecanismos de eleição de líder sendo que alguns são voltados para casos genéricos e outros para a IoT apresentados nos trabalhos acima. O objetivo é identificar as limitações de cada trabalho. Os critérios adotados para a comparação são:

- **Tipo da Eleição**: utilizado a taxonomia apresentada na subseção 2.4, na qual os métodos podem ser: salto único, múltiplos saltos, hierárquicos e de K líderes. Um algoritmo pode se enquadrar em mais de um grupo;

- **Tipos de atributos**: utilização de um casamento de um ou mais atributos do dispositivo móvel (e.g., bateria, CPU, memoria) e atributos referentes ao serviço/objeto inteligente (e.g., RSSI, localização);

A tabela 3.2 sumariza proposta em relação à satisfação dos requisitos descritos.

Tabela 3.2: Comparativo entre os trabalhos relacionados

Eleição de Líder	Tipo de Eleição	Atributos
De Masi. [15]	Salto único	Dispositivo
Drãgan et al. [18]	Múltiplos Saltos	Dispositivo
Fernández et al. [21]	Múltiplos Saltos	Dispositivo
Vasudevan et al. [51]	Múltiplos Saltos	Dispositivo
Hermeto et al. [30]	Múltiplos Saltos	Dispositivo Objeto Inteligente

3.6.1 Discussão dos Trabalhos Relacionados em Relação ao Tipo de Eleição

A comparação mostra que os trabalhos relacionados adotam diferentes tipos de Eleição, como apresentado na Seção 2.4. O trabalho proposto por DeMasi [15] emprega um algoritmo baseado em salto único, ou seja, é possível enviar diretamente uma mensagem para qualquer nó. Para tal, é necessário utilizar-se de uma infra-estrutura para permitir a comunicação. O autor utiliza o protocolo de comunicação MQTT para publicar serviços em um *broker* e os nós se subscrever no tópico de serviços para se candidatar. Contudo, o autor não especifica como foi feito o esquema de diretórios no *broker* e como é feita a reeleição de um líder, quando o mesmo falha.

Os trabalhos de Drãgan et al. [18], Fernández et al. [21] e Vasudevan et al. [51] empregam uma abordagem de eleição de líder baseada em múltiplos saltos. Os algoritmos propostos por estes autores são capazes de enviar mensagens para um nó em particular, por meio de nós intermediários. Contudo para enviar as mensagens do algoritmo de eleição, é necessário utilizar um algoritmo de roteamento.

Outro aspecto importante dos algoritmos de eleição de líder com múltiplos saltos é que eles não necessitam de uma infra-estrutura de redes para transmitir suas mensagens, ou seja, utilizam uma comunicação *peer-to-peer*. Por esse motivo, esses algoritmos devem ser capazes de suportar particionamentos e agrupamentos de um nó ou conjunto de nós. Como apresentado por Vasudevan et al. [51], o particionamento ocorre quando um nó, que une dois grupos, se move de modo que impossibilite a comunicação entre os grupos. Já o agrupamento é quando um novo nó funciona com um ponte de comunicação entre dois grupos distintos, formando assim um único grupo. O trabalho Hermeto et al. [30] apesar de utilizar uma eleição baseada em múltiplos saltos, não apresenta problemas de particionamento e agrupamento pois os nós são fixos.

3.6.2 Discussão dos Trabalhos Relacionados em Relação ao Tipos de Atributos utilizados

Como apresentado na Seção 2, os atributos utilizados no algoritmo de eleição de líder depende do cenário de aplicação e do que se deseja alcançar ao aplicá-lo. O trabalho proposto por DeMasi [15] tem como objetivo distribuir serviços para serem processados por outros dispositivos. Desse modo o autor utiliza um série de atributos relacionados ao dispositivo. Contudo, poderiam ser empregados atributos relacionados ao serviço, optimizando assim a escolha do nó responsável por processá-lo.

Os autores em Drãgan et al. [18] propuseram duas abordagens de eleição de líder com o objetivo de coletar, processar e distribuir os dados obtidos por outros dispositivos. Os autores utilizam um conjunto de atributos relacionadas com a interação entre os dispositivos. Diferentemente do trabalho apresentado nesta dissertação, os autores permitem que todos os dispositivos coletem os dados dos sensores, e elejam um dispositivo para concentrar os dados coletados e para transmitilos.

No entendimento dos autores em Fernández et al. [21] para que um nó se torne o líder, ele deve possuir um baixo número de falhas. Por esse motivo, os autores contabilizam o número de falhas (e.g., desconexões, omissões). Contudo, em cenários em que os nós se comunicam por meios de tecnologias de comunicação sem fio e

que apresentam mobilidade, nós que sejam estacionários ou com pouca mobilidade poderão ter mais chances de ser tornarem líderes, pois não serão punidos devido às falhas de comunicação.

No trabalho de Hermeto et al. [30] os autores apresentam um algoritmo de eleição de líder com base em lógica *Fuzzy* em uma RSSF. Para gerar as regras do motor de inferência *Fuzzy* foi utilizada a energia residual do dispositivo e a RSSI em relação ao sorvedouro.

3.7 Conclusão

No presente capítulo foram exploradas as características dos principais algoritmos para eleição de líder apresentados na literatura. Foram descritos os principais parâmetros utilizados spara se determinar qual o melhor líder para os cenários explorados em cada trabalho.

No entanto, como visto neste capítulo, apesar das variedades de funcionalidades apresentadas pelos algoritmos analisados, nenhum deles foi concebido tendo por objetivo a disponibilização de funcionalidades que permitam a eleição de líder para o cenários de IoMT. Desse modo, a implementação desta solução foi realizada pela análise do cenário proposto neste trabalho, utilizado-se do conhecimento de abordagens dos trabalhos apresentados no presente capítulo. A solução proposta, diferentemente de outras soluções de eleição de líder citadas anteriormente, fornece um suporte a escolha de um líder/sub-líder para os diversos objetos descobertos no ambiente.

4 Solução Proposta

Visando adiciona ao *middleware* M-Hub novas funcionalidades, com o objetivo de permitir a comunicação entre M-Hubs, com a finalidade de coordenar a ações entre os mesmos com o intuito de optimizar a coleta e distribuição do dados de M-Objs.

Nesse contexto, este trabalho de mestrado se concentra na implementação dos mecanismos de comunicação e coordenação de um algoritmo de eleição de líderes que seleciona os M-Hubs com base em um conjunto de parâmetros relacionados as capacidades computacionais dos M-Hubs e a intensidade de sinal recebida dos M-Objs. Os componentes da arquitetura geral da solução são apresentados e descritos na Seção 4.1 a seguir. Nela é possível identificar os componentes que implementam as funcionalidades necessárias para se chegar ao objetivo desse projeto.

4.1 Arquitetura Geral da Solução Proposta

A arquitetura proposta, como pode ser visto na Figura 4.1, contempla novos componentes incluídos para dar suporte ao problema abordado neste trabalho. Esta é acoplada ao M-Hub, contudo como trabalho futuro pode-se remodelar a arquitetura para que seja possível torna a solução proposta independente do M-Hub. Os componentes da camada M-Hub já foram discutidos na Seção 2.2. Passamos a detalhar os novos componentes da infraestrutura pertencentes a solução proposta:

- **Conteiner de Objetos**: é um contêiner dos objetos descobertos pelo M-Hub. Este módulo gerencia os objetos, atualizando os dados de um objeto que já esteja presente no contêiner e inserindo-o caso não esteja no contêiner. Este componente também é responsável por remover os objetos cujos os dados não foram recebidos durante um certo período de tempo, implicando que o M-Hub não está na área de transmissão do objeto;

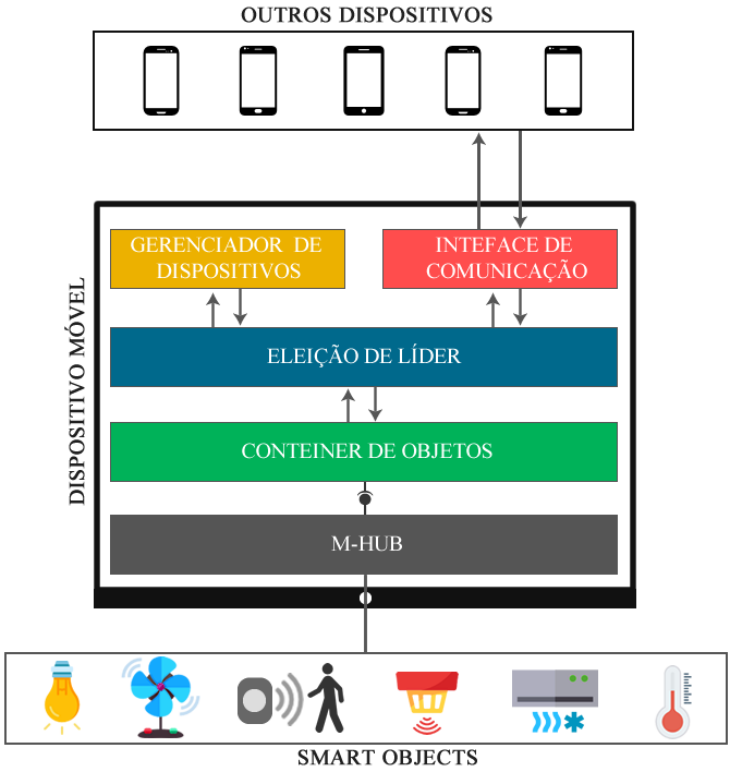

Figura 4.1: Arquitetura da solução proposta.

- **Gerenciador de Dispositivos**: contém as informações do dispositivo móvel. O objetivo principal é fornecer ao módulo de eleição de líder os parâmetros necessários para calcular o *score* utilizado para a definição do líder durante o processo de eleição de líder;

- **Interface de Comunicação**: permite que um M-Hub se comunique com outros M-Hubs através da troca de um conjunto de mensagens. O tipo de mensagem enviada e as informações nele contidas são decididas pelo módulo de eleição de líder. A comunicação entre os M-Hubs é realizada por meio da tecnologia IEEE [1] [2] 802.11, porém outros meios de comunicação poderão ser acrescentados em trabalhos futuros;

- **Eleição de líder** é um componente composto por vários componentes que interagem para permitir que o M-Hub chegue a uma decisão sobre participar ou não de um processo de eleição de líder, e se deve iniciar um processo com base no estado atual do sistema. O núcleo deste modulo é o algoritmo de eleição de líder. Este módulo é responsável por gerar as ações e a comunicação com outros

[1] Instituto de Engenheiros Eletricistas e Eletrônicos
[2] `https://www.ieee.org/`

M-Hubs. Também é responsável por invocar as funções que verificam os tipos de mensagens recebidas, assim como quais mensagens devem ser empregadas para responder tais mensagens. Além disso, ele é responsável por verificar se o **Conteiner de Objetos** possui os objetos contidos nas mensagem trocadas entre os M-Hubs. Por fim, este componente é responsável por solicitar as informações sobre o dispositivo junto ao **Gerenciador de Dispositivos** e por combina-los com os dados dos objetos de modo a poder realizar o cálculo do *score*.

4.1.1 Interação entre os Componentes

Como descrito na Seção 1.1, o escopo desse trabalho de mestrado é o desenvolvimento de mecanismos para proporcionar a coordenação entre M-Hubs para otimizar a coleta e distribuição de dados a serem integrados ao *middleware* M-Hub.

Uma das formas que os componentes poderiam interagir para atingir sucesso, está descrito no diagrama de sequência representado pela Figura 4.2. O componente S2PA do M-Hub inicializa as tecnologias implementadas, no caso concreto a tecnologia BLE (BLETechnology), o serviço S2PA passa a desencadear periodicamente o processo de descoberta de cada tecnologia.

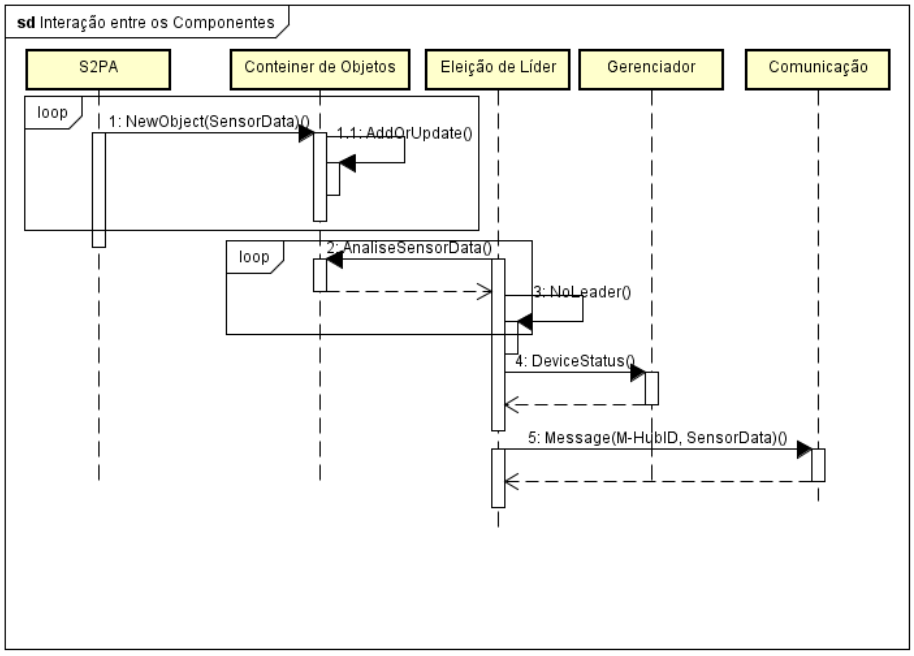

Figura 4.2: Diagrama de Sequência de interação entre os componentes.

Ao encontrar M-Objs durante essa fase e extrair as suas informações (SensorData) e as envia para o **Conteiner de Objetos** (passo 1). Que por sua vez, adiciona ou atualiza as informações do M-Obj (passo 1.1).

O módulo de **eleição de líder** por sua vez analisa (passo 2), periodicamente, os objetos contidos no **Conteiner de objetos**. Dessa forma, ele pode decidir qual mensagem será transmitida para os demais M-Hubs em sua proximidade (passo 3). O **Gerenciador de Dispositivo** envia, quando solicitado, as informações do dispositivo para o **eleição de líder** (passo 4). E por fim, as mensagens são transmitidas e recebidas pelo módulo de **interface de comunicação** (passo 5).

4.2 Algoritmo de Eleição de Líder Proposto

A solução proposta neste trabalho é um algoritmo de eleição de K líderes para determinar os melhores *gateways* nos cenários da IoMT. Um nó (i.e., um M-Hub) se comunica apenas trocando mensagens e cada nó é conectado por um enlace de comunicação bidirecional. Em relação ao tempo, assume-se um sistema parcialmente síncrono (i.e., há limites superiores no tempo de processamento e no atraso da comunicação de mensagens). As premissas da solução proposta são:

1. Cada nó possui um identificador único e fixo;
2. Cada nó tem um valor de prioridade (*Score*) associado a ele. A prioridade de um nó indica sua atratividade como líder e pode levar em consideração qualquer atributo relacionado ao desempenho, como o nível de bateria, as capacidades computacionais (e.g., memória e processamento), assim como a intensidade do sinal entre ele e o M-Obj;
3. Os canais de comunicação devem ser bidirecionais e a entrega da mensagem só é garantida quando o remetente e o destinatário estiverem conectados;
4. Os nós podem ser heterogêneos em relação aos atributos relacionados ao desempenho;
5. Os objetos inteligentes podem ser fixos ou móveis (i.e., M-Objs). Eles podem ser heterogêneos e terem diferentes propósitos, como: sensores de temperatura,

luminosidade e umidade. Eles podem ter diferentes interfaces de rede e, consequentemente, variados alcances de transmissão;

6. A mobilidade dos nós pode resultar em topologias arbitrárias, incluindo particionamento e agrupamento. Além disso, nós e objetos podem falhar a qualquer momento. As falhas podem ser causadas por: (1) problemas de comunicação (e.g., conectividade sem fio fraca ou intermitente), e (2) componentes de hardware (i.e., defeito nas antenas de comunicação dos objetos, quedas de energia que ocasionam o desligamento). As falhas podem ser permanentes, mas os M-Hubs e objetos podem também serem recuperados de uma falha e voltarem a operar.

4.2.1 Tipos de Mensagens

Para realizar o processo de eleição de líder são utilizados os seguintes tipos de mensagens:

- ELECTION (M-HubID, *ElectionList*(M-ObjID, *Score*)): mensagem para informar o início de um novo processo de eleição de líder:
  - M-HubID – string *Universally Unique Identifier* (UUID) de 128 bits, gerada a cada início do aplicativo.
  - *ElectionList* – lista contendo as seguintes informações: (1) o identificador do M-Obj, (2) *Score*;
- ALIVE (*LeaderTupleToPropagate*): a mensagem ALIVE é enviada pelo nó líder em intervalos de tempo pré-definidos. Dessa forma é possível detectar se o nó líder sofreu uma falha ou perdeu conexão com o um determinado objeto. Ela também é utilizada para informar aos demais M-Hubs quem são os novos líderes ao final de um processo de eleição.
  - *LeaderTupleToPropagate* – lista contendo os objetos e seus respectivos líderes.
- PENDING (*ListObjs*): mensagens utilizadas para solicitar informações sobre o líder, e também direcionadas ao líder de um determinado objeto ou conjunto de objetos, solicitando respostas por meio de mensagens (ALIVE) que eventualmente não tenham sido recebidas.

4.2.2 Descrição do Algoritmo

O algoritmo deve selecionar os dois melhores *gateways*, um líder e um sub-líder, para cada objeto descoberto. O líder e o sub-líder devem ser os nós com os maiores *Scores* de um determinado conjunto de nós em relação a um determinado objeto. O *Score* é determinado pela posição do nó em relação ao objeto e pelos parâmetros referentes às capacidades computacionais do nó. O objetivo do sub-líder é manter mais de um nó como candidato para se tornar líder. Desse modo, caso o líder não esteja acessível por algum motivo, o sub-líder se tornará o novo líder, evitando-se a necessidade por uma nova eleição.

O algoritmo proposto é dividido em três partes: (i) inicialização e descoberta de objetos, (ii) funções periódicas e (iii) eleição de líder. A primeira parte do algoritmo consiste na inicialização de um conjunto de variáveis que o nó utiliza para guardar informações sobre si e sobre os objetos encontrados.

O serviço de descoberta dos objetos é disponibilizado pelo M-Hub através do componente S2PA. Toda vez que um objeto é encontrado, ele é inserido no **Conteiner de objetos**, e o marca como sem líder. Todo objeto contido neste **conteiner** recebe um *timer*, valor pré-definido que permite controlar o tempo de chegada das mensagens do tipo ALIVE. Caso uma mensagem esperada do líder de um determinado objeto não seja recebida no tempo pré-estabelecido, significa que o objeto não possui um líder ou alguma falha pode ter acontecido.

A segunda parte do algoritmo inclui três funções que são executadas periodicamente e tratam a recepção de mensagens. A função **onAlive**, trata do envio das mensagens ALIVE. As mensagens ALIVE e PENDING recebidas são processadas na função **receiveAliveAndPending**. Primeiramente, é verificado se a mensagem ALIVE recebida contém algum objeto descoberto pelo nó que a recebeu. Em seguida, atualiza as informações referentes aos objetos. Por fim, reinicia o *timer* associado aos objetos.

Caso uma mensagem ALIVE não seja recebida no tempo pré-estabelecido, uma mensagem PENDING é enviada ao líder daquele determinado objeto. As mensagens PEDING são processadas pelo líder na função **receiveAliveAndPending**. Se o remetente não receber uma resposta do líder, o remetente assumirá que o líder sofreu uma falha ou movimentou-se de modo a perder conectividade com os objetos.

Desse modo, o remetente verifica se ele é o sub-líder, tornando-se o novo líder, evitando a sobrecarga de uma nova eleição. Caso contrário, envia um mensagem PENDING e, não obtendo resposta, o nó inicia um novo processo de eleição.

A função **receiveElection** trata as mensagens ELECTION. Um M-Hub, ao receber uma mensagem desse tipo, calcula o *Score* para cada objeto contido na mensagem e que esteja no **Conteiner de objetos**, e os insere na **ElectionList**. A mensagem ELECTION é enviada para o nó que iniciou o processo. Se o nó não conhece nenhum objeto contido na mensagem, a mesma é ignorada. Para determinar o *Score*, algumas características devem ser levadas em consideração. O *Score* do líder para cada objeto é calculada usando a fórmula 4.1:

$$\text{Score} = \text{W}_r * \text{V}_r + \text{W}_b * \text{V}_b + \text{W}_c * \text{V}_c \tag{4.1}$$

A notação dos elementos é a seguinte: *Wr*, *Wb*, *Wc* são pesos para, respectivamente, o Indicador de Intensidade de Sinal Recebido (*Received Signal Strength Indicator* - RSSI), o nível da bateria e do processador (cpu). Os pesos são empregados para gerar uma normalização dos valores de cada parâmetro, de modo a definir para cada um, um grau de importância diferente para determinar o melhor *gateway*. Os valores de *Vb*, *Vc* são valores para as propriedades acima mencionadas de um nó e *Vr* é o valor do RSSI.

Como os objetos estão distribuídos no ambiente, o nível de conectividade de um nó pode variar de acordo com sua posição. Uma característica importante que o líder deve ter é acesso fácil ao objetos em disputa. Assim, o RSSI é usado para determinar os nós mais acessíveis. O RSSI é representado por um valor negativo dado em dBm: quanto mais próximo ao valor zero, mais acessível é o nó.

O valor de RSSI dos canais entre os M-HUbs e M-objs oscila bastante com o tempo. Dessa forma, é necessário calcular a média dos dados de RSSI ao longo do tempo. Para se calcular o valor do RSSI foi utilizado o cálculo da média móvel. A fórmula 4.2 apresenta o cálculo da média móvel:

$$\text{M}_{\text{RSSI}} = \alpha * \text{V}_{\text{RSSI}} + (1 - \alpha) * \text{M}_{\text{RSSI}} \tag{4.2}$$

em que M_{RSSI} representa a média móvel do RSSI, o valor de V_{RSSI} indica o valor instantâneo do RSSI, ou seja, o último valor do RSSI obtido, e o $alpha$ é uma constante que representa um valor de ponderação , ou seja, um peso que indica a importância relativa do valor instantâneo em relação à média histórica. Dessa modo, o valor de $alpha$ foi definido em 0.7, pois este valor permite que o valor da media seja projetado de forma mais gradual em relação ao último valor de RSSI obtido, ou seja, o valor do V_{RSSI}.

O valor de W_r, Formula 4.1, é igual a 5, pois considera-se a característica mais importante para escolher um líder, uma vez que deseja-se que os nós com maior acessibilidade para o objeto tenham maiores chances de se tornarem um líder. O *Vr* é valor que vária entre 0 e 5. A fórmula 4.3 é utilizada para calcular o valor de W_r:

$$V_r = \frac{-30 W_r}{|M_{RSSI}|} \tag{4.3}$$

em que M_{RSSI} é o valor médio da intensidade do sinal recebida e -30 é o valor do RSSI utilizado como referência. *Vb* e *Vc* são características relacionadas ao nó. A Formula 4.4 mostra como é calculado o valor de *Vb*.

$$V_b = \frac{W_b * \text{RemainBattery}}{100} \tag{4.4}$$

em que, o valor W_b é igual a 3 e a vida útil da bateria é obtida a partir do nó, W_c está relacionado ao valor de carga da CPU, ou seja, a quantidade de processamento livre do dispositivo (medido em valores percentuais) e é igual a 2. O valor de V_c é calculado da seguinte forma, Formula 4.5 :

$$V_c = \frac{W_c * \text{CPULoad}}{100} \tag{4.5}$$

O valor do *Score* é calculado para cada objeto encontrado e varia entre 0 e 10. Ressalta-se que, um conjunto de nós seleciona um líder com base em seu estado e informações recolhidas sobre o estado de um objeto. Um nó é considerado líder se (i) estiver conectado com o objeto e (ii) tiver o maior *Score*. O **Conteiner de objetos** gerencia cada objeto, atribuindo seus respectivos líderes. Ao término do processo de eleição de líder, os líderes dos objetos podem ser alterados.

A terceira parte do algoritmo trata do processo de eleição de líder. O processo ELECTION (algortimo 1) contém a implementação básica do mecanismo de eleição. O procedimento verifica todos os objetos sem um líder eleito e envia uma mensagem ELECTION para todos os nós conectados a ele. O nó que iniciou o processo de eleição aguarda durante um *timer* para que os outros nós enviem suas mensagens ELECTION. Uma variável temporária *TempLeaderList* é criada para receber a mensagem de eleição dos outros nós. Por fim, o M-Hub que iniciou o processo de eleição calcula o *Score* para cada objeto.

Algoritmo 1: Election()

```
1 Enviar (Election) em Broadcast;
2 repita
3     TempLeaderList = Recebe (Election);
4 até TimerElection >= 0 ;
5 leaderListToPropagate = getBestLeaders(TempLeaderList, Hub);
6 Enviar (ALIVE(leaderListToPropagate));
```

Os critérios de seleção do líder são modularizados pela função **GetBestLeaders**, linha 5 do algoritmo 1, a qual analisa as informações sobre todos os nós que enviam candidaturas, assim como as informações do nó que iniciou o processo, retorna os nós com os maiores *Scores* e os insere na lista *leaderListToPropagate*. Finalmente, a mensagem ALIVE é propagada para todos os nós para informar o novo líder de um objeto ou um conjunto de objetos.

4.3 Aspectos de Implementação

A implementação do algoritmo proposto foi escrita em Java para a plataforma Android. O código usa as chamadas para a API do Android para obter informações utilizadas para realizar o calculo do *score* dos objetos durante o processo de eleição de líder. A descoberta dos objetos é realizada pelo M-Hub.

4.3.1 Descoberta dos Objetos Inteligentes

O serviço de descoberta dos objetos é disponibilizado pelo M-Hub através do componente S2PA. O S2PA é responsável pela descoberta, conexão e comunicação com os M-Objs. Este componente define uma API que fornece uma abstração para a comunicação com *smart objetcts* que utilizam tecnologias de curto alcance (e.g., *Bluetooth Low Energy* e *Bluetooth Classic*). Para a realização da troca de dados com os *smart objects* é necessária a utilização do *driver* do objeto que deverá ser fornecido pelo seu fabricante;

Toda vez que um objeto é encontrado, ele é inserido na **Conteiner de objetos** e são marcados como sem líder. Todo objeto contido no **Conteiner de objetos** recebe um *timer*, valor pré-definido que permite controlar o tempo de chegada das mensagens do tipo ALIVE. Caso uma mensagem esperada do líder de um determinado objeto não seja recebida no tempo pré-estabelecido, significa que o objeto não possui um líder ou alguma falha pode ter acontecido. Além do *timer* para controlar a chegada das mensagens ALIVE, há um *timer* para remover os objetos do **conteiner**. Desse modo é possível remover os objetos, caso o M-Hub saia de sua área de alcance.

4.3.2 Mensagens

A implementação da troca de mensagens na solução é feita por meio de tecnologia *multicast*. *Multicast* é baseado no tipo de comunicação um-para-muitos em que, um pacote enviado é recebido por muitas entidades. O receptor tem a responsabilidade de manipular os pacotes e determinar sua usabilidade. Desse modo, os nós (M-Hubs) são diretamente conectados a um ponto de acesso usando tecnologia IEEE 802.11.

As mensagens são trocadas em formato *JavaScript Object Notation* (JSON). O JSON é compacto e a análise é mais rápida que outras técnicas de serialização. A primeira letra da mensagem é o cabeçalho e permite que o algoritmo determine o tipo da mensagem. Um exemplo de uma mensagem ELECTION pode ser visto na Figura 4.3. Ela contém o *id* do M-Hub que iniciou o processo de eleição e uma lista de M-Objs com seus respectivos identificadores e *scores*.

```
e{ "ID":"babeaa18-6496-4bad-9b69-6fc373180547",
   "objectIDs":[ { "MID":"0C:F3:EE:0E:34:9D", "score":5.353914 },
    { "MID":"0C:F3:EE:0E:32:20", "score":5.2899466 },
    { "MID":"0C:F3:EE:0E:31:CE", "score":4.682632 }]}
```

Figura 4.3: Exemplo de Mensagem ELECTION

A Figura 4.4 é um exemplo de uma mensagem do tipo ALIVE, na qual contem o *id* do M-Hub e um lista de M-Objs na qual ele é o líder.

```
a{ "ID":"babeaa18-6496-4bad-9b69-6fc373180547",
   "objectIDs":[ { "MID":"0C:F3:EE:0E:34:9D",
      "liderID":"babeaa18-6496-4bad-9b69-6fc373180547"},
      { "MID":"0C:F3:EE:0E:32:20",
      "liderID":"babeaa18-6496-4bad-9b69-6fc373180547"},
      { "MID":"0C:F3:EE:0E:31:CE",
      "liderID":"babeaa18-6496-4bad-9b69-6fc373180547"}]}
```

Figura 4.4: Exemplo de Mensagem ALIVE

A Figura 4.5 é um exemplo de uma mensagem do tipo PENDING, a qual contém o *id* do M-Hub que está solicitando as informações do líder ou sub-líder de um ou de um conjunto de M-Objs.

```
p{ "ID":"babeaa18-6496-4bad-9b69-6fc373180547",
   "objectIDs":[ { "MID":"0C:F3:EE:0E:34:9D"},
                 { "MID":"0C:F3:EE:0E:32:20"},
                 { "MID":"0C:F3:EE:0E:31:CE",}]}
```

Figura 4.5: Exemplo de Mensagem PENDING

4.4 Conclusão

Neste capítulo foram abordados diversos aspectos do algoritmo de eleição de líder proposto para o contexto de IoMT. Foram mostradas as motivações, os componentes e suas principais funcionalidades. Foi visto que o desenvolvimento

do algoritmo foi motivado pela necessidade de se determinar o melhor *gateway* no cenário da IoMT para realizar a coleta de dados de contexto dos objetos inteligentes oportunisticamente descobertos no ambiente. A solução proposta foi desenvolvida sobre o M-Hub, um *middleware* que transforma *smartphones* pessoais móveis em *gateways* IoT. Foram apresentadas os atributos utilizados, assim como as formulas utilizadas para se determinar o melhor *gateway*.

5 Avaliação

Este capítulo apresenta as avaliações realizadas para validar a proposta deste trabalho, na qual se deu, um caso de uso, um experimento para analisar o tempo de detecção e recuperação de falhas, e por fim, a complexidade da solução proposta baseado na troca de mensagens.

Para a realização do experimento foi instalada uma aplicação teste com o algoritmo de eleição de líder proposto, na qual foram utilizados dois dispositivos móveis, como visto na Tabela 5.1, além de três objetos (sensores). Os identificadores (MAC) dos objetos foram inseridos manualmente no **conteiner de objetos** como uma forma de isolar o sistema de objetos que não farão parte do estudo de caso. Os dispositivos móveis executando o M-Hub foram isolados em uma rede *Wi-Fi* própria com um *laptop* Intel i7 de 7a geração com 8 GB de memória RAM, possuindo 8 núcleos de processamento de frequência 2,4 GHz e com o sistema operacional Linux 18.04, configurado como *access point* (Figura 5.1) de modo a evitar ao máximo interferências de fatores externos no processo de avaliação. Os M-Objs foram conectados aos dispositivos móveis por meio da tecnologia *bluetooth LE* no modo *broadcast*.

Tabela 5.1: Informações dos Dispositivos Móveis

Construtor	Modelo	Sistema Operacional	Tipo
Samsung	Galaxy S8	Android 8.0	Smartphone
Samsung	Galaxy S6	Android 6.1	Smartphone

Como a API do Android não dá acesso à configuração de hardware, não é possível modificar a taxa de emissão de energia (TX) da antena *Wi-Fi* para limitar a potência de transmissão [5].

Foram utilizados 3 *beacons* da marca *Radius Network*, todos com nível de bateria em 100%, distribuídos no ambiente como apresentado na Figura 5.2. A Tabela 5.2 apresenta como os *beacons* estavam configurados durante a execução. O protocolo utilizado foi o *iBeacon*. A *taxa de anúncio* define a frequência em que os pacotes de

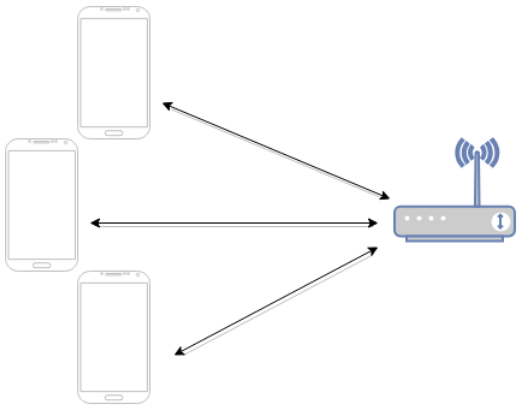

Figura 5.1: Fluxo de dados entre os M-hubs.

publicidade eram enviados. Já o *Potência transmissão* está associado à potência do sinal, indicando o poder de transmissão do *beacon BLE*.

Tabela 5.2: Parâmetros de configuração dos *beacons* utilizados no cenário de teste

Parâmetro de Configuração	**Valor**
Protocolo de Transmissão	*iBeacon*
Taxa de Anúncio	*3 anúncios por segundo*
Potência de Transmissão	- 3dBm

5.1 Cenário

O experimento foi realizado no Laboratório de Sistemas Distribuídos Inteligentes (LSDi) - UFMA, onde foram instalados 3 M-Objs em diferentes posições, disposto no ambiente como ilustrado na Figura 5.2. Como é possível perceber, cada M-Obj utilizado possui uma cor diferente para representar que se tratam de objetos diferentes, representados pelas cores: azul, verde e vermelho.

5.1.1 Caso de Uso: Validação de Eficácia

Este caso de uso teve como objetivo validar o uso da solução proposta em relação à eleição de líderes para os M-Objs descritos anteriormente. Os dispositivos móveis utilizados para este caso de uso estavam executando uma aplicação teste com

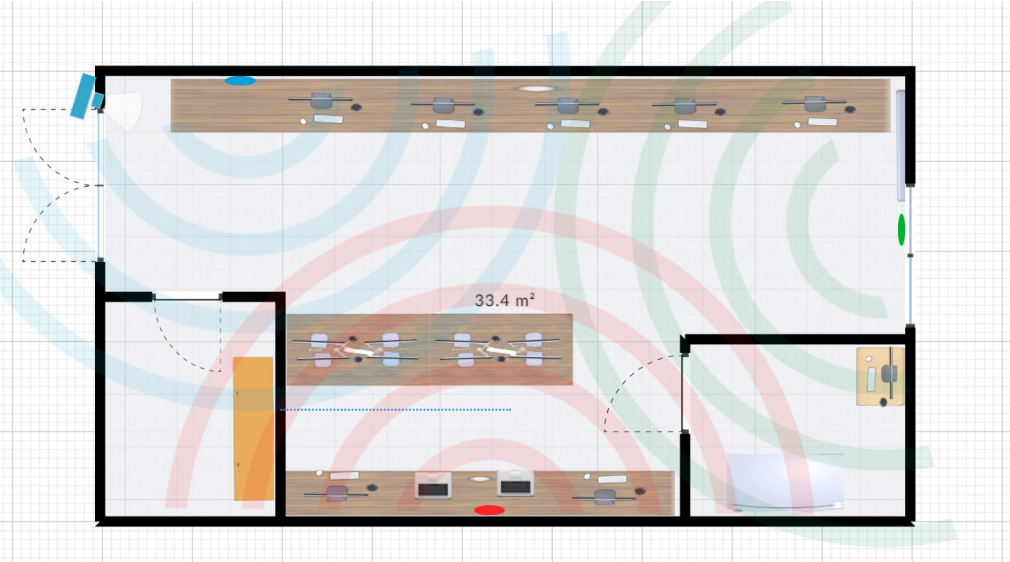

Figura 5.2: Cenário do Caso de Uso.

a implementação da solução proposta. Os dispositivos estão localizados em diferentes locais dentro do laboratório descrito, conforme pode ser visto na Figura 5.2.

A Figura 5.3 apresenta um exemplo de log gerado pelo algoritmo durante a execução do teste. Pode se observar a mensagem ELECTION enviada pelo dispositivo *Galaxy S6*, o qual iniciou o processo de eleição.

```
e{ "ID":"b1188fe9-94f3-4da9-b4b2-bfe5bfd498d5",
  "objectIDs":[
    { "MID":"0C:F3:EE:0E:32:20",
      "score":6.0026217 },
    { "MID":"0C:F3:EE:0E:34:9D",
      "score":5.92527 },
    { "MID":"0C:F3:EE:0E:31:CE",
      "score":6.0161295 } ]
}
```

Figura 5.3: Validação: Mensagem ELECTION.

Na Figura 5.4 mostra o *log* contendo a mensagem ELECTION de resposta enviada pelo dispositivo *Galaxy S8* ao dispositivo que iniciou processo de eleição.

Ao receber as mensagens ELECTION, o dispositivo que iniciou o processo de eleição executa o método *getBestLeaders()* para determinar os líderes para os M-

```
e{ "ID":"cd5cf42f-e639-4d52-a737-7b0ffd662f5b",
  "objectIDs":[
    { "MID":"0C:F3:EE:0E:32:20",
      "score":5.0759764 },
    { "MID":"0C:F3:EE:0E:34:9D",
      "score":5.1027074 },
    { "MID":"0C:F3:EE:0E:31:CE",
      "value":5.424999 }]
}
```

Figura 5.4: Validação: Mensagem ELECTION de resposta.

Objs em disputa. Por fim, uma mensagem ALIVE (Figura 5.5) com o resultado do algoritmo de eleição de líder proposto é propagada para os dispositivos móveis próximos, informando esse fato aos demais líderes.

```
a{ "ID":"b1188fe9-94f3-4da9-b4b2-bfe5bfd498d5",
   "objectIDs":[
     { "MID":"0C:F3:EE:0E:32:20",
       "liderID":"b1188fe9-94f3-4da9-b4b2-bfe5bfd498d5"
       "subLiderID":"cd5cf42f-e639-4d52-a737-7b0ffd662f5b"},
     { "MID":"0C:F3:EE:0E:34:9D",
       "liderID":"b1188fe9-94f3-4da9-b4b2-bfe5bfd498d5"
       "subLiderID":"cd5cf42f-e639-4d52-a737-7b0ffd662f5b"},
     { "MID":"0C:F3:EE:0E:31:CE",
       "liderID":"b1188fe9-94f3-4da9-b4b2-bfe5bfd498d5"
       "subLiderID":"cd5cf42f-e639-4d52-a737-7b0ffd662f5b"}]
}
```

Figura 5.5: Validação: Resultado do Algoritmo de Eleição proposto.

5.2 Tempo de Detecção de Falhas e de Recuperação

A velocidade do algoritmo de eleição proposto de detectar e de recuperação após uma falha foi medida pela métrica tempo de detecção da falha (T_D) proposta por Chen et al. [7] e o tempo de detecção de recuperação (T_{DR}) de Ma et al. [37]. Intencionalmente, foi provocada uma falha no nó líder e, em seguida, mediu-se o tempo decorrido para cada nó detectar a falha e, depois, o tempo decorrido para cada nó detectar a presença de um novo líder.

Os parâmetros de tempo relacionados ao envio de mensagens de tipo ALIVE foi definido em 600 ms. O *timer* que controla o tempo de chegada das mensagens do tipo ALIVE foi definido em 1200 ms, ou seja, caso uma mensagem esperada não seja recebida dentro deste intervalo de tempo, será considerado que o líder sofreu alguma falha.

O experimento consistiu em 10 repetições de um ciclo de recuperação de falha, com 60 segundos entre uma falha e a recuperação. O tempo de detecção de falha (T_D) pode ser obtido pela diferença do tempo da falha do líder (t_c) e do tempo de detecção de falha (t_d), como pode ser visto na Formula 5.1:

$$T_D = t_d - t_c \tag{5.1}$$

De um modo muito semelhante, o tempo de detecção de recuperação foi calculado (Formula 5.2) considerando o tempo de recuperação t_r e o tempo de detecção de uma recuperação t_{dr} por um outro nó:

$$T_{DR} = t_{dr} - t_r \tag{5.2}$$

Os valores observadas no experimento foram plotados no *boxplot*. O *boxplot* ou diagrama de caixa é uma ferramenta gráfica que permite visualizar a distribuição e valores discrepantes (*outliers*) dos dados, fornecendo assim um meio complementar para desenvolver uma perspectiva sobre o caráter dos dados. Além disso, o *boxplot* também é uma disposição gráfica comparativa. As medidas de estatísticas descritivas como o mínimo, máximo, primeiro quartil, segundo quartil ou mediana e o terceiro quartil formam o *boxplot*.

Os quartis nada mais são que os percentis 25, 50 e 75, representando respectivamente o primeiro, segundo e terceiro quartil. Veja que o segundo quartil equivale ao percentil 50, valor em que pelo menos 50% da amostra está acima dele e pelo menos 50%, ou seja, o percentil 50 ou segundo quartil equivale à mediana.

Durante o experimento foram obtidas 10 amostras para avaliar o tempo do detecção de falhas e de recuperação, como pode ser visto na Figura 5.6. Os valores do primeiro quartil, segundo quartil (mediana) e o terceiro quartil podem ser observados no Tabela 5.3.

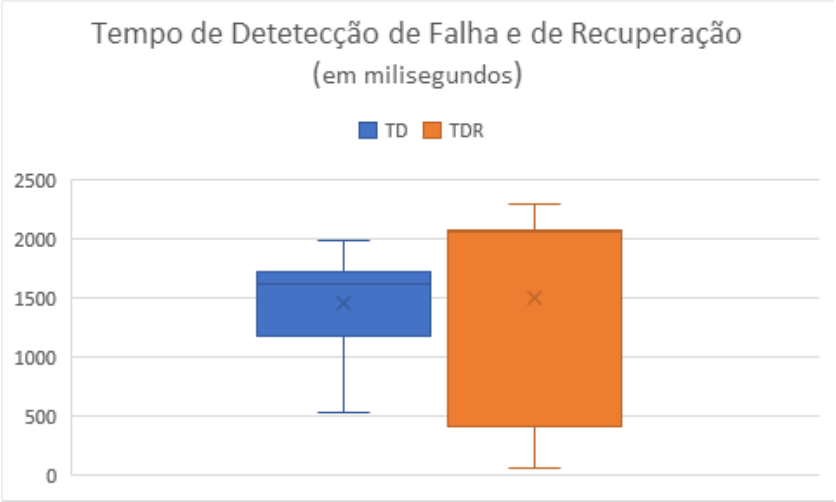

Figura 5.6: Tempo de Detecção de Falha e de Recuperação do Líder.

Tabela 5.3: Sumário dos Valores plotados.

Métrica	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
T_D (Figura 5.6)	533 ms	1173.5 ms	1615.5 ms	1720.25 ms	1983 ms
T_{DR} (Figura 5.6)	58 ms	410.75 ms	2055.5 ms	2075 ms	2297 ms

Conclui-se, a partir deste experimento, considerando o ambiente de uma rede local *Wi-Fi* 802.11, que o tempo médio que a aplicação leva para perceber a falha de um líder (em média 1457 ms) é adequado para que a mesma possa reagir prontamente a uma falha. Com relação ao tempo que o algoritmo leva para recuperar-se de uma falha (em média 1455,9 ms), é rápido o suficiente para que não haja uma grande perda de dados. Vale ressaltar que durante o experimento foi realizado o procedimento de reinicialização do sensor utilizado, e neste caso, durante a eleição os dois dispositivos utilizados se candidataram a líder resultado assim na eleição de um líder e um sub-líder. Sendo assim, foram obtidos os valores mínimos (T_D = 533 ms

e T_{DR} = 58 ms) apresentados no Tabela 5.3, comprovando assim a eficácia de manter mais de um nó como candidato para se tornar líder, evitando a necessidade de uma nova eleição.

5.2.1 Número de Mensagens Trocadas

O número de processos de eleição, dado um arranjo com n M-Hubs e m M-Objs, até m eleições podem ser realizadas de forma concorrente, e cada M-Hub pode iniciar no máximo uma eleição por vez. Desse modo, no pior caso, a quantidade de eleições geradas é de m, ou seja, uma eleição para cada M-Obj. O melhor caso de execução acontece quando um M-Hub inicia um único processo de eleição para um conjunto de m objetos, assim o número de eleições geradas será de 1.

Em relação ao número de mensagens trocadas, são geradas $n-1$ mensagens para propagar uma eleição e divulgar o líder. Desse modo, no pior cenário, o número de mensagens geradas é de $e(n-1)$, onde e é o número de processos de eleição concorrentes. No melhor caso, o número de mensagens trocadas é de $n-1$. Assim, o custo do algoritmo proposto é de $O(n^2)$.

5.3 Conclusão

Este capítulo apresentou avaliações por meio de um caso de uso de um protótipo da solução proposta, análise do tempo de detecção e recuperação de falha de um nó líder e por fim, a complexidade do algoritmo proposto baseado na troca de mensagens. Foi desenvolvido um protótipo que adota a solução proposta na qual atende aos objetivos especificados neste trabalho, a saber: descoberta de objetos inteligentes, permitir a comunicação entre os M-Hubs e definir o melhor *gateway* para cada M-Obj, utilizando atributos relacionados ao dispositivo móvel, assim como atributos relacionados aos M-Objs. Foi realizado um experimento para calcular o tempo de detecção de falhas e de recuperação de um nó líder. Com este experimento, foi possível determinar se o algoritmo proposto consegue detectar e posteriormente, recuperar-se, ou seja, elegendo um novo líder em tempo hábil. Desta forma, evita-se uma grande perda de dados no intervalo entre a detecção da falha e a eleição de um novo líder.

6 Conclusões e Trabalhos Futuros

Diversos algoritmos distribuídos exigem a existência de um nó central que desempenhe o papel de coordenador ou líder para realizar alguma função especial durante a execução de um processo. Em alguns nós distribuídos, é essencial a utilização de algoritmos de eleição de líder para determinar qual nó assumirá a função de líder. É necessário que todos os nós estejam de acordo com a escolha do novo líder e que as novas eleições possam ser realizadas para eleger substitutos de um líder.

A eleição do líder é um problema importante para construir sistemas distribuídos tolerantes a falhas. A eleição do líder é uma das partes mais críticas de qualquer sistema distribuído e também desafiadora, pois os dados são distribuídos entre diferentes nós que estão geograficamente separados. Para manter a coordenação entre os nós, um líder deve ser selecionado.

Este trabalho apresentou um levantamento do estado de arte em IoMT e eleição de líder. Porém, o ponto central deste trabalho foi a apresentação do algoritmo de eleição de líder para cenários da IoMT. Esse algoritmo, diferente dos outros trabalhos analisados, implementa uma abordagem de escolha de líder e sub-líder, levando em consideração a posição do nó em relação ao objeto inteligente. A ideia central de manter um sub-líder é evitar a sobrecarga de uma nova eleição quando o atual líder de um determinado objeto falhar. Este trabalho também descreveu um estudo de caso de utilização da solução proposta para validade sua eficácia e por fim, foi realizado um experimento para analisar o tempo de detecção e recuperação de falhas sofridas pelo nó líder.

As principais contribuições deste trabalho são:

- A investigação do estado da arte em relação à abordagem de algoritmos de eleição de líder para o apoio ao desenvolvimento de um algoritmo para ambientes da IoMT. Através desta investigação, ficou evidenciada a escassez de mecanismos específicos para o cenário no qual este trabalho de mestrado está inserido. Contudo, foi possível conhecer e entender o funcionamento de outras abordagens, que possibilitaram o desenvolvimento deste trabalho.

- O desenvolvimento de um algoritmo de eleição de líder e sub-líder para o cenário da IoMT, utilizando como base o *Middleware* M-Hub. O algoritmo proposto utiliza uma abordagem baseada em salto único, que permite que todos os nós possam se comunicar diretamente, e emprega uma abordagem baseada em N líderes, ou seja, uma única eleição pode eleger vários líderes para diferentes processos ou serviços, no caso deste trabalho, diversos M-Objs.

A partir deste trabalho inicial, identificamos algumas possibilidades de trabalhos futuros que poderiam ser desenvolvidos, tais como:

- A solução proposta não consegue realizar a troca de líder nos casos em que os M-Hubs movimentam-se muito rapidamente, pois pode haver a perda de conectividade, requerendo uma execução do algoritmo também muito rápida. Objetivando propor soluções para essas limitação, pretende-se explorar o uso de Processamento de Eventos Complexos (*Complex Event Processingn* (CEP)) [10, 36] para processar os dados oriundos dos objetos. Desse modo, poderá ser possível determinar a movimentação de um nó, e que ele está próximo a se desconectar de um objeto.

- Uma segunda limitação do algoritmo é não considerar mecanismos de segurança no processo de eleição de líder e coleta de dados dos M-Objs. Devido à comunicação ser feita por meio de tecnologias sem fio, algum usuário mal intencionado pode simular um objeto, adulterando seus parâmetros, ou mesmo provocar um ataque de negação de serviço.

- Devido ao fato do algoritmo proposto ser executado em dispositivos móveis pessoais heterogêneo, consequentemente, se um nó tem uma maior probabilidade de acessar os recursos necessários para a execução de uma tarefa, este nó poderá alocar mais tarefas. Contudo, se muitas tarefas estiverem alocadas em um determinado conjunto de nós, as tarefas exigirão muito mais tempo para acessar os recursos necessários para sua execução. Desse modo, é necessário aplicar o processo de balanceamento de carga [31, 32] em tempo de execução entre os nós móveis [49] disponíveis, conectados via uma conexão sem fio, de modo transparente, dinâmico e, principalmente, o balanceamento de carga deve ter um impacto mínimo sobre o desempenho do sistema.

Referências Bibliográficas

[1] Arik gabbai and kevin ashton. kevin ashton describes the internet of things. https://www.smithsonianmag.com/innovation/kevin-ashton-describes-the-internet-of-things-180953749/. Acessado em 17/03/2018.

[2] A. Aijaz and A. H. Aghvami. Cognitive machine-to-machine communications for internet-of-things: A protocol stack perspective. *IEEE Internet of Things Journal*, 2(2):103–112, 2015.

[3] A.-V. Anttiroiko, P. Valkama, and S. J. Bailey. Smart cities in the new service economy: building platforms for smart services. *AI & SOCIETY*, 29(3):323–334, Aug 2014.

[4] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[5] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, pages 280–293. ACM, 2009.

[6] A. Biswas and A. Dutta. A timer based leader election algorithm. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 432–439. IEEE, 2016.

[7] W. Chen, S. Toueg, and M. K. Aguilera. On the quality of service of failure detectors. *IEEE Transactions on computers*, 51(5):561–580, 2002.

[8] R. I. Ciobanu, C. Dobre, and V. Cristea. Social aspects to support opportunistic networks in an academic environment. In *International Conference on Ad-Hoc Networks and Wireless*, pages 69–82. Springer, 2012.

[9] R. I. Ciobanu, C. Dobre, and V. Cristea. Sprint: Social prediction-based opportunistic routing. In *2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–7, June 2013.

[10] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012.

[11] F. DaCosta. *Rethinking the Internet of Things: a scalable approach to connecting everything*. ApressOpen, New York, NY, 2013.

[12] O. Dagdeviren and K. Erciyes. A hierarchical leader election protocol for mobile ad hoc networks. In *International Conference on Computational Science*, pages 509–518. Springer, 2008.

[13] E. Dahlgren and H. Mahmood. Evaluation of indoor positioning based on bluetooth smart technology. *Master of Science Thesis in the Programme Computer Systems and Networks*, 2014.

[14] L. David, R. Vasconcelos, L. Alves, R. André, and M. Endler. A dds-based middleware for scalable tracking, communication and collaboration of mobile nodes. *Journal of Internet Services and Applications*, 4(1):16, Jul 2013.

[15] A. De Masi. *Load Balancing in P2P smartphone based distributed IoT systems*. PhD thesis, University of Lorraine, 2015.

[16] J. Decuir. Introducing bluetooth smart: Part 1: A look at both classic and new technologies. *IEEE Consumer Electronics Magazine*, 3(1):12–18, 2014.

[17] E. W. Dijkstra and C. S. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4, 1980.

[18] R. Drãgan, R. I. Ciobanu, and C. Dobre. Leader election in opportunistic networks. In *2017 16th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 157–164, July 2017.

[19] M. Endler, G. Baptista, L. D. Silva, R. Vasconcelos, M. Malcher, V. Pantoja, V. Pinheiro, and J. Viterbo. Contextnet: Context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. In *Proceedings of the Workshop on Posters and Demos Track*, PDT '11, pages 2:1–2:2, New York, NY, USA, 2011. ACM.

[20] L. Farhan, S. T. Shukur, A. E. Alissa, M. Alrweg, U. Raza, and R. Kharel. A survey on the challenges and opportunities of the internet of things (iot). In *2017 Eleventh International Conference on Sensing Technology (ICST)*, pages 1–5, Dec 2017.

[21] C. Fernández-Campusano, M. Larrea, R. Cortiñas, and M. Raynal. Eventual leader election despite crash-recovery and omission failures. In *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 209–214, Nov 2015.

[22] G. N. Frederickson and N. A. Lynch. Electing a leader in a synchronous ring. *Journal of the ACM (JACM)*, 34(1):98–115, 1987.

[23] M. Galeev. Bluetooth 4.0: an introduction to bluetooth low energy-part i. *EE Times, Design*, 2011.

[24] F. S. Gharehchopogh and H. Arjang. A survey and taxonomy of leader election algorithms in distributed systems. *Indian Journal of Science and Technology*, 7(6), 2014.

[25] B. Gomes, L. Muniz, F. J. d. S. e Silva, L. E. T. Ríos, and M. Endler. A comprehensive cloud-based iot software infrastructure for ambient assisted living. In *Cloud Technologies and Applications (CloudTech), 2015 International Conference on*, pages 1–8. IEEE, 2015.

[26] B. d. T. P. Gomes, L. C. M. Muniz, F. J. da Silva e Silva, D. V. dos Santos, R. F. Lopes, L. R. Coutinho, F. O. Carvalho, and M. Endler. A middleware with comprehensive quality of context support for the internet of things applications. *Sensors*, 17(12), 2017.

[27] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[28] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[29] C. Harrison and I. Donnelly. A theory of smart cities. *Proceedings of the 55th Annual Meeting of the International Society for the Systems Sciences*, pages 1–15, July 2011.

[30] R. T. Hermeto, D. S. Kridi, A. R. Rocha, and D. G. Gomes. Um algoritmo distribuido para eleicao de lideres de clusters semanticos em redes de sensores sem fio. In *Simposio Brasileiro de Computao Ubiqua e Pervasiva (SBCUP 2013), Anais do XXXIII Congresso da Sociedade Brasileira de Computao*, pages 2022–2031, 2013.

[31] Y. Jiang. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):585–599, 2016.

[32] Y. Jiang, Y. Zhou, and W. Wang. Task allocation for undependable multiagent systems in social networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1671–1681, 2013.

[33] P. Kumar, S. Ranganath, H. Weimin, and K. Sengupta. Framework for real-time behavior interpretation from traffic video. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):43–53, 2005.

[34] S. B. Letaifa. How to strategize smart cities: Revealing the smart model. *Journal of Business Research*, 68(7):1414–1419, 2015.

[35] A. C. R. Lima and G. L. Gonçalves. Transmissão de áudio sem fio por tecnologia bluetooth. 2010.

[36] A. Lundberg. Utilização do processamento de eventos complexos (cep) para melhoria do desempenho operacional. *BUSINESS INTELLIGENCE JOURNAL*, 11(1):16.

[37] T. Ma, J. Hillston, and S. Anderson. On the quality of service of crash-recovery failure detectors. *IEEE Transactions on Dependable and Secure Computing*, 7(3):271–283, 2010.

[38] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516, 2012.

[39] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. Vu. Internet of mobile things: Mobility-driven challenges, designs and implementations. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 25–36, April 2016.

[40] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. Vu. Internet of mobile things: Mobility-driven challenges, designs and implementations. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 25–36. IEEE, 2016.

[41] U. Nations. World urbanization prospects: The 2014 revision, highlights. department of economic and social affairs. *Population Division, United Nations*, 2014.

[42] J. D. Pereira, F. J. da Silva e Silva, L. R. Coutinho, B. de Tácio Pereira Gomes, and M. Endler. A movement activity recognition pervasive system for patient monitoring in ambient assisted living. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 155–161. ACM, 2016.

[43] M. Rahman, M. Abdullah-Al-Wadud, and O. Chae. Performance analysis of leader election algorithms in mobile ad hoc networks. *Int'l J. of Computer Science and Network Security*, 8(2):257–263, 2008.

[44] V. Raychoudhury, J. Cao, and W. Wu. Top k-leader election in wireless ad hoc networks. In *Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on*, pages 1–6. IEEE, 2008.

[45] J. D. P. Ribeiro Filho, F. J. d. S. e Silva, L. R. Coutinho, and B. d. T. P. Gomes. Mhars: A mobile system for human activity recognition and inference of health situations in ambient assisted living. *Journal of Applied Computing Research*, 5(1):44–58, 2016.

[46] A. R. Rocha, L. Pirmez, F. C. Delicato, É. Lemos, I. Santos, D. G. Gomes, and J. N. de Souza. Wsns clustering based on semantic neighborhood relationships. *Computer Networks*, 56(5):1627–1645, 2012.

[47] J. Rodas, C. J. Escudero, and D. I. Iglesia. Bayesian filtering for a bluetooth positioning system. In *Wireless Communication Systems. 2008. ISWCS'08. IEEE International Symposium on*, pages 618–622. IEEE, 2008.

[48] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e Silva. The mobile hub concept: Enabling applications for the internet of mobile things. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 123–128. IEEE, 2015.

[49] R. O. Vasconcelos, M. Endler, B. Gomes, and F. Silva. Autonomous load balancing of data stream processing and mobile communications in scalable data distribution systems. *Int. J. Adv. Intell. Syst*, 6(3&4):300–317, 2013.

[50] R. O. Vasconcelos, L. Talavera, I. Vasconcelos, M. Roriz, M. Endler, B. d. T. P. Gomes, and F. J. d. S. e Silva. An adaptive middleware for opportunistic mobile sensing. In *Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on*, pages 1–10. IEEE, 2015.

[51] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, pages 350–360, Oct 2004.

[52] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert. Bluetooth positioning using rssi and triangulation methods. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 837–842. IEEE, 2013.

[53] M. Yun and B. Yuxin. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *Advances in Energy Engineering (ICAEE), 2010 International Conference on*, pages 69–72. IEEE, 2010.

[54] G. Zhang, X. Kuang, J. Chen, and Y. Zhang. Design and implementation of a leader election algorithm in hierarchy mobile ad hoc network. In *Computer Science & Education, 2009. ICCSE'09. 4th International Conference on*, pages 263–268. IEEE, 2009.