Disaster-FD: Um detector de falhas para ambientes suscetíveis a desastres

Abadio de Paulo Silva

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2024

Abadio de Paulo Silva

Disaster-FD: Um detector de falhas para ambientes suscetíveis a desastres

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Paulo Rodolfo da Silva Leite Coelho
Coorientador: Prof. Dr. Rafael Pasquini

Uberlândia
2024

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S586 Silva, Abadio de Paulo, 1977-
2024 Disaster-FD - Um detector de falhas para ambientes suscetíveis a desastres [recurso eletrônico] / Abadio de Paulo Silva. - 2024.

Orientador: Paulo Rodolfo da Silva Leite Coelho.
Coorientador: Rafael Pasquini.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2024.620
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Coelho, Paulo Rodolfo da Silva Leite,1982-, (Orient.). II. Pasquini, Rafael,1981-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação de Mestrado, 21/2024, PPGCO				
Data:	20 de agosto de 2024	Hora de início:	14:03	Hora de encerramento:	16:48
Matrícula do Discente:	12212CCP001				
Nome do Discente:	Abadio de Paulo Silva				
Título do Trabalho:	Disaster-FD: Um detector de falhas para ambientes suscetíveis a desastres				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	----------				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Rafael Pasquini - FACOM/UFU (Coorientador), Rodrigo Sanches Miani - FACOM/UFU, Fernando Luís Dotti - PUCRS e Paulo Rodolfo da Silva Leite Coelho - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Fernando Luís Dotti - Porto Alegre/RS . Os outros membros da banca e o aluno participaram da cidade de Uberlândia.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Paulo Rodolfo da Silva Leite Coelho, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir ao candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos,

conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Rafael Pasquini**, **Professor(a) do Magistério Superior**, em 21/08/2024, às 11:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Paulo Rodolfo da Silva Leite Coelho**, **Professor(a) do Magistério Superior**, em 21/08/2024, às 15:59, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rodrigo Sanches Miani**, **Professor(a) do Magistério Superior**, em 26/08/2024, às 11:11, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Fernando Luís Dotti**, **Usuário Externo**, em 29/08/2024, às 16:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5534785** e o código CRC **320764E1**.

Referência: Processo nº 23117.045086/2024-73 SEI nº 5534785

Agradecimentos

É com profunda gratidão que inicio estes agradecimentos, primeiramente a Deus, por me dar força e orientação em cada passo desta jornada. Extendo minha mais sincera apreciação ao Prof. Dr. Paulo Coelho e ao Prof. Dr. Rafael Pasquini, cuja orientação, paciência e conhecimento não só moldaram este trabalho, mas também enriqueceram meu crescimento pessoal e profissional. Eles foram mais do que orientadores; foram verdadeiros mentores.

Dedico um agradecimento especial à minha família: mãe, esposa e filhos. Vocês foram a rocha em tempos de incerteza, a alegria em momentos de celebração e o consolo nos desafios. Sua presença, apoio e amor incondicionais foram a luz guiando meu caminho.

Não posso deixar de agradecer ao Departamento Municipal de Água e Esgoto - DMAE, cuja compreensão e flexibilidade no horário de trabalho foram fundamentais para conciliar minhas responsabilidades acadêmicas e profissionais. Este suporte foi um componente chave para tornar possível a conclusão deste mestrado.

A todos vocês, meu mais sincero agradecimento. Vocês são parte integrante desta conquista.

"Na jornada da vida, cada desafio superado com o apoio de amigos e familiares marca o início de um novo capítulo, repleto de aprendizado e crescimento. Este caminho é uma fotografia de momentos, onde o amor e o apoio que recebemos são os alicerces que nos fortalecem para enfrentar cada novo desafio com esperança e determinação."

(Maristela Aparecida Pontes Machado)

Resumo

Disaster-FD é um sistema de detecção de falhas para redes IoT em ambientes remotos sujeitos a desastres. Construído sobre o *Impact-FD*, o *Disaster-FD* concentra-se no monitoramento ativo e na avaliação da confiabilidade da rede, investigando elementos essenciais como limiar de confiabilidade, nível de confiança e fator de impacto. O sistema apresenta melhorias em relação ao seu predecessor, com foco especial no monitoramento em tempo real e na precisão do cálculo do tempo estimado para a próxima mensagem. A eficácia do *Disaster-FD* foi demonstrada através de experimentos na plataforma IoT-LAB. Os testes incluíram o monitoramento de duas regiões distintas, permitindo a simulação de cenários reais de desastres e a avaliação da confiabilidade da rede em diferentes localizações. A capacidade de monitoramento federado no *Disaster-FD* é um componente crucial para o detector de falhas realizar o monitoramento eficaz entre múltiplas regiões geográficas. Além disso, o sistema realiza o monitoramento do consumo de energia dos dispositivos IoT, visando à sustentabilidade e à eficiência energética em cenários prolongados de desastres. Esses avanços tecnológicos posicionam o *Disaster-FD* como uma solução promissora na gestão de falhas e resposta a desastres em redes IoT. Vale destacar que os resultados dos testes realizados com o *Disaster-FD* foram superiores em comparação ao seu predecessor *Impact-FD*.

Palavras-chave: IoT, Gerenciamento de desastres, Detecção de Falhas, Monitoramento Ativo, Nível de confiança da rede, Monitoramento em Tempo Real, Margem de Segurança, Monitoramento Federado, Impact-FD, IoT-LAB, Eficiência Energética, Cálculo de Tempo Estimado, Sustentabilidade Energética.

Abstract

Disaster-FD is a fault detection system for IoT networks in remote environments subject to disasters. Built on top of *Impact-FD*, *Disaster-FD* focuses on active monitoring and network reliability assessment, investigating essential elements such as reliability threshold, trust level, and impact factor. The system presents improvements over its predecessor, with a special focus on real-time monitoring and the accuracy of the estimated time calculation for the next message. The effectiveness of *Disaster-FD* was demonstrated through experiments on the IoT-LAB platform. The tests included monitoring two distinct regions, allowing for the simulation of real disaster scenarios and the assessment of network reliability in different locations. The federated monitoring capability in *Disaster-FD* is a crucial component for the fault detector to perform effective monitoring across multiple geographic regions. Additionally, the system monitors the energy consumption of IoT devices, aiming for sustainability and energy efficiency in prolonged disaster scenarios. These technological advancements position *Disaster-FD* as a promising solution in fault management and disaster response in IoT networks. It is worth noting that the test results obtained with *Disaster-FD* were superior compared to its predecessor, *Impact-FD*.

Keywords: IoT, Disaster Management, Fault Detection, Active Monitoring, Network Confidence Level, Real-Time Monitoring, Safety Margin, Federated Monitoring, Impact-FD, IoT-LAB, Energy Efficiency, Estimated Time Calculation, Energy Sustainability.

Lista de ilustrações

Lista de tabelas

Lista de siglas

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

TSCH Time-Slotted Channel Hopping

ADMITS projetoArchitecting Distributed Monitoring and Analytics in IoT-based Disaster Scenarios

HB Heartbeats

CNM Confederação Nacional de Municípios

CoAP Constrained Application Protocol

CLI Linha de Comando

CN Computação em Nuvem

DF Detectores de Falhas

DTLS Datagram Transport Layer Security

DNS Sistema de Nomes de Domínio - Domain Name System

DAARP Data-Aggregation Aware Routing Protocol

EGs Exterior Gateways

EA Estimated Arrival

Fog Computação em Névoa

FSN Redes de Sensores Federadas

CDF Cumulative Distribution Function

GST Global Stabilization Time

HTTP Hypertext Transfer Protocol

IoT Internet das Coisas

IETF Internet Engineering Task Force

IP Internet Protocol

IGs Interior Gateways

ICMP Internet Control Message Protocol

LDAP Directory Access Protocol

L-FSN Layered Federated Sensor Network

M2M Machine to Machine

MQTT Message Queuing Telemetry Transport

MID Message ID

OSGP Open Smart Grid Protocol

OAR Software de Agendamento

P.A. Probabilidade da Precisão da Consulta - Query Accuracy Probability

QoS Qualidade de Serviço - Quality of Service

RTT Round Trip Time

RSSF Rede de sensores sem fio

SWIM Scalable Weakly-consistent Infection-style Membership protocol

TD Tempo de Detecção - Detection Time

TMR Tempo de Retorno do Erro - Mistake Recurrence

TM Duração do Erro - Mistake Duration

TG Duração do Período Bom - Good Period Duration

TFG Duração do Período Bom à Frente - Forward Good Period Duration

TCP Transmission Control Protocol

TSCH Time-Slotted Channel Hopping

UDP User Datagram Protocol

UCI Machine Learning Repository

URLs Uniform Resource Locators

WSN Wireless Sensor Network

WAN Wide Area Network

VPN Rede Privada Virtual - Virtual Private Network

Sumário

Capítulo 1

Introdução

Os desastres naturais são atualmente um dos grandes problemas que afetam a sociedade, sendo alguns deles: furacões, tempestades, deslizamentos, epidemias, inundações e estiagens (SOUZA, 2011). Seus impactos podem causar, por exemplo, o interrompimento de serviços essenciais como o abastecimento de água e energia, gerar prejuízos econômicos e financeiros às propriedades públicas, privadas, indústria e comércio; além de ocasionar perda de vidas humanas, ferimentos, doenças e outros efeitos negativos ao bem-estar da população. Esses eventos de desastres assolam o mundo, em particular no Brasil a maioria dos desastres naturais têm suas causas relacionadas ao clima. Insta dizer que nos desastres naturais ocorridos no Brasil em 2022, cerca de 3,4 mil pessoas foram afetadas diretamente, segundo levantamento produzido pela CNM - Confederação Nacional de Municípios (JANONEDA, 2022).

Ainda conforme relatado pela CNM - Confederação Nacional de Municípios (JANONEDA, 2022), quase oito milhões de brasileiros foram afetados indiretamente por catástrofes ambientais nos primeiros três meses do ano de 2022. Entre os eventos adversos, as secas e estiagens foram as mais recorrentes, sendo responsáveis por 40% dos problemas ambientais no Brasil em 2022. Já as fortes chuvas, as enxurradas, as inundações e os alagamentos representam, juntos, 15,7% das ocorrências. Os vendavais e os deslizamentos também aparecem na lista com um percentual de 3,2% e 1,3%, respectivamente. O estado de Minas Gerais foi o mais afetado por catástrofes ambientais, com 8 mil eventos ao longo dos primeiros meses do ano de 2022. Logo em seguida aparecem Bahia, Paraíba e Santa Catarina, que totalizam 14 mil anormalidades. O Rio de Janeiro também aparece como destaque negativo, com quase 1,5 mil desastres. Políticas públicas e privadas que fomentem programas e estudos sustentáveis de monitoramento e gerenciamento de riscos para minimizar os danos causados por desastres naturais são essenciais. Uma tecnologia que tem sido utilizada para amenizar os impactos dos desastres naturais é o monitoramento remoto via sensores, advindos da Internet das Coisas - IoT, segundo (ATZORI; IERA; MORABITO, 2010). O paradigma IoT pode ser definido como a interconectividade de diversos tipos de objetos a uma rede, interagindo e cooperando entre si para que possam atingir objetivos em comum.

Vale destacar que em cenários de desastres, principalmente os de grande escala, a infraestrutura de comunicação além dos dispositivos IoT instalados em locais estratégicos para o monitoramento de desastres pode falhar, tanto temporariamente, quanto permanentemente, tornando indisponível o envio dos dados coletados. Logo, o desastre pode ser subdimensionado pela falta de dados. Do mesmo modo, alarmes que poderiam ser disparados notificando a criticidade do problema deixariam de ser enviados. Assim sendo, o objetivo desse estudo é desenvolver um algoritmo que possa monitorar uma rede IoT, a fim de que seja possível aferir a disponibilidade desses dispositivos e estabelecer o nível de confiança dos processos oriundos deste ecossistema, além de propor melhorias no gerenciamento dos sensores.

Chandra e Toueg definiram os detectores de falhas através de duas propriedades a serem respeitadas: *completeness* (abrangência, completeza) e *accuracy* (precisão) (CHANDRA; TOUEG, 1996). A propriedade *completeness* refere-se à capacidade do detector identificar todos os processos que estão fa-

lhos, enquanto *accuracy* determina a precisão desta suspeita, a fim de evitar a inclusão de processos corretos nas listas de suspeitos. Ao respeitar essas duas propriedades, um detector de falhas garante que os algoritmos que o utilizem não perderão a consistência das decisões, nem ficarão indefinidamente bloqueados.

O monitoramento com definição de limiar é uma técnica utilizada para detectar falhas em cenários críticos, como desastres naturais, estabelecendo limites específicos para diferentes tipos de eventos. Quando uma métrica específica, como a movimentação sísmica, atinge um valor predefinido, o ambiente é considerado não confiável, e ações corretivas são iniciadas. Essa abordagem baseia-se na premissa de que limites claros ajudam a distinguir entre condições normais e situações de risco.

De maneira semelhante, o *Impact-FD* apresentado no trabalho (ROSSETTO et al., 2018), utiliza a técnica de monitoramento por limiar para identificar falhas na comunicação entre dispositivos e estabelecer o nível de confiança da rede. No *Impact-FD*, os limiares são estabelecidos levando-se em consideração a importância dos dispositivos monitorados e um quantitativo mínimo de dispositivos que deve estar funcional para que o sistema funcione adequadamente. Quando essas métricas excedem os limites predefinidos, o sistema considera que houve uma falha na rede, assim como em cenários de desastres naturais, onde a definição de limiares pode sinalizar um evento iminente de desastre natural.

O monitoramento federado para detectores de falha utiliza uma rede de múltiplos sensores distribuídos em várias regiões geográficas para coletar e compartilhar dados em tempo real. Essa abordagem permite que diferentes entidades, como agências governamentais, serviços de emergência e organizações de proteção ambiental, colaborem para monitorar grandes áreas e detectar possíveis falhas, como terremotos, inundações ou incêndios florestais.

No caso de incêndios florestais, o trabalho apresentado por (GRANEMANN; CARNEIRO, 2009), o monitoramento federado torna-se relevante pois integra dados de sensores localizados em diferentes regiões. Por exemplo, um aumento gradual na temperatura detectado por sensores em diversas partes de uma floresta pode indicar o início de um incêndio. Se o incêndio se espalha rapidamente, ele pode afetar não apenas a floresta inicial, mas também regiões florestais vizinhas e até mesmo áreas urbanas próximas.

Ao conectar dados de múltiplas regiões, o monitoramento federado permite que diferentes entidades trabalhem juntas para entender a propagação potencial de um incêndio florestal e coordenar uma resposta rápida e eficaz. Isso inclui a antecipação de ações preventivas para proteger tanto as áreas florestais quanto as urbanas ameaçadas, minimizando o impacto ambiental, os danos à infraestrutura e os riscos para a população.

Essa capacidade de monitorar e compartilhar informações entre regiões conectadas garante que as respostas a desastres sejam mais coordenadas e eficazes, proporcionando uma visão abrangente do cenário e facilitando a mobilização rápida de recursos para conter o desastre antes que ele se espalhe para áreas adicionais.

1.1 Motivação

Na contemporaneidade, a integração da tecnologia nos esforços para mitigar e responder a desastres naturais se tornou imperativa. A vulnerabilidade de comunidades em todo o mundo aos impactos devastadores de eventos extremos como inundações, terremotos, e tempestades – exige uma abordagem inovadora e eficaz para a prevenção e resposta a desastres.

Neste contexto, o desenvolvimento do sistema *Disaster-FD*, uma infraestrutura de detecção de falhas projetada especificamente para ambientes suscetíveis a desastres naturais, constituirá uma contribuição para o campo da Ciência da Computação e a gestão de desastres.

Este sistema, fundamentado nos avanços da Internet das Coisas (IoT) e nas redes de sensores sem fio (RSSF), visa melhorar a confiabilidade e eficácia do monitoramento remoto e da resposta a desastres, empregando algoritmos de detecção de falhas adaptados às peculiaridades de cenários de desastres naturais.

A relevância deste trabalho reside na sua capacidade de oferecer uma resposta ágil e precisa em momentos críticos, potencializando o uso da tecnologia IoT para salvar vidas, proteger infraestruturas, e minimizar os impactos econômicos e sociais dos desastres naturais. Por meio da implementação e validação do sistema *Disaster-FD*, esta pesquisa não apenas avança no entendimento técnico e aplicado dos detectores de falhas em contextos de desastres, mas também destaca a importância da interdisciplinaridade e da inovação tecnológica na construção de sociedades mais resilientes e preparadas para enfrentar os desafios impostos por fenômenos naturais extremos.

1.2 Objetivos e Desafios da Pesquisa

Este estudo é parte de uma colaboração internacional entre instituições acadêmicas da França, Uruguai e Chile, focada em explorar e aplicar tecnologias de monitoramento baseadas na Internet das Coisas (IoT), bem como protocolos de comunicação entre dispositivos IoT, como sensores. O objetivo é desenvolver um sistema que suporte um ambiente de computação distribuída. Esse ambiente é projetado para fornecer monitoramento em tempo real e federado, detecção eficiente de falhas em dispositivos IoT operando em cenários suscetíveis a desastres naturais.

O projeto denominado ***Disaster-FD***, busca inspiração no trabalho anterior *Impact-FD* de (ROSSETTO et al., 2018). De maneira similar, o projeto*Architecting Distributed Monitoring and Analytics in IoT-based Disaster Scenarios* - (ADMITS), desenvolvido por (PASQUINI et al., 2020), também se inspira nesse marco. Ambos os projetos buscam não apenas expandir a capacidade de detecção de falhas em tais cenários, mas também fomentar o desenvolvimento de monitoramento distribuído e análise de dados para fortalecer a resiliência e eficácia na resposta a eventos adversos.

Um foco chave do ADMITS envolve a criação de algoritmos, protocolos e arquiteturas que melhorem significativamente o ambiente de computação distribuída, promovendo um monitoramento mais ágil, uma detecção de falhas mais precisa e uma análise de dados mais abrangente em situações de desastre. Ademais o ADMITS foca em arquitetar soluções para monitoramento distribuído e análises em cenários de desastres naturais, com ênfase especial na melhoria da infraestrutura de monitoramento através da integração com a camada de névoa. A camada de névoa, atuando como uma extensão da computação em nuvem, possibilita o processamento e a análise de dados mais próximos da fonte, reduzindo a latência e aumentando a eficácia na detecção e resposta a falhas em tempo real.

Complementando esses esforços, o "*Disaster-FD*" tem o objetivo específico de desenvolver um sistema capaz de monitorar redes IoT. Este sistema não somente avalia a disponibilidade dos dispositivos e estabelece um nível de confiança global para os processos dentro do ecossistema, mas também prioriza o monitoramento do consumo de energia dos dispositivos. Essa funcionalidade é essencial para garantir a sustentabilidade e eficiência energética dos sistemas IoT, especialmente em cenários prolongados de desastres naturais onde a manutenção da operação dos dispositivos é crítica. Além disso, incorporando o monitoramento federado entre regiões em tempo real, este sistema adota uma abordagem integrada e coordenada, alinhando-se à visão do ADMITS e expandindo sua aplicabilidade para além da detecção precoce de falhas, ao incluir também a gestão do monitoramento do consumo de energia.

Reconhecendo a necessidade crítica de infraestruturas de monitoramento e análise robustas diante de desastres naturais, o estudo se empenha em utilizar a IoT e tecnologias correlatas para melhorar a detecção precoce de falhas e facilitar decisões baseadas em dados.

Assim, com o desenvolvimento do "*Disaster-FD*", aspira-se oferecer uma contribuição valiosa na evolução das abordagens de gestão de desastres. Ao aprimorar a capacidade de monitoramento e análise de falhas em redes IoT, tem-se como objetivo não apenas melhorar a eficácia das estratégias de mitigação de desastres, mas também promover um aumento significativo na segurança e qualidade de vida da população em face desses imprevistos. Dessa maneira, é esperado que este trabalho seja um passo adiante na direção de um futuro mais seguro e resiliente, onde a tecnologia e a inovação desempenham papéis cruciais na proteção contra as adversidades naturais.

1.3 Hipótese

O avanço da Internet das Coisas gerou um novo nível estratégico para identificar os possíveis riscos e desastres antes que aconteçam, possibilitando uma ênfase maior na prevenção de perdas materiais e vidas humanas. A IoT possibilita o uso de dispositivos conectados à internet como sensores e atuadores, que podem ser instalados em diferentes locais para coletar e transmitir dados em tempo real. Esses dispositivos podem funcionar mesmo sob condições adversas, como conexão sem fio fraca e pouca fonte de energia, sendo uma boa opção de instalação em áreas de difícil acesso para monitoramento de possíveis deslizamentos, alagamentos, entre outros desastres.

Sendo assim, elencamos as seguintes hipóteses:

- A primeira hipótese é que os detectores de falhas empregados em redes tradicionais, que monitoram cada sensor de forma individualizada em ambientes suscetíveis a desastres, podem ser adaptados para redes de sensores IoT. Nessas redes IoT, o foco estaria no monitoramento a nível de sistema ou do processo como um todo, permitindo uma detecção de falhas mais integrada e abrangente em cenários de desastres.

- A segunda hipótese seria a utilização de comunicação federada entre detectores de falhas em diferentes regiões pode melhorar a eficiência na detecção de possíveis cenários de desastres naturais.

1.4 Contribuições

O monitoramento da rede IoT é importante para garantir que os sensores estejam funcionando corretamente e coletando dados precisos. Em caso de desastre natural, é ainda mais importante que a rede de sensores esteja funcionando corretamente, pois os dados coletados pelos sensores podem ser utilizados para a tomada de decisões importantes, como a evacuação de áreas afetadas ou o envio de ajuda. A contribuição principal deste trabalho é uma infraestrutura baseada em um algoritmo de detector de falhas para monitoramento que, diferente dos Detectores de Falhas - (DF) tradicionais, considera limiar a partir do qual o ambiente deixa de ser considerado confiável.

Outras contribuições:

- Federação de monitoramento: o algoritmo comunicando-se com múltiplos sensores em diversos ambientes de execução (como exemplo o FIT IoT-LAB) permite que diferentes entidades (nós monitores) trabalhem em conjunto para monitorar uma área maior e detectar possíveis falhas. Consequentemente o monitoramento federado amplia a análise de detecção e auxilia na rápida tomada decisão em situações de emergência.

- Monitoramento do consumo de energia dos dispositivos: o algoritmo realiza o monitoramento do consumo de energia dos dispositivos. Essa funcionalidade, no cenário da detecção de falhas em redes IoT, especialmente em contextos de desastres naturais, torna-se particularmente relevante.

Considera-se a crescente demanda por sistemas IoT eficientes em termos energéticos, capazes de operar de maneira autônoma por períodos prolongados, especialmente em ambientes adversos ou inacessíveis.

1.5 Organização da Dissertação

Fundamentação Teórica (Capítulo 2) explora os conceitos fundamentais sobre a Internet das Coisas (IoT), Modelos de Sistema, e Detectores de Falhas. Aborda a estrutura e integração de redes de sensores sem fio (RSSF) com IoT, modelos de sincronia (síncrono, assíncrono, e parcialmente síncrono), e os princípios dos canais de comunicação. Discute o consenso em sistemas distribuídos e a impossibilidade de FLP, apresentando a importância dos detectores de falhas não confiáveis.

Trabalhos Relacionados (Capítulo 3) revisa as pesquisas anteriores e as soluções propostas no domínio de detectores de falhas para IoT e monitoramento de desastres. Compara diferentes abordagens e destaca as lacunas no conhecimento que o presente trabalho visa preencher.

Proposta (Capítulo 4) detalha o desenvolvimento do *Disaster-FD*, incluindo definições, formalizações, a fórmula para estimativa da chegada de *heartbeats*, e as principais diferenças em relação ao *Impact-FD*. Aborda a implementação com ênfase no gerenciamento de *threads* e na diferenciação entre monitoramento por ICMP e dispositivos por CoAP.

Experimentos e Análise dos Resultados (Capítulo 5) descreve a metodologia adotada para a avaliação da proposta, incluindo a configuração dos experimentos, métricas de desempenho utilizadas, e análise detalhada dos resultados obtidos. Apresenta estudos de caso e a aplicação prática na Plataforma IoT-LAB.

Conclusão (Capítulo 6) analisa as principais contribuições e resultados alcançados, discute as implicações práticas e teóricas do estudo, e sugere direções futuras para a pesquisa na área de detectores de falhas e IoT para monitoramento de desastres.

Capítulo 2

Fundamentação Teórica

Este capítulo estabelece a base conceitual necessária para compreender os desafios e soluções envolvendo a detecção de falhas em cenários de desastres, especialmente no contexto da Internet das Coisas (IoT). Aborda-se inicialmente a arquitetura da IoT e sua importância na coleta de dados em ambientes propícios a desastres. Discutem-se também os modelos de sistema, destacando as diferenças entre sistemas síncronos, assíncronos e parcialmente síncronos, e sua relevância na implementação de sistemas de detecção de falhas eficientes, a comunicação e o consenso em sistemas distribuídos são examinados. Por fim, o capítulo explora os detectores de falhas, focando na importância de detectores não confiáveis e nas métricas de QoS para avaliar e garantir a eficácia do sistema *Disaster-FD*.

2.1 Internet das Coisas - (IoT)

O termo Internet das Coisas - (IoT) foi criado em 1999 por Kevin Ashton, durante uma apresentação de negócios feita à Procter&Gamble (CARRION; QUARESMA, 2019). O objetivo de Ashton era de incorporar microchips aos produtos, recebendo, assim, dados que indicariam se os itens haviam sido vendidos ou se estavam em falta nas prateleiras. Foi dessa forma que ele acabou idealizando um sistema de sensores que poderiam conectar o mundo físico à Internet. O paradigma de IoT pode ser definido como a interconectividade de diversos tipos de objetos a uma rede, interagindo e cooperando entre si para que possam atingir objetivos em comum (ATZORI; IERA; MORABITO, 2010).

Atualmente a Internet das coisas pode ser considerada um dos principais pilares da Indústria 4.0. A tecnologia IoT vive em constante evolução devido ao número de dispositivos conectados em larga escala à internet, assim como sua implementação, o que contribui para a interoperabilidade entre sistemas e aplicações distribuídas. A tecnologia IoT possui papel fundamental na estruturação de ambientes inteligentes com a sua aplicação aliada a outras tecnologias promovidas pela mobilização da quarta revolução industrial, também conhecida como a era da conectividade.

2.1.1 Estrutura de Camadas da Arquitetura IoT

A estrutura da Internet das Coisas (IoT), conforme postulado por Yang et al. (2011), é meticulosamente delineada em uma arquitetura tripartida, compreendendo três camadas essenciais, cada qual com atribuições e funções distintas. Este modelo estratificado é instrumental na sistematização e distinção de responsabilidades dentro do desenvolvimento de sistemas IoT, promovendo uma abordagem modular e coesa.

Portanto a arquitetura básica que serve como ponto de partida para as demais, pode ser resumida em três camadas, conforme apresentado na Figura 1, que ilustra a estrutura fundamental da Internet

das Coisas (IoT), destacando suas três camadas colaborativas: percepção, rede e aplicação. Juntas, essas camadas formam a base que permite que a IoT funcione, coletando e utilizando dados para uma variedade de soluções inteligentes.

Figura 1 – Arquitetura Clássica IoT de 3 Camadas.

A Camada de Percepção, corresponde à aquisição direta de dados do ambiente; a **Camada de Rede** facilita a transferência de dados; e a **Camada de Aplicação** converte os dados processados em soluções e interações significativas para os usuários finais. A articulação dessas camadas fundamenta o ecossistema da IoT, permitindo a integração e o funcionamento harmonioso dos seus componentes.

2.1.2 Redes de Sensores Sem Fio (RSSF)

As Redes de Sensores Sem Fio (RSSF) representam um componente crucial dentro do espectro mais amplo da Internet das Coisas (IoT), especialmente no que diz respeito ao monitoramento e à resposta a desastres naturais. Estas redes são compostas por sensores distribuídos geograficamente, capazes de coletar e transmitir dados sobre o ambiente circundante. A integração desses sensores com a IoT amplia seu alcance e eficiência, permitindo uma comunicação contínua e em tempo real com sistemas de processamento e análise de dados.

Ademais, as Redes de Sensores Sem Fio (RSSF) são estruturadas a partir de algumas unidades até milhares de sensores, que têm a habilidade de captar e espalhar informações utilizando comunicação sem fio. Estas redes surgem da integração entre tecnologias de detecção, sistemas computacionais embarcados e processamento de dados distribuídos. Tipicamente, os dispositivos sensoriais em uma RSSF caracterizam-se por terem capacidades computacionais limitadas, serem alimentados por baterias, o que implica uma energia disponível restrita, e serem frequentemente posicionados em locais que apresentam riscos ou difícil acesso (VIEIRA, 2017).

2.1.3 Integração entre RSSF e IoT

A integração da Internet das Coisas (IoT) com Redes de Sensores Sem Fio (RSSF) marca um avanço significativo na coleta, processamento e aplicação de dados do mundo real, visando a tomada de decisões informadas, automação de processos e aprimoramento da qualidade de vida. Esta convergência oferece uma base robusta para a implementação de soluções inteligentes em diversos domínios, incluindo saúde, agricultura, indústria, cidades inteligentes e monitoramento contra desastres naturais. A IoT é um paradigma que se refere à interconexão digital de objetos cotidianos com a internet, permitindo que

eles enviem e recebam dados. A integração das RSSF na IoT amplia o alcance e a capacidade da IoT, permitindo a coleta de dados em tempo real de ambientes físicos vastos e variados.

Figura 2, ilustra como as Redes de Sensores Sem Fio (RSSF) se integram ao ecossistema da Internet das Coisas (IoT), estabelecendo um vínculo fundamental entre a coleta de dados do ambiente físico e sua utilização em aplicações digitais.

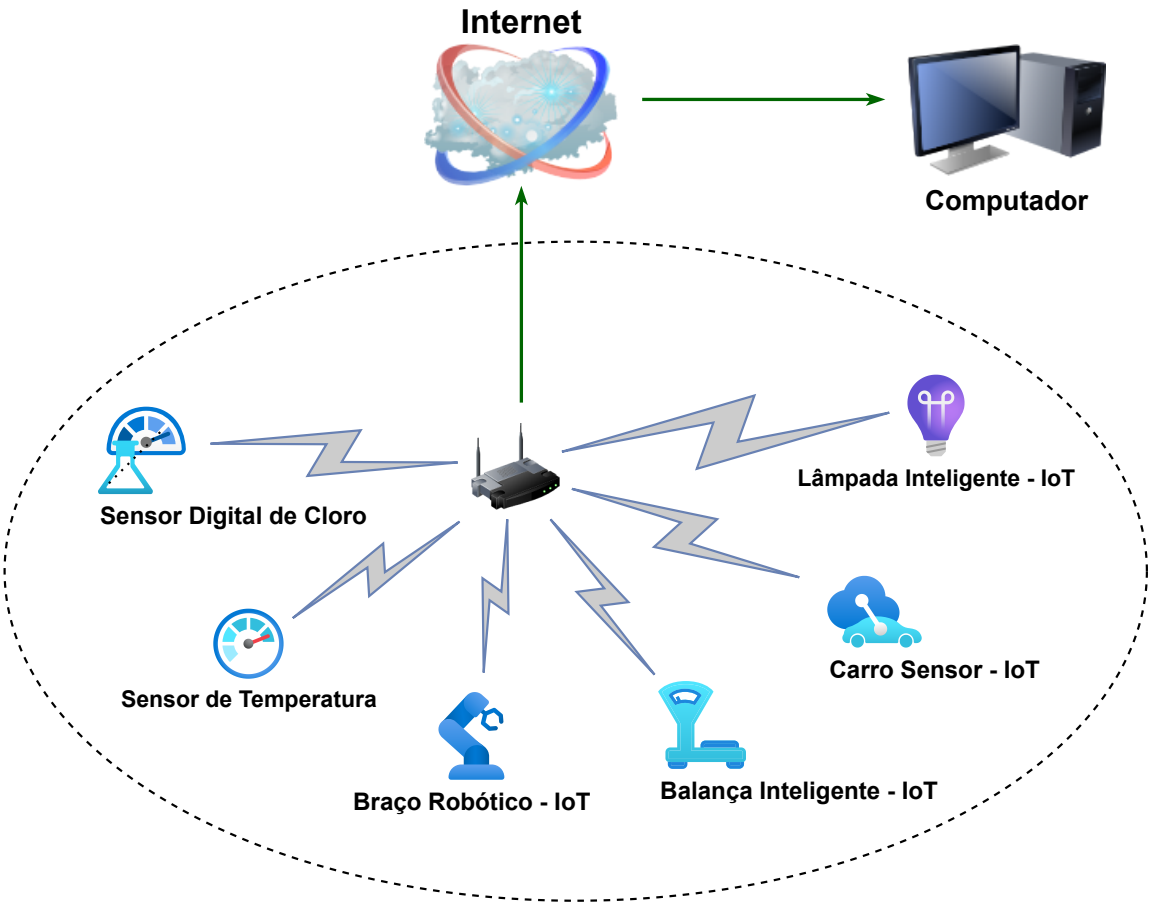

Figura 2 – Integração entre RSSF *e* IoT.

2.2 Modelo de Sistema

Este trabalho considera um sistema distribuído definido por um conjunto finito de processos, denotado como $\Pi = \{p_1, p_2, \ldots, p_n\}$. Esta formação reflete a organização do sistema, onde cada processo, individualmente identificado dentro do conjunto, tem a capacidade de operar de forma interdependente dos outros, permitindo assim a realização de tarefas distribuídas através de comunicação mútua.

Neste cenário, a comunicação e a sincronização entre os processos emergem como aspectos cruciais, dado que a falta de um tempo global sincronizado aumenta a complexidade da coordenação entre os processos. Diante desta complexidade propõe-se vários modelos de sincronismo e comunicação, cada um com suas peculiaridades e aplicações específicas. A compreensão desses modelos é essencial para o desenvolvimento de sistemas distribuídos que sejam capazes de manejar eficientemente as incertezas e as variações nos tempos de transmissão e processamento.

2.2.1 Modelo Síncrono

Nos modelos **síncronos**, existe a premissa de que os atrasos na rede e as velocidades dos processadores se mantêm dentro de limites conhecidos e previsíveis, conforme destacado por (VERISSIMO; RODRIGUES, 2012). Esta suposição permite um design mais simples e eficiente de protocolos, com a possibilidade de determinar *timeouts* específicos para a detecção de falhas. A clareza desses limites possibilita a diferenciação entre processos que estão apenas operando com lentidão e aqueles que efetivamente travaram, assim como entre mensagens que estão atrasadas e as que foram perdidas, facilitando a detecção de falhas de maneira precisa. Contudo, a exigência de estabelecer limites claros para garantir o funcionamento adequado sob as piores condições apresenta desafios, pois a violação dessas premissas pode comprometer a validade dos algoritmos utilizados, em concordância com (VERISSIMO; RODRIGUES, 2012).

2.2.2 Modelo Assíncrono

Em sistemas distribuídos, os modelos assíncronos se caracterizam pela ausência de suposições sobre limites exatos de tempo de processamento e atrasos na entrega de mensagens. Essa abordagem reflete a natureza imprevisível de muitos ambientes de rede, onde variabilidades significativas nas latências de comunicação e nos tempos de processamento podem ocorrer (CRISTIAN; FETZER, 1999; VERISSIMO; RODRIGUES, 2012). Diferentemente dos modelos síncronos, que operam com base em suposições temporais estritas, os modelos assíncronos permitem um maior grau de flexibilidade, adaptando-se de forma mais eficaz a condições de rede dinâmicas e imprevisíveis.

No entanto, essa flexibilidade vem com seus próprios desafios, especialmente no que diz respeito à detecção de falhas. Em um contexto assíncrono, a ausência de garantias temporais torna extremamente difícil discriminar entre um processo que efetivamente falhou e um que está meramente respondendo de forma lenta devido a atrasos na rede ou sobrecargas no sistema. Esta incerteza complica significativamente a concepção de mecanismos de detecção de falhas eficazes, uma vez que qualquer tentativa de inferir falhas com base em *timeouts* pode levar a falsos positivos, onde processos funcionais são incorretamente considerados como falhos.

2.2.3 Modelo Parcialmente Síncrono

Para superar as limitações dos modelos puramente assíncronos sem impor as restrições rigorosas dos modelos síncronos, foram propostos os modelos de **sincronia parcial**. Nestes sistemas, assume-se que existem limites para os tempos de processamento e para a entrega de mensagens, mas tais limites não são conhecidos a priori. O sistema pode experimentar períodos de comportamento assíncrono, contanto que eventualmente se torne síncrono por tempo suficiente para permitir a realização de cálculos distribuídos.

A sincronia parcial, uma abordagem introduzida por (DOLEV; DWORK; STOCKMEYER, 1987), foi projetada para se adaptar à dinâmica da comunicação e do processamento, sem exigir limites temporais fixos, os autores propuseram uma elaboração mais detalhada, apresentando dois cenários fundamentais dentro deste modelo:

Limites Temporais Desconhecidos: O primeiro cenário admite que, embora os limites para o processamento e a comunicação existam, eles não são conhecidos de antemão. Isso requer que os sistemas sejam projetados com uma margem de flexibilidade para acomodar variações inesperadas nos tempos de resposta.

Conhecimento Pós-GST: No segundo cenário, os limites são conhecidos, mas essa informação só se torna aplicável após um período inicial indefinido, denominado *Global Stabilization Time* - (GST). Antes

do GST, o sistema comporta-se de maneira assíncrona; após o GST, transita para um comportamento síncrono.

Posteriormente, (CHANDRA; TOUEG, 1996) introduziram um terceiro modelo, que relaxa ainda mais essas condições, assumindo limites desconhecidos que se tornam relevantes apenas após o GST.

Essencialmente, para sistemas eventualmente síncronos, a permanência em sincronia não é constante; é suficiente que o sistema atinja estabilidade temporária para que os algoritmos distribuídos concluam suas tarefas. Este princípio permite flexibilidade no design de sistemas distribuídos, adequando-se à natureza variável de ambientes reais.

2.3 Canais/Links de Comunicação

A comunicação em sistemas distribuídos é mediada pela troca de mensagens entre os diversos processos do sistema. Essa interação, fundamental para garantir a integridade e a eficácia do sistema, é facilitada pelos canais de comunicação, também conhecidos como links. Os canais de comunicação desempenham um papel crucial na transmissão de informações entre os componentes do sistema. Na literatura, os canais que não admitem perdas são denominados canais confiáveis, enquanto aqueles que podem sofrer perdas são conhecidos como *Fair-lossy* links (BASU; CHARRON-BOST; TOUEG, 1996).

Os links *Fair-lossy* são caracterizados por três propriedades principais:

1. **Perda justa (*fair-lossy*)**: Se uma mensagem é enviada infinitas vezes por um processo p_i para um processo p_j, e nenhum dos processos sofreu falhas, então a mensagem é entregue infinitas vezes ao processo p_j. Essa propriedade garante que a perda de mensagens seja distribuída de maneira justa entre os processos. A especificação dos *fair-lossy* links leva em consideração aspectos mais realistas, pois ela admite que falhas temporárias podem causar a perda de mensagens, por exemplo, devido à capacidade finita de armazenamento.

2. **Duplicação finita (*finite duplication*)**: Se uma mensagem é enviada finitas vezes de um processo p_i para um processo p_j, então essa mensagem não pode ser entregue infinitas vezes ao processo p_j. Isso evita a duplicação excessiva de mensagens.

3. **Nenhuma criação (*no creation*)**: Se uma mensagem é entregue a um processo p_j, então essa mensagem foi previamente enviada para p_j por algum processo p_i. Essa propriedade garante a consistência na comunicação entre os processos.

Além disso, é importante considerar os diferentes tipos de links de comunicação:

1. **Links Unidirecionais** a comunicação ocorre em uma única direção, de um remetente para um receptor, embora simples requerem mecanismos adicionais para a confirmação de recebimento se a confiabilidade for necessária.

2. **Links Bidirecionais** permitem a comunicação em ambas as direções, facilitando o diálogo entre processos, são fundamentais para implementações de detectores de falha que dependem de acks (confirmações) e *heartbeats* (sinais de vida).

3. **Link Confiável** se distingue por sua capacidade de assegurar a entrega exata das mensagens como foram enviadas, sem perdas, alterações ou duplicações ao longo do caminho. Graças a essas características, esses canais são considerados plenamente seguros e confiáveis, sendo a escolha ideal para situações que exigem a mais alta fidelidade na troca de informações.

2.4 Consenso

O consenso em ambientes distribuídos é fundamental para assegurar a confiabilidade e a sincronização entre os diferentes componentes de um sistema. Este desafio envolve alcançar um acordo comum entre múltiplos processos sobre um determinado valor ou estado, apesar das dificuldades impostas pelo próprio ambiente distribuído. Tais dificuldades incluem problemas de comunicação, variações de latência, e o risco de falhas ou comportamentos mal-intencionados dos componentes. O principal objetivo é garantir que, mesmo diante desses obstáculos, o sistema possa funcionar de maneira coesa e segura (COULOURIS et al., 2013).

A relevância do consenso abrange várias operações críticas, como a seleção de líderes, a gestão de transações e a harmonização de estados entre os componentes distribuídos. Essas operações são essenciais para preservar a operacionalidade e a coerência dos dados em um sistema distribuído. A eleição de um líder, por exemplo, promove uma gestão centralizada em um agrupamento de servidores, enquanto a gestão de transações é fundamental para assegurar a consistência dos dados em sistemas de bancos de dados distribuídos.

(CHANDRA; TOUEG, 1996) definem o problema do consenso em três propriedades:

- **Finalização**: Cada processo eventualmente decide por algum valor.
- **Validade**: Se um processo decide pelo valor v, então v foi proposto por algum processo.
- **Concordância**: Dois processos corretos não decidem diferentemente.

2.5 Impossibilidade de FLP

A impossibilidade de FLP, demonstrada por (FISCHER; LYNCH; PATERSON, 1985), revela uma limitação fundamental dos sistemas distribuídos assíncronos: não é possível alcançar um consenso determinístico se pelo menos um componente falhar. Este resultado estabelece que, na presença de falhas, e sem suposições sobre o tempo de resposta ou a ordem de entrega das mensagens, não existe um algoritmo que garanta o consenso em todos os casos. A falha FLP destaca o desafio de distinguir entre componentes lentos e falhos, um problema crítico em sistemas onde o tempo de resposta pode variar significativamente.

Essencialmente, a identificação e a gestão de falhas são componentes fundamentais para assegurar a resiliência e a confiabilidade de sistemas distribuídos, sobretudo em ambientes assíncronos onde a impossibilidade FLP prevalece. Através do uso estratégico de detectores de falhas e outras técnicas adaptativas, é possível criar sistemas robustos capazes de alcançar consenso e manter a operacionalidade frente a desafios de sincronização e falhas imprevistas.

Os detectores de falhas não-confiáveis são uma solução prática para contornar a barreira da impossibilidade FLP, estes dispositivos operam como oráculos que sinalizam possíveis falhas entre os processos. Cada detector supervisiona um segmento específico do sistema e administra uma lista de processos suspeitos. O caráter "não-confiável" destes detectores é evidenciado pela possibilidade de falsos positivos, isto é, suspeitar erroneamente de um processo íntegro. A confiança entre processos, nesse contexto, é determinada pela ausência de suspeitas por parte dos respectivos detectores de falhas.

Uma outra estratégia para contornar o resultado da impossibilidade é considerar sistemas parcialmente síncronos, os quais são suficientemente mais fracos do que os sistemas síncronos para serem úteis como modelos de sistemas práticos e suficientemente mais fortes do que os sistemas assíncronos para que o consenso possa ser resolvido (COULOURIS et al., 2013).

2.6 Detectores de falhas não confiáveis

A identificação de erros envolve inspeções no estado do sistema para encontrar não conformidades. Isso é feito verificando o estado de erro provocado por falhas em componentes do sistema, onde a falha de um componente pode acarretar em uma falha no sistema como um todo, logo os detectores de falhas são cruciais na construção de sistemas distribuídos resistentes a falhas, fornecendo informações vitais sobre o estado dos componentes do sistema.

A detecção de falhas em sistemas distribuídos é um elemento chave para garantir a confiabilidade e a estabilidade desses sistemas complexos, especialmente considerando-se sistemas assíncronos e o uso de detectores de falhas para contornar a impossibilidade de FLP (FISCHER; LYNCH; PATERSON, 1985; CHANDRA; HADZILACOS; TOUEG, 1996).

O conceito de detectores de falhas não confiáveis foi introduzido por (CHANDRA; TOUEG, 1996), descrevendo-os como oráculos em sistemas distribuídos, oferecendo indicações aos processos sobre possíveis falhas presentes no sistema. A cada chamada, esses detectores apresentam um grupo de identidades dos processos dos quais desconfiam que possam ter falhado.

A caracterização de "não confiáveis" advém da possibilidade de esses detectores cometerem equívocos, podendo falhar ao identificar processos defeituosos ou, inversamente, ao levantar suspeitas sobre processos que na verdade estão operando corretamente. Assim sendo, foram definidas por duas propriedades fundamentais: completude e precisão.

Completude diz respeito à capacidade do detector de identificar corretamente todos os processos que falharam, enquanto **precisão** se refere à capacidade de não classificar incorretamente processos corretos como falhos.

Estas propriedades são essenciais para garantir que os sistemas que dependem desses detectores mantenham a consistência em suas operações e evitem bloqueios ou falhas indevidas. Na prática, estes detectores de falhas produzem como saída uma lista de processos considerados suspeitos.

Os detectores de falhas são classificados de acordo com duas propriedades de completude e quatro propriedades de precisão (CHANDRA; TOUEG, 1996).

- **Completude forte:** existe um tempo após o qual todo processo correto suspeita permanentemente de todo processo faltoso;
- **Completude fraca:** existe um tempo após o qual algum processo correto suspeita permanentemente de todo processo faltoso;
- **Precisão forte:** nenhum processo é suspeito antes de falhar;
- **Precisão fraca:** existe um processo correto que nunca será suspeito;
- **Precisão eventualmente forte:** em algum momento no futuro, todo processo correto não será suspeito por qualquer processo;
- **Precisão eventualmente fraca:** em algum momento no futuro, um processo correto não será suspeito por qualquer processo.

Tabela 1 apresenta as oito possíveis combinações entre as propriedades dos detectores de falhas. Um detector de falhas é dito perfeito se este apresenta as características de abrangência e precisão fortes e é denotado pelo símbolo P.

Tabela 1 – Classes de detectores de falhas e respectivos símbolos.

Completude	Precisão			
	Forte	Fraca	Eventual Forte	Eventual Fraca
Forte	Perfeito **P**	Forte **S**	Eventualmente Perfeito ◊ **P**	Eventualmente Forte ◊ **S**
Fraca	Fraco **Q**	Fraco **W**	Eventualmente Fraco ◊ **Q**	Eventualmente Fraco ◊ **W**

2.6.1 Outros tipos de Detectores de falhas

2.6.1.1 Detector de Falha Ômega (Ω)

O detector de falhas Ômega (Ω) é uma abstração fundamental em sistemas distribuídos para resolver o problema da eleição de líder em cenários onde falhas podem ocorrer, tal conceito foi proposto por (CHANDRA; HADZILACOS; TOUEG, 1996).

A especificação de Ω garante que, eventualmente, todos os processos corretos (i.e., não falhos) concordarão em um único processo correto como líder. Esta propriedade é conhecida como "Liderança Eventual".

A relevância de Ω se dá por ser o detector de falhas mais fraco capaz de resolver o problema de consenso em sistemas distribuídos, sob a condição de que a maioria dos processos esteja correto. A força de Ω reside na sua simplicidade e no fato de que ele não exige conhecimento prévio da afiliação ao sistema dos processos, diferenciando-se assim dos detectores de falhas ◊S e ◊W.

Propriedade Chave de Ω:

- **Liderança Eventual:** Existe um tempo após o qual todos os processos corretos concordam em um único líder, permanecendo este como líder daí em diante.

2.6.1.2 Detector de Falhas de Quorum Sigma (Σ)

Por outro lado, o sistema Sigma Σ de detecção de falhas age estabelecendo, para cada processo que funciona corretamente, uma lista de processos vistos como confiáveis a qualquer tempo. Esse grupo de processos confiáveis deve atender a dois requisitos fundamentais:

- **Intersecção:** Garante que há pelo menos um processo correto comum entre quaisquer duas listas de processos confiáveis.
- **Completude:** Assegura que, eventualmente, a lista de processos confiáveis de cada processo correto incluirá todos os processos corretos.

A classe Σ é notável por ser o detector de falhas mais fraco necessário para implementar um registro confiável em qualquer ambiente distribuído, conforme discutido por (DELPORTE-GALLET; FAUCONNIER; GUERRAOUI, 2003).

Importância Conjunta de Ω e Σ

(DELPORTE-GALLET; FAUCONNIER; GUERRAOUI, 2003), ampliaram a compreensão da comunidade sobre esses detectores de falhas, demonstrando que o par $\langle \Omega, \Sigma \rangle$ constitui o conjunto de detectores de falhas mais fraco capaz de resolver o consenso (seja uniforme ou não) em ambientes de passagem de mensagens assíncronas. Este resultado sublinha a eficácia de combinar Ω e Σ para alcançar consenso em sistemas onde todos os processos, exceto um, podem falhar.

2.6.2 Qualidade de Serviço em Detectores de Falha

(CHEN; TOUEG; AGUILERA, 2002) introduziram um conjunto de métricas para avaliação da eficácia dos detectores de falhas, focando na agilidade da detecção de falhas autênticas e na minimização de alarmes falsos. Essas métricas são exploradas dentro do contexto de um sistema composto por apenas dois processos, onde um processo, denominado q, é responsável por monitorar outro processo, p, sem que haja falhas em q, permitindo-lhe identificar possíveis falhas em p.

Para avaliar a performance dos detectores de falhas em sistemas distribuídos, este estudo propõe a análise das respostas emitidas por um detector q em um determinado instante, as quais podem indicar suspeita (S) ou confiança (T) em relação à operacionalidade de um processo p.

Ao longo do tempo, q pode alternar suas respostas entre S (indicando suspeita de falha em p) e T (indicando confiança na operacionalidade de p). Esta alternância é conhecida como **transição**.

Uma transição-S é identificada quando a saída do detector q muda de T (confiança) para S (suspeita), sugerindo uma percepção de potencial falha em p. Por outro lado, uma transição-T ocorre quando a saída de q muda de S (suspeita) para T (confiança), refletindo a recuperação da confiança na operacionalidade de p.

Estas definições de transição-S e transição-T são fundamentais para definir métricas de Qualidade de Serviço (QoS) que caracterizam a velocidade e a precisão com que os detectores de falhas identificam e reagem a defeitos.

Assume-se que o número de transições em qualquer período de tempo é finito, proporcionando uma base para avaliar a rapidez e precisão dos detectores de falhas através de dois conjuntos de métricas: **primárias e derivadas**.

2.6.2.1 Métricas primárias

- **Tempo de Detecção (*Detection Time - TD)***: é o tempo decorrido desde quando o processo p entra em colapso até o instante em que q suspeita de p permanentemente. Mais precisamente, é o tempo entre o momento em que p entrou em colapso e o momento da transição-S final, sem mais transições posteriores. A Figura 3 mostra um diagrama de estados representando a detecção de falhas em sistemas distribuídos, onde um processo p é monitorado por um detector de falhas denominado processo q. No eixo vertical, temos os estados de p, que podem ser “correto” ou “falho”. No eixo horizontal, temos as percepções do detector de defeitos em q, alternando entre “confia” e “suspeita”.

- **Tempo de Retorno do Erro (*Mistake Recurrence Time - TMR*)**: é o tempo decorrido entre dois erros de suspeita consecutivos. Mais precisamente, representa o tempo entre uma transição-S e a próxima transição-S.

- **Duração do Erro (*Mistake Duration - TM*)**: é o tempo decorrido até que o detector corrija um erro de suspeita. Mais precisamente, representa o tempo entre uma transição-S e a próxima transição-T.

2.6.2.2 Métricas derivadas

- **Taxa Média de Erro (*Average Mistake Rate - λM*)**: é a taxa de erros cometidos por um detector de defeitos, isto é, é a média de S-transitions por unidade de tempo. É importante para aplicações onde cada erro do detector (cada S-transition) resulta em um custo de interrupção.

- **Probabilidade da Precisão da Consulta (*Query Accuracy Probability - P.A.*)**: é a probabilidade de que a saída de um detector esteja correta em um tempo aleatório. Essa métrica é

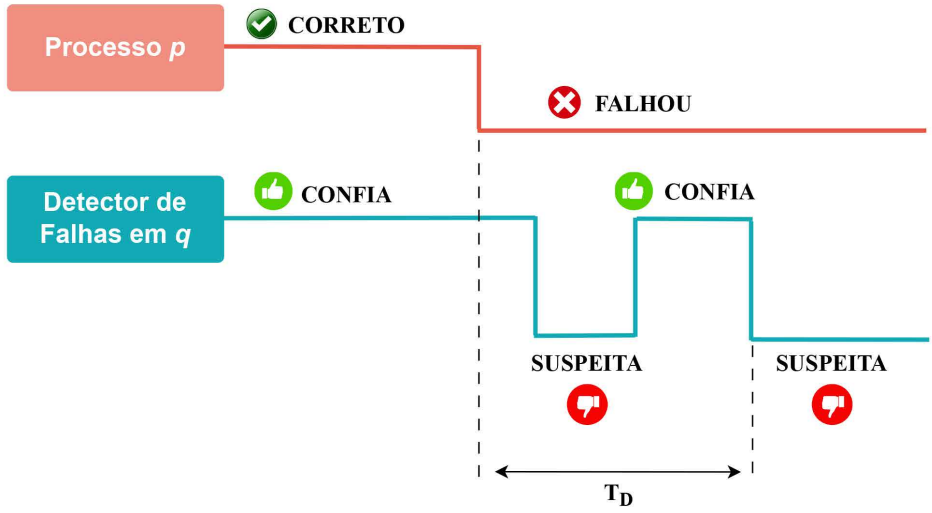

Figura 3 – Métrica Tempo de Detecção (Td).

importante para aplicações que interagem com o detector de falhas, fazendo requisições em tempos aleatórios. Algumas aplicações podem progredir apenas durante períodos bons – nos quais o detector não comete erros. Essa observação conduz a outras duas métricas descritas na sequência.

- **Duração do Período Bom (*Good Period Duration - TG*)**: é o intervalo do período bom, aquele em que o detector não comete erros. Representa, mais precisamente, o tempo entre uma transição-T e a próxima transição-S.
- **Duração do Período Bom à Frente (*Forward Good Period Duration - TFG*)**: é o tempo decorrido a partir de um tempo aleatório no qual q confia em p até a próxima transição-S (é a parte restante do período bom).

A Figura 4, ilustra a representação gráfica das Métrica de QoS (TM, TG, TMR e TFG) em um contexto de sistemas de detecção de falhas. A imagem descreve um processo onde um detector de falhas em um nó q observa outro processo ou nó p dentro do sistema, além da alternância entre estados de confiança e suspeita do detector de falhas em relação ao processo p, bem como o ciclo de vida das suspeitas e suas correções.

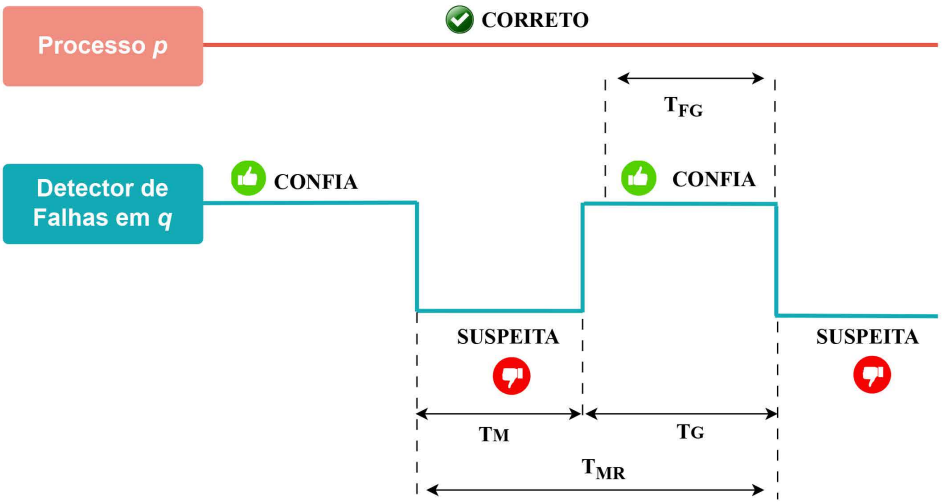

Figura 4 – Métrica de QoS (TM, TG, TMR e TFG).

2.7 Tecnologias

Nesta seção são apresentadas as tecnologias utilizadas na implementação do sistema proposto, detalhando os componentes tecnológicos que foram escolhidos para desenvolver e integrar os detectores de falhas no ambiente IoT, abordando desde a plataforma até o protocolo necessário para garantir a eficácia e a robustez do sistema em cenários suscetíveis a desastres.

2.7.1 Constrained Application Protocol - CoAP

O Protocolo de Aplicação *Constrained* - (CoAP) é um protocolo da camada de aplicação, projetado para redes de dispositivos restritos, como os utilizados em sistemas de Internet das Coisas (IoT). Ele foi desenvolvido pela *Internet Engineering Task Force* - (IETF) e definido no RFC 7252 (RFC Editor, 2023).

O CoAP é projetado para permitir que dispositivos simples se comuniquem interativamente sobre a Internet, operando sobre *User Datagram Protocol* - (UDP) para manter a simplicidade e a eficiência, características cruciais para dispositivos com recursos limitados de energia, memória e processamento. Esse protocolo é similar ao *Hypertext Transfer Protocol* - (HTTP) em termos de oferecer um método de requisição/resposta, mas é otimizado para a baixa potência e ambientes de rede de baixa largura de banda. Ele suporta descobertas de recursos e métodos semelhantes ao HTTP (GET, POST, PUT, DELETE), mas com uma menor sobrecarga de dados.

A Figura 5 ilustra a Camada Abstrata do protocolo CoAP, que descrever a pilha do protocolo, voltado para redes restritas como a Internet das Coisas (IoT). A imagem direcionada de baixo para cima, mostra o uso do UDP como camada de transporte de mensagens, sobre o qual as mensagens CoAP são construídas. Acima das mensagens CoAP, há a camada de interação Request/Response, que trata das requisições CoAP e das respostas correspondentes. E no topo, a camada de Aplicação, onde os aplicativos finais interagem com o protocolo CoAP para realizar suas operações junto ao usuário final.

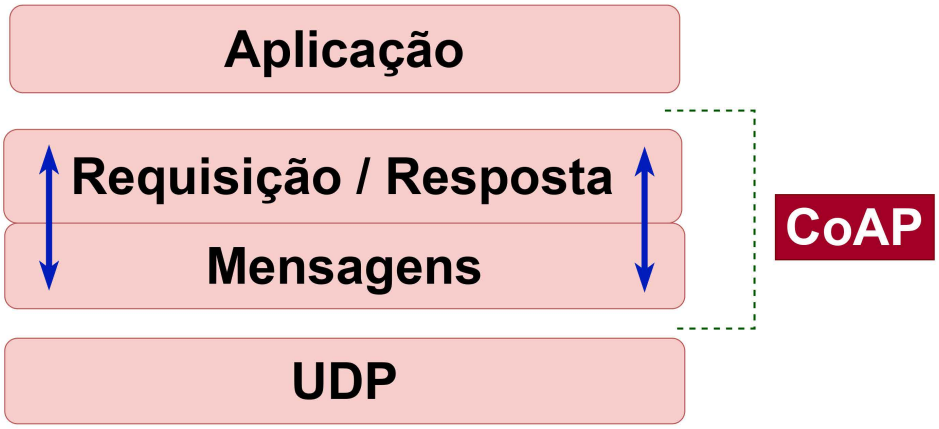

Figura 5 – Camadas Abstratas do CoAP.

O CoAP também oferece uma camada de segurança opcional através do *Datagram Transport Layer Security* - (DTLS), permitindo comunicação criptografada e autenticada em ambientes de rede inseguros. Outra característica importante deste protocolo é possuir um cabeçalho pequeno, limitado a 4 bytes, este cabeçalho é constituído de, entre outras coisas, um identificador da mensagem, um token e um tipo da mensagem. O identificador serve para o CoAP detectar mensagens em duplicidade. O token por sua vez, permite o CoAP relacionar as mensagens de requisição com a sua resposta.

O Disaster-FD utiliza o protocolo CoAP em sua implementação devido à sua adequação para ambientes de rede de sensores IoT, que são frequentemente limitados em termos de recursos, como largura

de banda, energia e capacidade de processamento. O CoAP é um protocolo leve, projetado especificamente para dispositivos com recursos restritos, permitindo comunicação eficiente e confiável em redes de sensores, mesmo em cenários de desastres onde a rede pode ser instável ou intermitente. Além disso, o CoAP suporta operações de *multicast*, o que é útil para enviar alertas ou atualizações para múltiplos dispositivos de maneira eficiente, garantindo uma resposta rápida em situações críticas.

2.7.1.1 Modelo de Mensagens

O protocolo CoAP facilita a comunicação entre dispositivos em redes IoT, empregando o protocolo UDP para a troca de informações tanto solicitações quanto respostas entre entidades de rede. O CoAP distingue-se por seu cabeçalho enxuto de apenas 4 bytes, otimizado para eficiência, que pode ser expandido com opções binárias compactas e um segmento de dados úteis (*payload*). Cada troca de mensagem é unicamente identificada por um ID de 16 bits, essencial para o reconhecimento de mensagens repetidas e para garantir a entrega confiável de dados.

Para assegurar a entrega confiável das mensagens, o CoAP define mensagens como Confirmáveis "CON" (*Confirmable*), isso aumenta a confiabilidade na comunicação, pois cada mensagem prevê uma resposta do dispositivo receptor através do envio de um "ACK" (*Acknowledgement*), contendo o ID original da mensagem. Em situações onde a resposta esperada não pode ser fornecida, o receptor envia uma mensagem de Reset (RST) ao invés de um ACK.

A Figura 6 ilustra dois exemplos de trocas de mensagens entre um cliente e um servidor utilizando o protocolo CoAP. No lado esquerdo, o cliente envia uma requisição "CON" (*Confirmable*) com um comando GET para "/temperatura" e recebe uma resposta "ACK" (*Acknowledgement*) com o status "2.05 *Content*" e o dado "22.5 C", indicando uma resposta bem-sucedida com a temperatura. No lado direito, uma requisição semelhante resulta em "4.04 *Not Found*", significando que o recurso solicitado não foi encontrado no servidor. Cada mensagem contém um *token* para correlacionar requisições e respostas.

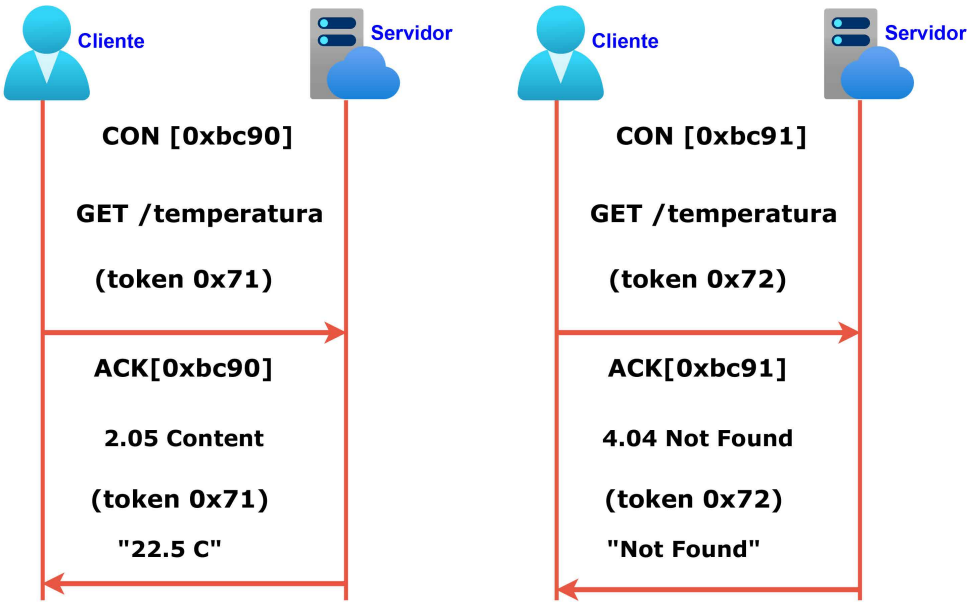

Figura 6 – Solicitações GET tipo CON.

Caso o servidor receba uma mensagem do tipo CON, mas não conseguir responder a esta solicitação imediatamente, ele enviará um ACK vazio para o cliente. Quando o servidor estiver pronto para responder a esta solicitação, ele enviará um novo CON ao cliente e o cliente responderá uma mensagem confirmável com confirmação, esse ACK serve apenas para confirmar a mensagem CON recebida do servidor, independentemente da solicitação ou resposta de transporte da mensagem CON.

Para comunicações que dispensam confirmação de recebimento, onde a prioridade é a minimização da latência a estratégia empregada envolve o envio de mensagens Não-Confirmáveis "NON" (*Non-Confirmable*), que apesar de não requererem ACK, ainda são identificadas por um ID para controle de duplicidade. Caso o receptor encontre dificuldades ao processar tal mensagem, ele pode enviar um RST como forma de resposta.

A Figura 7 ilustra um exemplo de uma troca de mensagens CoAP do tipo NON (*Non-confirmable*), onde o cliente envia uma requisição GET para o recurso "/temperatura" sem exigir confirmação de recebimento. O servidor responde com uma mensagem NON contendo o status "2.05 *Content*" e o valor da temperatura "22.5 C". Essa comunicação NON não requer ACKs, permitindo um fluxo de mensagens mais leve e rápido em cenários onde a confiabilidade não é crítica.

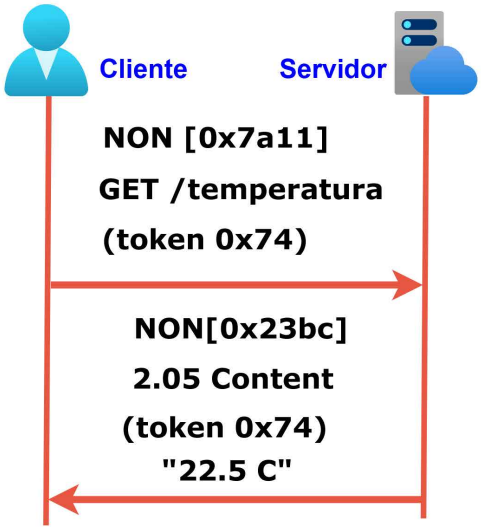

Figura 7 – Solicitações GET tipo NON.

2.7.1.2 Aplicação Prática do CoAP: Estudo de Caso

O protocolo CoAP desempenha um papel fundamental no avanço das tecnologias de Internet das Coisas (IoT), especialmente em aplicações voltadas para *Smart Cities e Smart Grids*, onde a eficiência, a interoperabilidade e a baixa sobrecarga de comunicação são essenciais. O trabalho apresentado por (MELLO; SILVA; LEITHARDT, 2019) ilustra a aplicabilidade e a eficácia do protocolo CoAP ao abordar desafios de comunicação em dispositivos de IoT com recursos limitados.

A implementação deste projeto foi possível graças à biblioteca libcoap2, que se destaca por sua habilidade em oferecer uma comunicação M2M (*Machine to Machine*) leve e eficiente, uma característica intrínseca do CoAP. O complemento desta tecnologia com o desenvolvimento de um cliente em Python especificamente para enviar requisições CoAP enfatiza a versatilidade e a facilidade de integração do protocolo em variados contextos tecnológicos. Este aspecto técnico não apenas facilitou a comunicação efetiva entre os protocolos CoAP e *Open Smart Grid Protocol* - (OSGP) mas também assegurou a adaptabilidade do sistema a diferentes tipos de dispositivos, mesmo aqueles com severas restrições de recursos.

Os resultados obtidos no estudo destacam a eficiência do gateway COSGP-IoT-SG em mediar a comunicação entre os protocolos CoAP e OSGP, além disso, o estudo realça a necessidade de avançar as pesquisas para simulações em redes elétricas reais e testes em dispositivos com capacidades ainda mais limitadas, abrindo caminho para inovações futuras no domínio da IoT.

A sinergia entre a biblioteca libcoap2 e a implementação de um cliente Python para requisições CoAP ilustra um modelo eficaz para a exploração do potencial pleno do CoAP, estabelecendo um marco

para o desenvolvimento de aplicações integradas para *Smart Cities e Smart Grids*. Este estudo não apenas confirma a adaptabilidade e a relevância do CoAP em ambientes de IoT mas também serve como inspiração para futuras inovações, promovendo soluções mais inteligentes, sustentáveis e integradas.

2.7.2 Biblioteca Californium

A Biblioteca Californium é um framework em Java dedicado à implementação do protocolo CoAP, essencial para aplicações de Internet das Coisas (IoT). Desenvolvida com o intuito de fornecer uma solução robusta e altamente escalável para a comunicação entre dispositivos IoT, a Californium se destaca por sua eficiência e adaptabilidade em diversos cenários de uso. Este framework suporta todos os recursos críticos do CoAP, incluindo descoberta de recursos, observação de recursos e a troca segura de mensagens através do *Datagram Transport Layer Security* - (DTLS).

Diferenciando-se por seu design otimizado, a Californium demonstrou um desempenho significativamente superior quando comparado a servidores web HTTP de alta performance, alcançando uma taxa transferência 33 a 64 vezes maior, conforme apresentado no trabalho proposto por (KOVATSCH; LANTER; SHELBY, 2014). Isso é particularmente relevante no contexto da IoT, onde a eficiência na comunicação e a capacidade de escalar para gerenciar um vasto número de dispositivos conectados são essenciais. Graças à sua natureza leve e eficiente na utilização de recursos, a Californium é especialmente adequada para dispositivos com recursos limitados, como sensores e atuadores, facilitando assim a implementação de aplicações IoT complexas e distribuídas.

A arquitetura da Biblioteca Californium é notavelmente rica e extensível, apresentando diversas classes e métodos que permitem aos desenvolvedores adaptar e expandir facilmente as funcionalidades do framework para atender às necessidades específicas de suas aplicações. Essa flexibilidade é complementada por uma comunidade ativa e engajada, hospedada pela (ECLIPSE FOUNDATION, 2024), que oferece suporte contínuo através de correções de bugs, melhorias de performance e adição de novas funcionalidades, contribuindo para a evolução constante do projeto. Além disso, a estrutura do Californium foi cuidadosamente projetada para tirar pleno proveito dos sistemas multi-core modernos, garantindo que aplicações IoT possam operar com a máxima eficiência.

No artigo (KOVATSCH; LANTER; SHELBY, 2014), a arquitetura de implementação do framework Californium, é descrito em 03 (três) estágios sendo:

- **Estágio 1**: Rede, este estágio é responsável por receber e enviar um conjunto de dados - (*bytes*) pela rede. Ele abstrai o protocolo de transporte, que geralmente é UDP ou DTLS para CoAP, permitindo uma manipulação flexível das mensagens de entrada e saída. Um detalhe importante é a configuração do número de threads para movimentação de dados através do socket, que pode variar dependendo da plataforma, para otimizar a taxa de transferência de dados. No Windows, por exemplo, a configuração de quatro threads de recepção e quatro de envio quase dobra a taxa de dados alcançável em comparação com uma configuração de um único thread para cada.

- **Estágio 2**: Protocolo CoAP, neste estágio ocorre a execução propriamente dita do protocolo, e é aqui que a maior parte do processamento das mensagens ocorre. Ele inclui várias subcamadas, como a camada de confiabilidade (que gerencia retransmissões), a camada de tokens (para correspondência única de respostas às solicitações), a camada observe (para gerenciamento de observações de recursos) e a camada de transferência em blocos (para fragmentação de mensagens). A arquitetura mantém um pool de *threads*, que geralmente correspondente ao número de núcleos do CPU, permitindo o processamento paralelo das mensagens.

- **Estágio 3**: No último estágio, a lógica de negócios específica da aplicação é processada. Isso pode incluir a manipulação de solicitações CoAP por recursos específicos no servidor IoT ou o envio

de solicitações por um cliente CoAP. Diferentemente dos estágios anteriores, o estágio de lógica de negócios é altamente dependente da aplicação e pode ser configurado para utilizar *pools* de *threads* específicos para tratamento de solicitações, dependendo das necessidades de performance e priorização da aplicação.

O processamento das mensagens CoAP flui através desses estágios de maneira eficiente, desde a recepção das mensagens do dispositivo IoT na rede, passando pelo processamento do protocolo CoAP, até a execução da lógica de negócios específica da aplicação. Cada estágio é desacoplado dos outros por filas de eventos, permitindo que se ajustem dinamicamente e operem de forma independente para otimizar o desempenho e a escalabilidade do sistema.

Esse modelo de processamento em estágios permite que o framework Californium lide com uma grande quantidade de mensagens CoAP de forma eficaz, aproveitando os recursos do servidor de nuvem IoT e garantindo a escalabilidade necessária para suportar o crescente número de dispositivos IoT conectados. No artigo analisado, foram conduzidos testes comparativos entre dois cenários distintos de gerenciamento de conexão: com keep-alive e sem keep-alive. No cenário com keep-alive, uma única conexão TCP é reutilizada para várias requisições, minimizando os atrasos de *handshakes* TCP e os problemas de congestionamento.

A análise incluiu servidores HTTP como Apache HTTP Server, Tomcat, Node.js, Grizzly, Jetty e Vert.x. Dentre estes, o Vert.x se destacou por manter um alto *throughput* mesmo sob carga de muitas conexões simultâneas. No entanto, a biblioteca Californium, baseada no protocolo CoAP, superou todas essas soluções HTTP, alcançando até quase 400 mil requisições por segundo. Esse desempenho evidencia a capacidade superior do CoAP para lidar com as comunicações leves e eficientes que são características dos cenários de IoT (KOVATSCH; LANTER; SHELBY, 2014).

Por outro lado, o cenário sem *keep-alive* simula a prática comum em IoT, na qual dispositivos fazem conexões breves que são terminadas após a transmissão dos dados. Esta abordagem é crucial em IoT para preservar a energia dos dispositivos. Em tal configuração, a necessidade de estabelecer novas conexões TCP para cada requisição apresenta uma desvantagem significativa para os servidores HTTP, devido ao *overhead* envolvido. Em contrapartida, o CoAP, operando sobre UDP, não enfrenta este problema, mantendo um alto *throughput* (KOVATSCH; LANTER; SHELBY, 2014).

Em resumo, a comparação realizada no artigo demostra a superioridade do protocolo CoAP, implementado pelo framework Californium, em relação aos servidores web baseados em HTTP, especialmente em termos de escalabilidade, eficiência e adequação ao ambiente restritivo de recursos dos dispositivos IoT. Isso posiciona o Californium como uma solução altamente eficaz para o desenvolvimento de aplicações IoT que demandam comunicações leves e eficientes.

2.7.2.1 Aplicação Prática da Biblioteca Californium: Estudo de Caso

A biblioteca Californium representa um avanço significativo para o desenvolvimento de aplicações na Internet das Coisas (IoT), oferecendo uma implementação robusta e flexível do protocolo CoAP, em Java. Este protocolo, desenvolvido especificamente para dispositivos de baixa potência e limitações de capacidade, permite uma comunicação eficiente M2M (*Machine to Machine*), essencial para a implementação de serviços inteligentes em contextos como *Smart Cities*.

O trabalho apresentado por (FILHO; FILHO; GOMES, 2013), explora a biblioteca Californium dentro de um *framework* projetado para integrar objetos inteligentes a redes sociais, promovendo o desenvolvimento de aplicações IoT com uma visão voltada para a facilidade de uso e extensibilidade. Este *framework*, através da utilização da Californium, não apenas facilita a comunicação CoAP entre aplicativos e dispositivos (nós) mas também exemplifica sua aplicação em um protótipo que interage com o Facebook, uma das maiores plataformas de redes sociais.

A implementação deste protótipo demonstra de forma prática a adaptabilidade e eficiência da biblioteca Californium. Utilizando dispositivos reais e virtuais, como sensores integrados via TelosB/TinyOS e servidores CoAP emulando sensores em um ambiente desktop, a arquitetura do protótipo exemplifica uma comunicação eficiente e adaptada às restrições dos dispositivos IoT. Os sensores são conectados ao servidor central através de um router, facilitando a integração de dispositivos em uma rede distribuída, refletindo um cenário realista de aplicação em *Smart Cities*. Além disso, a escolha da biblioteca Californium permitiu a incorporação de comunicação CoAP em uma estrutura modular do sistema, evidenciando a flexibilidade do *framework* em suportar diferentes tecnologias e protocolos.

A implementação baseada em padrões de design como Bridge, para a criação de *drivers* de comunicação, reforça a capacidade de extensão do sistema, permitindo a adição de suporte a novas redes sociais e tipos de dispositivos inteligentes sem a necessidade de grandes modificações na arquitetura existente. Este estudo de caso ressalta o papel fundamental da biblioteca Californium na promoção da interoperabilidade e na redução de sobrecargas na comunicação entre dispositivos e aplicações de IoT. A capacidade de facilitar a integração de dispositivos de baixa potência em aplicações urbanas inteligentes, mantendo a comunicação eficiente e minimizando o consumo de energia, destaca a biblioteca como uma ferramenta essencial para o desenvolvimento futuro da Internet das Coisas, especialmente em aplicações que demandam a interação com redes sociais para a criação de serviços inovadores e personalizados.

2.7.3 Plataforma Iot-Lab

A IoT-LAB emerge como uma infraestrutura de pesquisa no campo da Internet das Coisas (IoT), oferecendo uma plataforma para o desenvolvimento e teste de tecnologias IoT em um ambiente realista e em larga escala. Descrita detalhadamente em (ADJIH et al., 2015), esta infraestrutura fornece um *testbed* aberto que facilita experimentos em redes IoT complexas, simulando a diversidade e a complexidade da Internet global. A plataforma destaca-se pela sua vasta rede de mais de 2.700 nós de sensores e 117 robôs móveis espalhados por seis locais na França, tornando-se um recurso valioso para pesquisadores em todo o mundo.

A IoT-LAB suporta um leque diversificado de protocolos de comunicação sendo alguns deles (CoAP, 6LoWPAN e MQTT) essenciais para a interoperabilidade e eficiência das redes IoT. Além disso, permite a execução de experimentos federados em múltiplos locais, oferecendo uma perspectiva única sobre o desempenho e a escalabilidade de soluções IoT em um contexto global. A infraestrutura possibilita aos pesquisadores explorar o impacto da mobilidade nos protocolos de rede, graças à inclusão de nós móveis que podem ser configurados em diversas topologias.

Um dos aspectos mais notáveis da IoT-LAB é a sua interface de usuário intuitiva com ferramentas disponíveis para facilitar a criação e gerenciamento de experimentos. Os usuários podem acessar o portal web da IoT-LAB, um ponto central para a criação de contas, reserva de nós e configuração de experimentos. Este portal web é complementado por uma API RESTful e ferramentas de linha de comando (CLI), que oferecem flexibilidade e eficiência na interação com o *testbed*. Através desses recursos, os usuários podem facilmente reservar um número específico de nós em um ou vários locais, carregar *firmware* personalizado e monitorar o desempenho dos seus experimentos.

A escolha de utilizar a plataforma IoT-LAB para testes é devido à sua capacidade de simular um ambiente realista com diversos dispositivos distribuídos em várias regiões. Essa configuração permite o monitoramento federado e aproxima o cenário de testes das condições reais, com sensores instalados em locais diferentes, o que é crucial para avaliar a eficácia do sistema em situações semelhantes às de um ambiente real. Além disso, a IoT-LAB fornece um ambiente controlado onde diferentes configurações e parâmetros podem ser ajustados para avaliar a robustez e a eficácia do sistema de detecção de falhas.

2.7.3.1 Estrutura da Plataforma

A infraestrutura do IoT-LAB é composta por um local principal em Grenoble, que atua como o núcleo de controle e coordenação, e locais distribuídos em vários sites (Grenoble, Lille, Saclay e Strasbourg) de implantação que estão interligados por uma Rede Privada Virtual (VPN). A VPN permite que esses locais diferentes se comuniquem de forma segura e eficiente, criando um ambiente de teste unificado e integrado, apesar da dispersão geográfica.

No trabalho apresentado por (ADJIH et al., 2015), é mencionado que o servidor mestre localizado em Grenoble, desempenha funções críticas para o gerenciamento da plataforma, incluindo:

- **Autenticação do Usuário**: Há um sistema de autenticação baseado em um diretório *Lightweight Directory Access Protocol* - LDAP, que é um protocolo de software aberto para acessar e manter serviços de informações distribuídas, como diretórios de usuários e redes. Este sistema é responsável por verificar as credenciais dos usuários e garantir que apenas usuários autorizados possam agendar e executar experimentos.

- **Sistema de Nomes de Domínio (DNS)**: O local principal opera seu próprio sistema DNS, facilitando a resolução de nomes e o gerenciamento de endereços IP dentro da infraestrutura do IoT-LAB. Isso é essencial para a comunicação entre os servidores e os vários nós de sensores e robôs móveis.

- **Interface RESTful**: A principal aplicação do site oferece uma interface RESTful, que expõe as interfaces do *testbed* para o usuário. Isso permite que os usuários interajam com o sistema de forma programática, usando padrões de web para criar, modificar e deletar recursos, que neste caso são os experimentos e configurações de teste.

- **Despachante de Requisições**: Um despachante é responsável por encaminhar as solicitações dos usuários para os diferentes locais de implantação. Isso garante que as requisições sejam tratadas eficientemente e que os recursos necessários estejam disponíveis para os experimentos dos usuários em qualquer um dos locais de implantação.

- **Software de Agendamento OAR**: O local principal interage com o OAR, um software de gerenciamento de recursos open-source para grandes clusters. O OAR é responsável pelo agendamento de experimentos e alocação de recursos, administrando a infraestrutura de hardware de forma a maximizar sua utilização e evitar conflitos entre experimentos simultâneos. Esse software é encarregado de iniciar e parar experimentos, manejando a alocação de nós e a programação temporal dos testes.

Esses elementos de infraestrutura trabalham juntos para oferecer uma plataforma poderosa e flexível para experimentos em IoT, suportando uma ampla variedade de cenários de teste e pesquisa. A capacidade de gerenciar eficientemente a alocação de recursos e agendamento de experimentos é um aspecto crucial do IoT-LAB, permitindo que pesquisadores e desenvolvedores realizem experimentos em larga escala em um ambiente controlado e reproduzível.

Conforme demonstrado na Figura 8, a infraestrutura do IoT-LAB é visualizada, destacando a interconexão e o funcionamento integrado dos componentes da plataforma. Esta figura esquematiza o acesso dos usuários ao sistema, seja através do portal web ou via interface de linha de comando (CLI), e apresenta a estrutura centralizada no site mestre, que gerencia autenticação e comunicação.

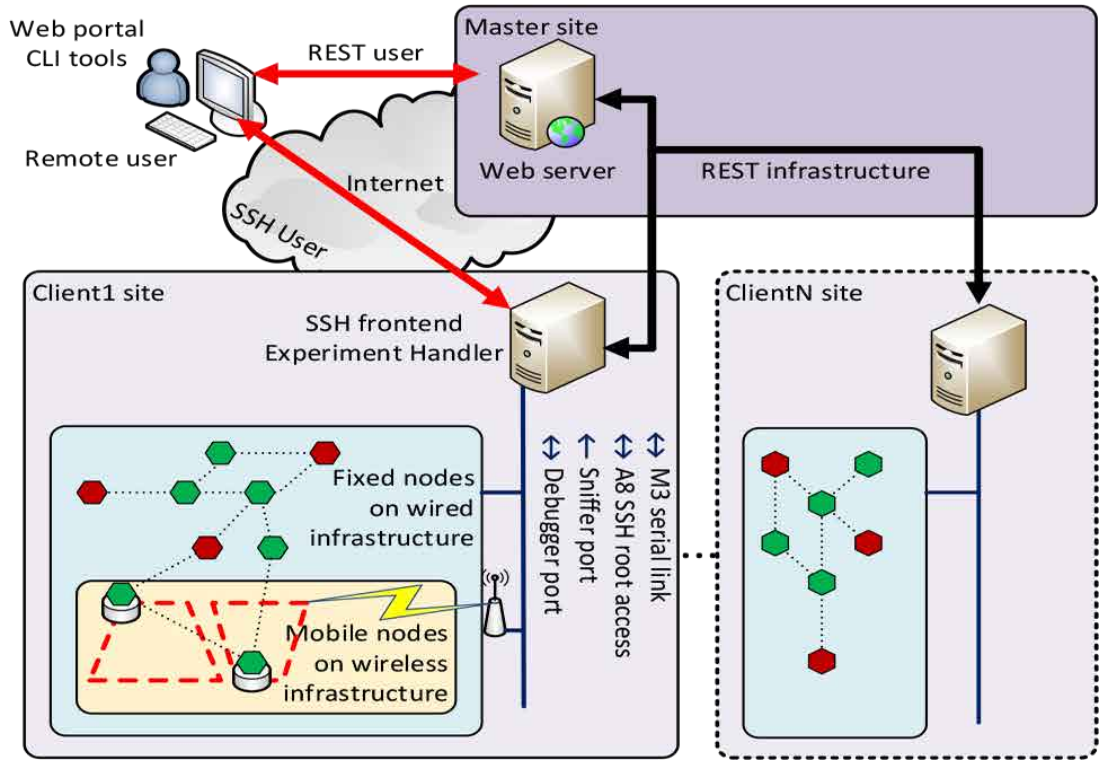

Figura 8 – Infraestrutura Iot-LAB.

Fonte: (ADJIH et al., 2015)

2.7.3.2 Ferramenta Command-Line Interface (CLI)

A ferramenta CLI do IoT-LAB é um cliente da API REST que permite executar as principais funcionalidades do *testbed* por meio de linha de comando. Facilita a automação e a criação de scripts de experimentos, cobrindo funções desde o armazenamento de credenciais até a gestão do ciclo de vida dos experimentos e interação com os nós. A instalação é simplificada através do comando `pip`. Os comandos disponíveis permitem gerenciar informações de status do *testbed*, submeter e controlar experimentos, interagir com os nós para gerenciamento de energia e atualização de *firmware*, e configurar perfis de monitoramento de consumo e rádio. Com essas ferramentas, os usuários podem executar uma variedade de ações, como reservar nós, iniciar e parar experimentos, tudo a partir da linha de comando.

2.7.3.3 Acessibilidade e Interação do Usuário na Plataforma IoT-LAB

A Plataforma IoT-LAB oferece um ambiente de teste aberto e versátil, projetado para simplificar a experiência do usuário na implementação e gerenciamento de experimentos em IoT. Os pesquisadores iniciam criando uma conta no site do IoT-LAB ou utilizando suas credenciais existentes do OneLab, garantindo uma entrada segura e personalizada na plataforma.

A interface do usuário é projetada para ser intuitiva: uma vez conectados, os usuários podem navegar facilmente pelo portal web para reservar e configurar nós sensoriais, selecionando-os de um mapa interativo que exibe a disponibilidade em tempo real dos nós fixos e móveis, cada um com suas características específicas, como localização e tipo de chip de rádio. Esta etapa inicial é crucial para definir as premissas de um experimento.

Os participantes podem reservar nós e enriquecer a execução dos seus projetos com recursos avançados de monitoramento, como análise do consumo de energia e ferramentas para interceptação de sinais de rádio. Eles têm a opção de escolher a fonte de alimentação dos nós, alternando entre baterias e a rede

elétrica, e selecionar as imagens de *firmware* específicas para serem carregadas nos dispositivos (ADJIH et al., 2015).

Uma vez que a reserva e a configurações estejam completas, o agendador do IoT-LAB assume, iniciando o experimento automaticamente quando os nós estão disponíveis ou em um momento específico previamente definido. Durante a realização do experimento, os usuários retêm o controle sobre os nós, com a capacidade de reinicializar, reconfigurar ou atualizar o *firmware* em qualquer momento, aplicando essas ações tanto individualmente quanto em grupo.

A Figura 9 demonstra como criar um experimento no IoT-LAB por meio do portal web. Mostra os passos iniciais para configuração do experimento, incluindo a escolha de nós, definição de duração e seleção de firmware. Este processo é feito através de uma interface gráfica, facilitando o acesso e a gestão dos recursos disponíveis na plataforma. Essencialmente, a figura destaca a facilidade com que os usuários podem iniciar seus projetos de IoT, promovendo a interação eficiente com a infraestrutura do laboratório.

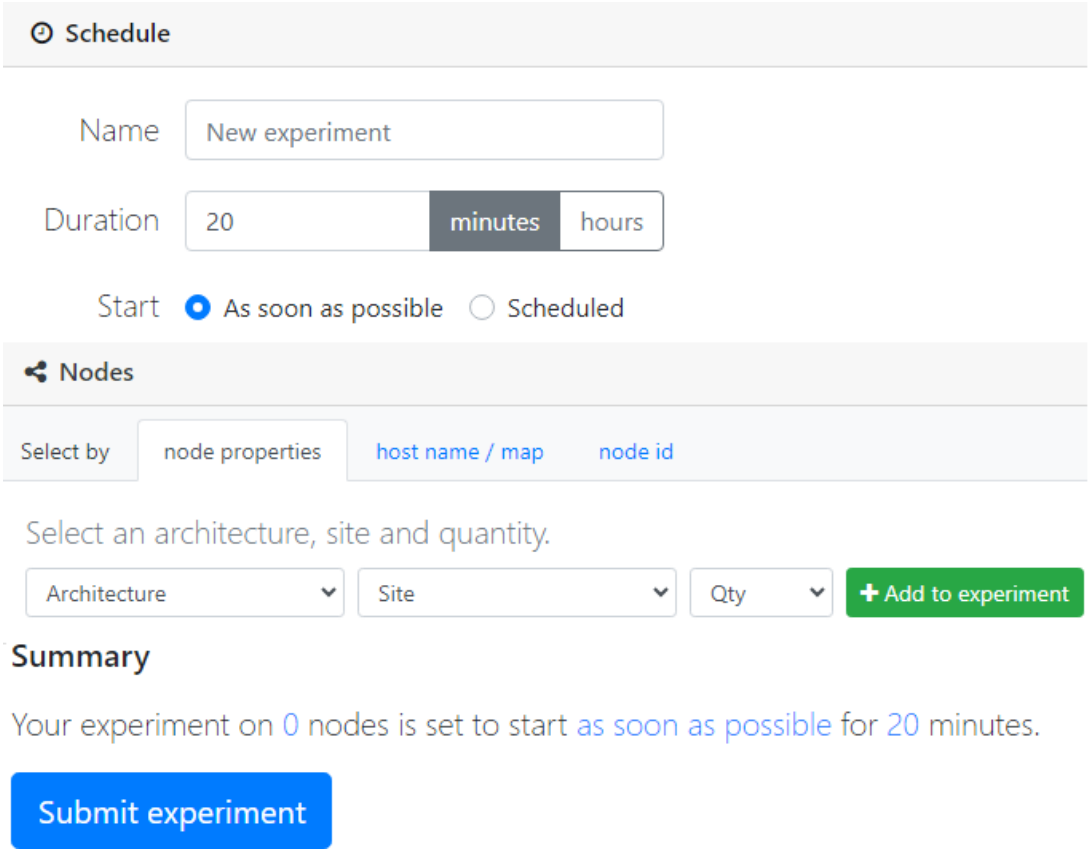

Figura 9 – Criar Experimento via Portal.

Fonte: https://www.iot-lab.info/testbed/experiment

A plataforma IoT-LAB desempenha um papel significativo no campo da IoT, servindo como um recurso aberto, robusto e versátil para o avanço da pesquisa e desenvolvimento. Ela permite a condução de experimentos em larga escala e em condições realistas, essenciais para o desenvolvimento de soluções IoT inovadoras e eficazes.

Essa infraestrutura não apenas contribui significativamente para o avanço acadêmico, mas também atua como um catalisador para inovações práticas na indústria de IoT, tornando-se um pilar fundamental para o progresso tecnológico e aplicação prática na área da Internet das Coisas.

2.7.3.4 Aplicação Prática na Plataforma IoT-LAB: Estudo de Caso

A pesquisa desenvolvida por (SASIKUMAR; NARAYANAN, 2020), evidencia a capacidade singular da plataforma FIT IoT-LAB em avançar o campo da Internet das Coisas (IoT). Esta infraestrutura, compreendendo mais de 2700 nós de sensores distribuídos em múltiplos locais, oferece um ambiente ideal para a experimentação com tecnologias IoT em um alcance amplo de aplicações.

No estudo em questão, a FIT IoT-LAB serviu como base para o teste e validação de uma rede de sensores, *Wireless Sensor Network* - WSN, focado na eficiência energética e na coleta de dados confiável. Utilizando o protocolo *Time-Slotted Channel Hopping* - TSCH dentro do padrão IEEE 802.15.4e, a pesquisa propôs uma nova abordagem para a detecção e agendamento de *"feather nodes"*, ou seja, nós que operam com uma carga de transmissão reduzida devido à ausência de nós filhos, visando a diminuição do consumo energético (SASIKUMAR; NARAYANAN, 2020).

Através de simulações conduzidas na plataforma IoT-LAB, o estudo conseguiu demonstrar resultados com a taxa de entrega de dados "end-to-end" alcançando 99,99%, enquanto significativamente reduzia o consumo energético e o tempo de atividade do rádio (SASIKUMAR; NARAYANAN, 2020). Tais resultados demostram a capacidade da plataforma não apenas para testar a eficácia de novas metodologias em ambientes controlados, mas também para explorar sua aplicabilidade em cenários realistas de grande escala.

Além de oferecer um ambiente robusto para a experimentação, a FIT IoT-LAB facilita o avanço do conhecimento científico e acadêmico, permitindo que pesquisadores de diversas áreas testem e validem suas hipóteses e soluções tecnológicas com uma abordagem prática. A eficácia do esquema proposto para WSN, validado pelos testes na IoT-LAB, exemplifica como esta plataforma pode acelerar o desenvolvimento de tecnologias IoT, contribuindo significativamente para a pesquisa e inovação.

Os resultados obtidos evidenciam a importância da FIT IoT-LAB como um catalisador para o estudo e desenvolvimento no campo da IoT, destacando seu papel crítico na avaliação de soluções que buscam endereçar os desafios contemporâneos de eficiência energética e confiabilidade na coleta de dados em redes de sensores sem fio. Este trabalho reforça o valor da plataforma IoT-LAB como um recurso essencial para a comunidade científica e acadêmica envolvida na exploração de novas fronteiras em tecnologias IoT.

2.8 Projeto ADMITS

O projeto *Architecting Distributed Monitoring and Analytics in IoT-based Disaster Scenarios* - (ADMITS), proposto por (PASQUINI et al., 2020) é um projeto inovador que mira na elaboração de uma arquitetura distribuída, englobando monitoramento, detecção de falhas e análise de dados em contextos de desastres, aproveitando o potencial da Internet das Coisas (IoT). A motivação para tal iniciativa surge da constatação do impacto crescente dos desastres, tanto naturais quanto humanos, como ataques terroristas e conflitos bélicos, que afetam milhões de pessoas e demandam extensos recursos dos governos para sua preparação, resposta imediata e reconstrução subsequente.

A solução proposta pelo ADMITS, que integra as tecnologias de *Internet das Coisas - (IoT), Computação em Névoa - (Fog) e Computação em Nuvem - (CN)*, representa um avanço significativo na gestão de desastres. Essa integração visa proporcionar respostas mais ágeis e eficazes, aprimorando a cooperação e a capacidade de previsão em operações emergenciais ligadas a desastres. A base da metodologia do ADMITS reside no desenvolvimento de um sistema apto a funcionar de maneira eficiente em ambientes que, por sua natureza, são dinâmicos e instáveis, características típicas dos cenários de desastre.

Um aspecto fundamental deste sistema é a detecção distribuída de falhas em ambientes IoT, que se concentra na identificação precisa e adaptável de falhas tanto nos dispositivos quanto na infraestrutura de rede. Esta detecção enfrenta desafios como a mobilidade dos dispositivos, partições de rede e as

limitações de energia e banda larga. Para tanto, é crucial o desenvolvimento de um mecanismo de detecção de falhas não apenas eficiente em termos energéticos e de comunicação, mas também flexível o suficiente para adaptar-se às mudanças ambientais e às suspeitas de falhas que daí surgem.

Em paralelo, destaca-se a análise de dados em tempo real como um componente chave para decifrar o grande volume de informações geradas pelos dispositivos IoT em cenários adversos. O projeto visa criar modelos analíticos robustos, aplicáveis tanto na borda da rede (*Fog*) quanto de forma centralizada (*Cloud*), com o intuito de possibilitar uma resposta rápida em momentos críticos. Esses modelos são elaborados para serem resilientes a falhas e adaptáveis às diversas condições ambientais, assegurando precisão e eficácia na antecipação e na identificação de desastres.

Além disso, a proposta inclui um sistema de processamento de fluxo em tempo real, que leva o processamento para mais próximo da borda da rede a fim de reduzir a latência e otimizar as respostas em tempo real. Este sistema é concebido para ser adaptativo e operar sobre dispositivos com recursos limitados, integrando modelos de detecção de falhas especificamente desenhados para este contexto ao sistema adaptativo de processamento de fluxo.

Por fim, a arquitetura integrada distribuída do ADMITS tem o objetivo de criar uma plataforma que suporte uma visão unificada do ciclo de vida dos sistemas de gestão de desastres. Isso abrange desde a coleta até o processamento de dados provenientes de dispositivos IoT e redes de sensores sem fio, utilizando recursos de processamento no *Fog* para diminuir o volume de dados enviados à Cloud. Essa estratégia não somente visa a eficiência no manuseio dos dados, mas também leva em consideração a dinâmica do ambiente, o consumo energético dos dispositivos, a volumosa produção de dados por parte da IoT e os desafios relacionados à implementação em larga escala.

Para avaliar e validar essa abordagem o ADMITS planeja realizar experimentos em *testbeds* de grande escala, como o FIT IoT-LAB, e fazer uso da plataforma DOJOT para o desenvolvimento de ambientes de teste que facilitem o monitoramento dos dispositivos IoT. Esta metodologia abrangente e integrada evidencia o potencial do ADMITS para transformar a gestão de desastres, fornecendo um sistema robusto, eficiente e adaptável, capaz de enfrentar os desafios impostos por desastres naturais e provocados pelo homem.

Capítulo 3

Trabalhos Relacionados

Este capítulo, apresenta trabalhos correlatos ao projeto proposto nesta dissertação, focando estudos que exploram detecção de falhas, resposta a desastres, e uso da Internet das Coisas (IoT) em ambientes críticos. Dentre eles, destacam-se o *Impact-FD* e o *Medley*, focados na detecção de falhas; o *Stab-FD*, que explora detecção de falhas adaptativa e cooperativa; e "Um modelo de rede de sensores sem fio auto-organizada e tolerante a falhas para Detecção de Incêndios", ressaltando a importância de sistemas resilientes para monitoramento ambiental. Também é examinado o "Querying on Federated Sensor Networks", evidenciando a eficiência na gestão de dados em um rede de sensores federada. Essa revisão destaca o contexto no qual o *Disaster-FD* se insere, sinalizando avanços e espaços para inovações.

3.1 Impact-FD: An Unreliable Failure Detector Based on Process Relevance and Confidence in the System

Neste estudo, propõe-se o *Impact-FD* (ROSSETTO et al., 2018), um detector de falhas para sistemas distribuídos, que fornece uma saída expressando o nível de confiança ou ***Trust Level*** do detector de falhas em relação ao sistema (ou conjunto de processos) como um todo, em um determinado instante, calculado por $TL_p^S(t)$, que representa a soma dos fatores de impacto dos processos corretos no instante (t), conforme detalhado na Equação 1.

$$TL_p^S(t) = \sum_i (I_i), \forall i \in T_p^S(t) \tag{1}$$

- $TL_p^S(t)$: Nível de Confiança do monitor p no conjunto de processos S no instante t.
- I_i: Fator de Impacto de cada processo i.
- $T_p^S(t)$: Conjunto de processos monitorados por p no conjunto S que não são suspeitos de falhas no instante t.

O *Impact-FD* utiliza um **fator de impacto** para definir a importância de cada processo, além de ajustar dinamicamente os valores de *timeout* com base em estimativas de tempo de chegada de mensagens, incorporando uma margem de segurança para minimizar erros e a degradação da qualidade do serviço (QoS).

Demonstrando sua versatilidade, o *Impact-FD* é adequado para sistemas distribuídos com diferentes características de rede e de processos, especialmente aqueles com heterogeneidade de nós. O cálculo estimado para a chegada da próxima mensagem no detector de falhas *Impact-FD*, utiliza um método proposto por (CHEN; TOUEG; AGUILERA, 2002). Este método baseia-se no histórico dos tempos de

chegada dos *heartbeats* anteriores, que são mensagens periódicas enviadas pelos processos para indicar que ainda estão funcionando, e incorpora uma margem de segurança (β) no cálculo.

A capacidade do *Impact-FD* de atribuir diferentes níveis de peso para os processos demostra a flexibilidade na configuração dos parâmetros, destacando-se como uma característica chave para sistemas distribuídos que requerem alta disponibilidade e confiabilidade.

Os resultados obtidos indicam que a incorporação de uma margem de segurança apropriada, juntamente com a calibração precisa dos parâmetros de limiar e tempo de espera, pode diminuir consideravelmente a frequência de erros e alertas falsos. Essa abordagem assegura também que a detecção de falhas se mantenha eficaz frente a condições desfavoráveis da rede.

Por fim, o **Limiar**, ou ***Threshold***, define o limite mínimo de confiabilidade para cada conjunto em S^*, matematicamente representado por $\{Th_1, Th_2, \ldots, Th_m\}$, no qual cada Th_i está relacionado ao nível mínimo de confiança necessário para um subconjunto de processos S_i.

No contexto do *Impact-FD*, uma variedade de métricas de Qualidade de Serviço - (QoS) é utilizada para avaliar sua eficácia e eficiência na detecção de falhas em sistemas distribuídos. Essas métricas incluem Tempo de Detecção (TD), Taxa Média de Erros (λR), e Probabilidade de Precisão de Consulta (P.A.), entre outras. Essas métricas são empregadas para medir a precisão e a rapidez com que o *Impact-FD* pode identificar falhas, bem como sua capacidade de minimizar falsos positivos e negativos, aspectos cruciais para manter a alta disponibilidade e a confiabilidade dos sistemas distribuídos.

Do ponto de vista de implementação, a tese de (ROSSETTO et al., 2018), oferece uma análise detalhada de vários algoritmos e estratégias para implementar o *Impact-FD*, incluindo métodos baseados em padrões de mensagens e temporizadores. A avaliação de desempenho, baseada em traces reais do PlanetLab, demonstra a capacidade do *Impact-FD* de tolerar falhas e suspeitas falsas de maneira eficaz, resultando em uma melhoria significativa da qualidade do serviço em comparação com os detectores de falhas não confiáveis tradicionais.

Além disso, um dos aspectos mais notáveis do *Impact-FD* é sua equivalência com os detectores de falhas fundamentais (Ω) e (Σ). Esta equivalência fornece uma base sólida para a aplicação do *Impact-FD* na resolução de problemas de consenso em sistemas distribuídos, enfatizando sua importância teórica e prática na detecção de falhas.

O conceito de "Flexibilidade do Impact-FD", foca em sua habilidade para adaptar-se e manter a confiabilidade do sistema mesmo diante de diferentes configurações e possíveis falhas nos processos. Esta flexibilidade é alcançada ao permitir que o sistema aceite diversos conjuntos de processos ativos que, coletivamente, atendem a um critério de confiabilidade pré-estabelecido, denominado limiar (*threshold*). A ideia aqui é que o conjunto S, englobe todas as combinações possíveis de subconjuntos de processos que satisfazem esse limiar de confiabilidade.

Utilizando um procedimento que emprega a função `TPowerSet`, o *Impact-FD* gera todos os possíveis subconjuntos de processos $S^* = \{S_1, S_2, S_3, \ldots, S_m\}$, selecionando apenas aqueles cuja soma atende ou excede um limiar específico para cada subconjunto.

Para formalizar essa confiabilidade dinâmica e adaptativa do sistema, duas propriedades são definidas:

- **Propriedade de Limiar de Impacto**: Esta propriedade assegura que, em qualquer momento, o sistema identifica um conjunto de processos considerados confiáveis que cumprem com um mínimo estabelecido de confiabilidade, chamado de limiar. Isso significa que o sistema, através do *Impact-FD*, pode sempre contar com um grupo de processos que, juntos, garantem que o sistema como um todo seja confiável. O critério para um processo ser considerado confiável depende de um cálculo que envolve a soma de seus fatores de impacto, que deve atingir ou ultrapassar o limiar especificado, denominado - (***Threshold***).

- **Propriedade Eventual de Limiar de Impacto**: Essa propriedade é uma extensão da anterior, com um foco no comportamento do sistema ao longo do tempo. Ela garante que, após um certo ponto, o sistema alcança um estado estável em que o conjunto de processos confiáveis se mantém constante e satisfaz o limiar de confiabilidade de maneira contínua. Em outras palavras, mesmo que o sistema passe por instabilidades ou ajustes, eventualmente ele se estabiliza, garantindo permanentemente a confiabilidade conforme definido pelo limiar - denominado (***Threshold***).

No que diz respeito às conexões *timely* (oportunas), elas são fundamentais para diferenciar o comportamento do *Impact-FD* em sistemas (*Asynchronous System*) - AS e (*Weak Eventually Timely System*) - W-ET. As conexões *timely* garantem a entrega de mensagens dentro de limites de tempo bem definidos, o que é crucial para o ajuste fino da performance do *Impact-FD*, especialmente em sistemas W-ET onde algumas conexões eventualmente se tornam oportunas.

Nos experimentos realizados para avaliar o *Impact-FD*, foram considerados dois tipos de sistemas: AS (*Asynchronous System*) e W-ET (*Weak Eventually Timely System*). No sistema AS, todas as conexões são assíncronas e sujeitas a perdas, refletindo um ambiente distribuído onde não existem garantias sobre os tempos de transmissão de mensagens. Este cenário desafia o *Impact-FD* a detectar falhas de maneira eficaz, mesmo sem expectativas claras sobre a entrega de mensagens. Já no sistema W-ET, algumas conexões são eventualmente oportunas ($\Diamond$-timely), o que significa que, após um certo ponto desconhecido no tempo (eventualidade), essas conexões começam a garantir limites superiores no tempo de entrega das mensagens. Essa propriedade eventual permite uma detecção de falhas potencialmente mais rápida e confiável para algumas conexões, contrastando com as puramente assíncronas do sistema AS.

Analisando os trabalhos ***Disaster-FD*** e ***Impact-FD***, percebe-se que ambos abordam soluções para a detecção de falhas em sistemas distribuídos, com foco em ambientes propensos a desastres e na importância da confiabilidade do sistema.

O *Disaster-FD* é uma extensão do *Impact-FD*, que propõe um detector de falhas baseado na relevância dos processos e na confiança no sistema. O *Disaster-FD* aprimora a ideia introduzindo monitoramento em tempo real e federado, levando em consideração aspectos adicionais como o consumo de energia dos dispositivos com ênfase em ambientes IoT em cenários de desastres naturais, destacando a importância do monitoramento ativo e da avaliação da confiabilidade da rede.

O *Impact-FD*, por sua vez, evidencia a criação de um detector de falhas não confiável flexível que fornece um valor de nível de confiança no sistema, permitindo ao usuário ajustar a configuração de detecção de falhas de acordo com os requisitos da aplicação. Este detector leva em consideração a relevância de cada processo por meio de um valor de fator de impacto, além de uma margem de falhas aceitáveis, permitindo uma estratégia de monitoramento mais adaptável.

Ambos compartilham o objetivo de melhorar a detecção de falhas em sistemas distribuídos, mas o *Disaster-FD* expande o conceito para abordagens mais específicas de monitoramento em tempo real e adaptabilidade em cenários de desastres, enquanto o *Impact-FD* foca na flexibilidade e na capacidade de configuração do detector de falhas.

3.2 Medley: A Novel Distributed Failure Detector for IoT Networks

Neste estudo, propõe-se o *Medley* (YANG et al., 2019) que é um detector de falhas descentralizado, desenvolvido para atender às especificidades dos sistemas IoT distribuídos que operam em ambientes de redes ad-hoc sem fio. Neste cenário, a robustez e a resiliência da rede são de suma importância, visto que os sistemas são tipicamente compostos por uma vasta gama de dispositivos ou nós como sensores

que não só necessitam manter uma lista atualizada de membros ativos, mas também precisam realizar comunicações periódicas, para garantir a integridade da rede.

Observa-se que (YANG et al., 2019), considera o modelo de falha por parada (*fail-stop*), onde um nó falha ao cessar todas as operações e não se recupera. Este modelo simplifica o tratamento de falhas, focando na detecção de nós que param de funcionar, sem considerar falhas bizantinas ou maliciosas. Assume-se que a rede é assíncrona, permitindo atrasos e perdas de mensagens, e que os nós podem falhar simultaneamente, além de permitir a entrada e saída voluntária de nós do sistema.

O detector de falhas *Medley* utiliza o protocolo SWIM (*Scalable Weakly-consistent Infection-style Membership protocol*) destacando a eficiência e escalabilidade do protocolo em detectar falhas e disseminar informações sobre a composição do grupo em sistemas distribuídos. O protocolo divide suas operações em detecção de falhas e disseminação de informações, utilizando um método de *pinging* direto e indireto para verificar a disponibilidade dos nós e atualizar as listas de membros dos nós.

O *Medley* introduz o conceito de "Spatial Pinging" (Pinging Espacial) como uma inovação chave para a detecção de falhas em redes IoT. Esta abordagem ajusta a probabilidade de um nó escolher outro como alvo de *ping* baseando-se na proximidade espacial, preferindo nós mais próximos aos mais distantes, isso visa minimizar a latência e o tráfego de rede, otimizando assim a eficiência.

A técnica de seleção de alvo espacial emprega uma função que faz com que a probabilidade de um nó ser escolhido para receber um *ping* seja inversamente proporcional à distância. Isso é fundamentado na premissa de que *pings* para nós próximos requerem menos recursos de rede e têm maior probabilidade de sucesso em ambientes ad-hoc.

O detector de falhas *Medley* introduz um mecanismo de detecção de falhas com limitação temporal usando a abordagem em que os nós da rede serão verificados dentro de um intervalo de tempo pré-definido, com uma estratégia de seleção de alvos baseada em "passes". Nesta estratégia, os nós são escolhidos para receber *pings* com base em uma distribuição ponderada que considera tanto a proximidade quanto a frequência com que foram previamente verificados.

Ao escolher um subconjunto representativo de nós para enviar mensagens de *ping*, a probabilidade de seleção é inversamente proporcional à distância entre os nós, privilegiando os vizinhos mais próximos para verificações de status. Além disso, as funcionalidades do *Medley* no monitoramento atua da seguinte forma, um nó, identificado como M_i, escolhe aleatoriamente outro nó, M_j, da sua lista de membros para enviar uma mensagem de *ping*. A resposta esperada é um *ack* de M_j.

Caso M_i recebe esse *ack* dentro de um tempo limite predeterminado, baseado no *Round Trip Time* - (RTT) da mensagem, o processo é considerado satisfatório, e M_i não executa mais ações nesse período. Contudo, na ausência de um *ack*, M_i inicia um procedimento de *ping* indireto, selecionando k nós adicionais para tentar estabelecer comunicação com M_j. Este passo oferece uma segunda chance para nós que não responderam ao *ping* inicial, potencialmente devido a congestionamentos ou lentidão, e visa reduzir o risco de falsos positivos na identificação de falhas.

Caso M_i não receber nenhuma confirmação indireta de que M_j está vivo dentro de um determinado período, M_i marcará M_j como falho. Em seguida, M_i atualizará sua lista local de membros ativos para refletir a falha de M_j e iniciará o processo de disseminação dessa informação através da rede. A informação sobre a falha de M_j é disseminada por toda a rede IoT usando um mecanismo de propagação estilo *gossip*. Isso significa que as atualizações de estado são compartilhadas entre os nós através de suas comunicações regulares, garantindo que a informação da falha se espalhe rapidamente e eficientemente por toda a rede.

O ***Medley*** se concentra em reduzir a latência e o tráfego de rede ao favorecer nós próximos na seleção de alvos para *pings*, utilizando um mecanismo de seleção espacial. Adapta o protocolo SWIM para ambientes IoT, proporcionando detecção de falhas descentralizada e eficiente com baixo *overhead* de comunicação. Enquanto isso, ***Disaster-FD***, é projetado para ambientes propensos a desastres, en-

fatizando a monitoração ativa e a avaliação da confiabilidade da rede. Utiliza conceitos como limiar de confiabilidade, nível de confiança e fator de impacto para monitorar e avaliar a integridade da rede em tempo real, visando maximizar a resiliência em cenários de desastres.

3.3 Stab-FD: A Cooperative and Adaptive Failure Detector for Wide Area Networks

Stab-FD é um detector de falhas baseado no envio de mensagens de *"heartbeat"* (batimento cardíaco) para monitorar a disponibilidade de outros nós em uma rede. Ele se distingue pela sua capacidade de ajustar dinamicamente os temporizadores de detecção de falhas com base na estabilidade percebida dos links de entrada de cada nó, enquanto o $Stab^C$-FD é a versão cooperativa do *Stab-FD* que se destina a melhorar ainda mais a qualidade da detecção de falhas por meio da troca de visões de estabilidade dos links e listas de nós suspeitos entre os nós (SENS et al., 2024).

Ambos os detectores de falhas utilizam estratégias baseadas em *heartbeat* e focam na adaptação dinâmica para responder às condições variáveis da rede. A principal diferença reside na abordagem cooperativa do $Stab^C$-FD, que aproveita a inteligência coletiva ao compartilhar e comparar informações sobre a estabilidade dos links e suspeitas entre os nós, potencialmente reduzindo o tempo de detecção de falhas e aumentando a precisão.

Enquanto o *Stab-FD* se concentra na adaptação individual dos temporizadores com base na estabilidade dos links, o $Stab^C$-FD expande essa estratégia permitindo que os nós colaborem e se informem mutuamente sobre suspeitas de falhas, criando um sistema de detecção mais resiliente e adaptável a condições de rede em constante mudança. Essa abordagem cooperativa pode ser particularmente benéfica em redes amplas e distribuídas, onde a visibilidade limitada e as variações de latência podem desafiar os detectores de falhas tradicionais.

A margem de segurança é ajustada pela função ComputeMargin que tem o propósito de calcular a margem de segurança para que ela seja adaptativa às condições atuais da rede, especificamente às variações na qualidade dos links. Essa margem de segurança é usada para ajustar os temporizadores que determinam quanto tempo um nó espera por um heartbeat antes de considerar o emissor como suspeito de falha.

O ComputeMargin começa com a avaliação da estabilidade dos links de entrada para cada nó, que é representada por um vetor de estabilidade (stab). A estabilidade é uma medida de quão confiável é a comunicação entre os nós, baseada na frequência de *heartbeats* recebidos com sucesso em comparação com os perdidos ou atrasados. O algoritmo calcula o coeficiente de variação (C_v) da estabilidade dos links. O C_v é uma medida estatística que representa a relação entre o desvio padrão e a média, fornecendo uma noção de dispersão relativa dos valores de estabilidade.

Com base no C_v e na posição relativa da estabilidade do link em questão dentro do conjunto de todos os links (classificados em quartis), onde o valor percentual pode ser definido como 25, 50 ou 75, correspondendo, respectivamente, aos quartis Q1, Q2 e Q3. O `ComputeMargin` determina o ajuste necessário na margem de segurança, sendo que os links mais estáveis (aqueles com alta estabilidade relativa) terão suas margens de segurança reduzidas para permitir uma detecção de falhas mais rápida, enquanto links menos estáveis terão suas margens aumentadas para evitar falsos positivos. O ajuste específico da margem é feito de acordo com a distribuição quartil dos valores de estabilidade, onde links com estabilidade nos quartis inferiores (menos estáveis) recebem um aumento maior na margem de segurança e links nos quartis superiores (mais estáveis) recebem um ajuste negativo ou nenhum ajuste, promovendo uma resposta mais rápida.

A versão cooperativa, o $Stab^C$-FD, expande as funcionalidades do *Stab-FD* ao facilitar a troca de informações entre os nós sobre a estabilidade dos links e as listas de nós suspeitos. Essa estratégia cooperativa é projetada para permitir ajustes informados na margem de segurança usando os dados compartilhados, resultando em uma detecção de falhas mais rápida e precisa. Essa estratégia ilustra a importância da cooperação e do compartilhamento de informações em ambientes distribuídos, especialmente em redes de grande escala.

Durante os experimentos, foi constatado que períodos instáveis são comuns em rastreamentos de WAN, o que reforça a necessidade de detectores de falhas que possam se adaptar eficientemente a essas condições. A pesquisa concluiu que monitorar a estabilidade dos links para calibrar a margem de segurança é uma estratégia eficaz para melhorar a precisão dos detectores de falhas. Além disso, a abordagem cooperativa proposta pelo $Stab^C$-FD foi capaz de melhorar significativamente o tempo de detecção em situações onde a qualidade dos links é heterogênea, validando a eficácia da colaboração entre os nós.

Em suma, o *Stab-FD* e o $Stab^C$-FD representam avanços significativos na detecção de falhas em sistemas distribuídos, oferecendo uma solução eficaz para os desafios de manter a precisão na detecção de falhas em ambientes de rede dinâmicos. Através da adaptação dinâmica de temporizadores e margens de segurança, juntamente com a implementação de estratégias de cooperação, esses sistemas fornecem uma contribuição valiosa para a resiliência e eficiência de sistemas distribuídos frente às incertezas de rede.

O ***Disaster-FD*** é um detector de falhas projetado para ambientes propensos a desastres, focando no monitoramento em tempo real de redes IoT, enquanto o ***Stab-FD*** ajusta dinamicamente a margem de segurança com base na estabilidade dos links em sistemas distribuídos, especialmente útil em redes WAN, a abordagem do *Stab-FD* é ortogonal e poderia ser aplicada no *Disaster-FD*.

3.4 Querying on Federated Sensor Networks

O artigo Querying on Federated Sensor Networks,(CAN; DEMIRBAS, 2016), discute o desenvolvimento do protocolo *Layered Federated Sensor Network* - (L-FSN). Este protocolo foi projetado para otimizar a gestão e a consulta de dados em Redes de Sensores Federadas - (FSN), que integram múltiplas *Wireless Sensor Networks* - (WSNs) geograficamente distribuídas. Estas redes operam de forma autônoma, mas são interconectadas através de *Exterior Gateways* - (EGs) para formar uma estrutura federada, permitindo consultas e gestão de dados em larga escala.

Os EGs facilitam a comunicação entre diferentes "ilhas" de sensores, atuando como pontos de conexão e tradutores de protocolo, enquanto os *Interior Gateways* - (IGs) gerenciam a comunicação interna, otimizando a distribuição de dados e a eficiência energética.

A metodologia proposta no presente trabalho, abrange duas abordagens principais para a federação de redes de sensores, denominadas "*layered federation*" e "*flat federation*". Na abordagem de federação em camadas, cada "ilha" de rede de sensores pode ser gerenciada como um sistema autônomo e executar um protocolo de consulta específico para a ilha, permitindo uma gestão autônoma e otimizada para as características particulares de cada ilha. Este modelo contrasta com a gestão "flat", onde um único protocolo de consulta é aplicado uniformemente em toda a rede federada, sem distinções entre comunicação interna e externa às ilhas.

A arquitetura federada em camadas utiliza gateways exteriores - (EGs) e interiores (IGs) para conectar e gerenciar a comunicação entre as ilhas de sensores e dentro delas, respectivamente. Esse modelo promove um processamento de dados em rede e técnicas de indexação para armazenamento e recuperação de dados eficientes, permitindo que gateways e nós respondam diretamente às consultas quando possuem dados válidos ou encaminhem o pacote para a próxima unidade.

Os *Exterior Gateways* - (EGs), atuam como pontes entre diferentes *Wireless Sensor Networks*, permitindo a integração de redes independentes em uma estrutura federada. Eles são responsáveis por traduzir

e rotear consultas entre redes, facilitando a comunicação entre ilhas de sensores. Enquanto o *Interior Gateways* (IGs), focam na gestão interna da comunicação dentro de uma WSN específica, otimizando a distribuição de dados e melhorando a eficiência energética. Eles são essenciais para a implementação de políticas de gerenciamento de dados e energia adaptativas ao contexto de cada ilha de sensores.

O L-FSN é composto por três componentes principais: tabelas de roteamento, mecanismo de consulta entre ilhas (*inter-island querying*) e mecanismo de consulta dentro da ilha (*intra-island querying*). As tabelas de roteamento são fundamentais para o funcionamento do L-FSN, permitindo a comunicação eficaz entre as ilhas. No início da construção da rede, os gateways externos (EGs) trocam tabelas de roteamento e continuam a fazê-lo até que não haja mais informações novas a serem compartilhadas. Essa troca inicial ajuda a detectar vizinhos e estabelecer rotas de comunicação baseadas em políticas de seleção de caminho, como a seleção do caminho mais curto, utilizando o algoritmo de vetor de distância.

O mecanismo de consulta entre ilhas permite a comunicação de alto nível entre diferentes ilhas da FSN, utilizando os gateways como roteadores e tradutores de protocolo. Essa abordagem garante que os pacotes de consulta sejam direcionados corretamente para as ilhas de destino, levando em conta a privacidade e as políticas específicas de cada ilha. O protocolo permite que uma ilha envie consultas diretamente para uma ilha específica, adaptando-se a diversos cenários de aplicação e políticas de gestão. Para otimizar a eficiência e alcançar um sistema de consulta em larga escala, o L-FSN permite o uso de diferentes protocolos de consulta intra-ilha em cada ilha. Isso permite que os pacotes de consulta sigam caminhos específicos dentro de uma ilha, de acordo com o protocolo de consulta intra-ilha selecionado, que pode variar de acordo com as necessidades e características da ilha. O L-FSN destaca-se por sua flexibilidade e eficiência, permitindo a gestão autônoma de cada ilha com protocolos específicos de consulta intra-ilha, enquanto mantém uma comunicação eficiente entre ilhas. Esse design promove uma gestão de rede mais escalável, eficiente em termos de energia e com menor latência nas consultas, comparado a modelos de federação plana, onde um único protocolo de consulta é aplicado uniformemente em toda a rede federada.

A implementação do Protocolo *Layered Federated Sensor Network* - (L-FSN) começa com a inicialização das tabelas de roteamento, onde os *Exterior Gateways* (EGs) compartilham e atualizam suas tabelas para refletir os caminhos mais eficientes com base em algoritmos de vetor de distância. Esse processo é crucial para estabelecer rotas de comunicação eficazes entre as ilhas de sensores, facilitando o roteamento de pacotes com base em coordenadas geográficas e identificadores de destino. No que diz respeito à detecção de falhas e recuperação, o L-FSN emprega mecanismos robustos para garantir a resiliência da rede.

Quando um EG não recebe confirmação de um pacote enviado, inicia-se o processo de recuperação de falhas, onde caminhos alternativos são selecionados com base nas tabelas de roteamento. Informações sobre falhas são disseminadas através da rede, permitindo uma rápida adaptação e minimização de interrupções. O protocolo também detalha métodos para consultas entre ilhas (*inter-island querying*), onde *gateways* de nível superior formam uma estrutura em árvore que facilita o roteamento e a tradução de protocolos. Utilizando informações de fronteira, esses *gateways* direcionam pacotes de consulta para as áreas geográficas de destino, otimizando a eficiência da comunicação entre as ilhas. O protocolo estabelece uma comunicação robusta através da troca de tabelas de roteamento entre os *Exterior Gateways* (EGs), facilitando a detecção e a recuperação de falhas. Quando um EG envia um pacote, ele aguarda por uma mensagem de confirmação do recebimento. Se essa confirmação não for recebida, indica-se a possibilidade de uma falha, acionando o mecanismo de recuperação de falhas.

As falhas podem ser detectadas tanto nos EGs quanto nos *Interior Gateways* (IGs). No caso dos EGs, as informações de falha são propagadas por toda a Federação de Redes de Sensores (FSN) através da troca de tabelas de roteamento, permitindo que rotas alternativas sejam estabelecidas para manter a comunicação. Para os IGs, a informação de falha é disseminada internamente dentro da ilha usando

protocolos de roteamento intra-ilha, que são selecionados com base nas propriedades e necessidades específicas de cada ilha. Esses mecanismos garantem que a rede possa adaptar-se e recuperar-se de falhas, mantendo a integridade e a continuidade da coleta e do processamento de dados.

Analisando os dois trabalhos apesar das diferenças nos objetivos e metodologias específicas de cada sistema, existem semelhanças fundamentais entre o ***L-FSN*** e o ***Disaster-FD*** no que diz respeito ao monitoramento de sensores. O *L-FSN* e o *Disaster-FD* diferenciam-se em seus contextos de aplicação, porém compartilham um foco comum no monitoramento avançado. O primeiro otimiza o gerenciamento de redes de sensores federadas, melhorando consultas e a comunicação, enquanto o segundo se concentra na vigilância de redes IoT em áreas sujeitas a desastres, com o objetivo de identificar proativamente falhas. Ambos implementam estratégias de detecção de falhas essenciais para a continuidade operacional e a preservação da integridade dos dados. O *L-FSN* permite a comunicação entre ilhas geograficamente distribuídas e o *Disaster-FD* se adapta a ambientes suscetíveis a desastres, demonstrando a versatilidade de cada sistema, com monitoramento em tempo real tanto de seus próprios dispositivos quanto os dispositivos federados. Essas semelhanças ressaltam a importância do monitoramento avançado e da detecção de falhas em redes de sensores, independentemente do contexto específico de aplicação, sendo assim, ambos os sistemas contribuem para o avanço das redes de sensores, oferecendo *insights* valiosos para a construção de infraestruturas de monitoramento mais robustas, confiáveis e eficientes.

3.5 Um modelo de rede de sensores sem fio auto-organizada e tolerante a falhas para Detecção de Incêndios

Neste estudo (GIUNTINI, 2016), foi proposto um modelo de Redes de Sensores Sem Fio - (RSSF) com capacidades auto-organizáveis e tolerantes a falhas, projetado especificamente para a detecção de incêndios em áreas de conservação ambiental. O modelo se distingue por sua abordagem holística que integra mecanismos de auto-organização e tolerância a falhas, não somente na camada de roteamento, mas crucialmente também na camada de aplicação. Essa integração se dá através da utilização do protocolo *Data-Aggregation Aware Routing Protocol* - (DAARP), que otimiza o processo de roteamento e gestão de dados de maneira eficaz, representando um avanço significativo.

O modelo proposto é cuidadosamente estruturado em quatro fases principais, detalhadas a seguir, que delineiam desde a configuração inicial da rede até a gestão e análise de eventos de incêndio detectados:

- **Implementação e Estabelecimento da Rede**: Nesta primeira fase, ocorre a distribuição aleatória dos nós sensores na área de interesse, estabelecendo uma base sólida para a formação da rede. Um nó central, denominado nó *sink*, inicia a formação da árvore de roteamento através do envio de uma mensagem de controle em broadcast. Esta mensagem permite que cada nó calcule a distância até o nó *sink* em termos de saltos, garantindo uma cobertura de rede eficiente e organizada.

- **Notificação de Eventos e Gestão de Clusters**: Na segunda fase logo após a configuração inicial, a rede passa a monitorar ativamente a área designada para a detecção de possíveis sinais de incêndio, como um aumento significativo na temperatura. Ao identificar tais eventos, os nós se organizam em *clusters*, com um coordenador eleito para cada um, otimizando a coleta e análise de dados, minimizando redundâncias.

- **Detecção de Incêndios e Construção de Soluções Tolerantes a Falhas**: Na terceira fase a solução de detecção de incêndios é implementada em todos os sensores, mas é ativada apenas nos sensores que assumem a função de coordenador de *cluster*. Isso reduz o consumo de energia, pois a análise de dados e a detecção de incêndios ocorrem apenas quando e onde são necessárias. A detecção de incêndios é baseada na componentização de software, permitindo que diferentes

estratégias de detecção operem simultaneamente e de forma modular. Isso facilita a implementação de soluções robustas e tolerantes a falhas, uma vez que cada componente pode ser atualizado ou substituído independentemente, sem afetar o sistema como um todo. Cada uma dessas soluções é construída como um componente que pode ser carregado ou descarregado em tempo de execução pelo sensor coordenador. Isso permite uma análise diversificada dos dados coletados e aumenta a precisão da detecção de incêndios. Para determinar a presença de um incêndio, é utilizado um mecanismo de votação entre as diferentes soluções de detecção. Cada solução fornece um resultado (verdadeiro ou falso) junto com uma probabilidade de erro. Um consenso é alcançado por meio da votação, contando o número de resultados verdadeiros versus falsos e calculando a média das probabilidades de erro das soluções que indicaram o resultado majoritário. Isso permite ao sistema chegar a uma conclusão precisa sobre a ocorrência de um incêndio, minimizando o risco de falsos alarmes.

- **Comunicação de Dados e Economia de Energia**: Finalmente na quarta fase, após a confirmação de um incêndio através do consenso, o coordenador decide sobre a transmissão dos dados ao nó *sink*. Esta fase permite uma gestão inteligente dos recursos, onde o coordenador pode optar por enviar apenas resultados positivos, visando economizar energia, ou encaminhar todos os resultados para análises mais detalhadas, além de agregar dados semelhantes para reduzir duplicidades nas transmissões. Para avaliar a eficácia do modelo proposto, especialmente em termos de tolerância a falhas e capacidade de auto-organização, (GIUNTINI, 2016) utilizou o simulador Sinalgo, que é um *framework* destinado à simulação de redes algorítmicas, incluindo RSSFs.

O simulador foi configurado para replicar as condições de uma área de conservação ambiental, similar ao Jardim Botânico de Brasília, incluindo a distribuição espacial dos sensores e as condições ambientais variáveis, além de simular a operação completa da RSSF, desde a formação de *clusters* até a detecção de incêndios e gestão de falhas, permitindo a avaliação da eficácia do modelo em diferentes cenários de falha, para finalizar a simulação foram coletadas métricas de desempenho, como a precisão na detecção de incêndios, a eficiência na comunicação e o impacto das estratégias de tolerância a falhas na resiliência da rede, essas simulações utilizaram árvores de decisão, para o desenvolvimento das árvores de decisão foi utilizado o *Forest Fires Data Set* com 517 instancias do *UCI - Machine Learning Repository*.

Na comparação entre o modelo de Rede de Sensores Sem Fio auto-organizada e tolerante a falhas para Detecção de Incêndios, proposto por (GIUNTINI, 2016), e o detector de falhas *Disaster-FD*, ambos focam na aplicação de tecnologias para o monitoramento de ambientes sujeitos a desastres. Contudo, as abordagens adotadas por cada trabalho refletem diferentes perspectivas e estratégias na detecção de falhas e no gerenciamento de desastres naturais utilizando a Internet das Coisas (IoT) e Redes de Sensores Sem Fio (RSSF).

O modelo adotado por (GIUNTINI, 2016), destaca-se por sua especificidade na detecção de incêndios em áreas de conservação ambiental, utilizando um sistema RSSF auto-organizável e tolerante a falhas. Esta abordagem foca na eficiência energética e na precisão da detecção, empregando mecanismos de auto-organização, tolerância a falhas e estratégias de votação para a agregação de dados. A implementação desse modelo é direcionada para otimizar o processo de detecção de incêndios, minimizando falsos positivos e garantindo uma resposta rápida e eficaz na preservação de áreas ambientais.

Por outro lado, o *Disaster-FD* propõe um sistema de detecção de falhas para redes IoT em ambientes suscetíveis a diversos tipos de desastres. Este trabalho incorpora conceitos como limiar de confiabilidade, nível de confiança e fator de impacto, oferecendo uma abordagem mais abrangente e flexível na avaliação da rede. A capacidade de monitoramento federado é um dos aspectos inovadores do *Disaster-FD*, permitindo uma colaboração efetiva entre múltiplas entidades para uma detecção de falhas mais robusta e uma cobertura ampliada do monitoramento.

Enquanto o modelo de (GIUNTINI, 2016) oferece uma solução especializada na detecção de incêndios, refletindo um caso de uso específico com impacto significativo na conservação ambiental, o *Disaster-FD* destaca-se pela sua aplicabilidade mais ampla em ambientes IoT, introduzindo um mecanismo de detecção de falhas adaptável a diferentes cenários de desastres.

Capítulo 4

Proposta

O Capítulo 4 apresenta o *Disaster-FD*, um sistema de detecção de falhas desenvolvido especificamente para ambientes sujeitos a desastres. Utilizando a tecnologia da Internet das Coisas (IoT), este sistema proporciona um monitoramento abrangente em múltiplas regiões. Sua inovação reside na sua capacidade de realizar monitoramento eficiente tanto internamente quanto entre regiões, empregando os protocolos CoAP e ICMP para garantir uma resposta rápida e em tempo real diante da iminência de desastres. Com uma arquitetura que permite múltiplos processos, o *Disaster-FD* amplia significativamente a resiliência e a tolerância a desastres dos ambientes monitorados. Esta abordagem descentralizada não só melhora a eficácia na detecção de falhas, mas também realiza uma gestão de riscos efetiva essencial para mitigar os efeitos adversos dos desastres.

4.1 Disaster-FD

O trabalho proposto estende a definição de detectores de falha, com a adoção e refinamento da abordagem inicialmente delineada por (ROSSETTO et al., 2018).

Nesse contexto, o *Impact-FD*, introduzido por (ROSSETTO et al., 2018) é um detector de falhas inovador e flexível que atribui um fator de impacto a cada processo dentro de um sistema, refletindo sua relevância e contribuição para o nível de confiança geral do sistema. Ao contrário dos detectores de falhas tradicionais (FDs), que operam em um modelo binário, marcando processos como "confiáveis" ou "suspeitos", o *Impact-FD* oferece uma abordagem diversificada ao permitir a configuração da sensibilidade da detecção de falhas com base na importância relativa dos processos para caracterização do sistema como confiável ou não.

Isso significa que a falha de processos de alto impacto afeta significativamente o nível de confiança do sistema, enquanto falhas de processos menos críticos podem ser toleradas. A flexibilidade do *Impact-FD* permite que ele se adapte a vários requisitos e cenários de sistema, incluindo aqueles com processos redundantes ou nós heterogêneos, estabelecendo limiares apropriados para margens de falha aceitáveis.

Propriedades do Impact-FD:

- **Fator de Impacto e Subconjuntos**: Cada processo é avaliado com base em um fator de impacto que indica sua relevância. Isso permite uma diferenciação mais granular na detecção de falhas, considerando o peso de cada processo no funcionamento do sistema.

- **Nível de Confiança**: O *Impact-FD* calcula um nível de confiança para o sistema, levando em conta os fatores de impacto dos processos. Isso se difere dos detectores tradicionais que não agregam essa visão holística da confiabilidade baseada na importância relativa dos processos.

- **Margem de Falhas**: Este conceito permite ao sistema tolerar certas falhas sem que sua confiabilidade geral seja considerada comprometida. A flexibilidade em configurar essas margens permite adaptar a tolerância a falhas às necessidades específicas de cada aplicação ou cenário.

- **Flexibilidade**: O *Impact-FD* é altamente configurável, permitindo ajustes na sensibilidade da detecção de falhas conforme o impacto dos processos. Isso oferece uma ferramenta poderosa para gerenciar a confiabilidade em sistemas complexos e distribuídos.

A principal diferença do *Impact-FD* em relação aos detectores de falhas tradicionais reside na sua abordagem baseada em impacto e confiança, que oferece uma visão mais diversificada e ajustável da confiabilidade do sistema. Em vez de tratar todos os processos igualmente, o *Impact-FD* reconhece que alguns processos são mais críticos para a operação do sistema do que outros e ajusta sua detecção de falhas de acordo. Isso permite uma detecção de falhas mais eficaz e eficiente em ambientes distribuídos complexos, onde a importância dos processos pode variar significativamente.

Neste contexto, existe um processo $p \in \Pi$ que monitora um subconjunto S de Π. Cada processo em S conecta-se a p por meio de um canal de comunicação. Deste modo, diferentemente de detectores tradicionais definidos em (CHANDRA; TOUEG, 1996), o *Disaster-FD* pode ser definido como um detector de falhas não confiável que oferece uma saída relacionada ao **nível de confiança** nos processos em S. Caso o nível de confiança seja igual ou superior a um **limiar** definido pelo usuário, o sistema é considerado confiável. Em outras palavras, quando *Disaster-FD* é invocado no processo p, ele retorna o nível de confiança de p nos processos em S.

4.1.1 Definições

Podemos considerar um sistema distribuído composto de um conjunto finito de processos $\Pi = \{q_1, \ldots, q_n\}$, em que $|\Pi| = n, (n \geq 2)$, e a existência de apenas um processo por nó ou sensor. Em tal sistema, cada processo opera independentemente em um nó (ou processo). Cada processo possui um identificador único ($id \mid 1 \leq id \leq n$), que varia sequencialmente de 1 até n.

Os processos podem apresentar falhas por meio de travamentos, sem possibilidade de recuperação. Um processo é considerado correto se permanecer operante durante toda a execução do sistema. O conceito de tempo global, representado por T, é utilizado para monitorar o sistema. A função de padrão de falhas, F(t), é uma representação temporal de falhas, indicando quais processos falharam até um certo ponto no tempo global T.

Ao analisar o histórico de falhas através de F(t), o sistema pode determinar quais processos são "corretos" (nunca falharam de acordo com F) e quais são "falhos" (aparecem no padrão de falhas F).

Cada processo pode monitorar um subconjunto S de processos, estabelecendo links de comunicação para a troca de mensagens.

O conceito de *Disaster-FD* refere-se a um detector de falhas não confiável, que avalia o nível de confiança em um conjunto de processos. Se o nível de confiança calculado supera um limiar definido pelo usuário, o conjunto de processos é considerado confiável. Para um processo p monitorando um conjunto S, o *Disaster-FD* retorna uma medida de confiança, ***Trust Level***, $TL_p^S(t)$ em relação a confiabilidade desse conjunto S.

Além disso, as funções correto(T_S) e falho(F_S) são usadas para ajustar o conjunto de processos "corretos" e "falhos" considerando apenas aqueles dentro do subconjunto S que está sendo monitorado pelo processo p. Isso permite uma visão mais granular e específica da integridade dos processos dentro de um escopo definido.

4.1.2 Federação

A Figura 10 ilustra o monitoramento de múltiplos processos monitores, cada um implantado em uma região monitorada. A Federação no contexto do *Disaster-FD*, refere-se à capacidade de integrar e coordenar sistemas de monitoramento distribuídos em várias regiões, cada um operando de forma independente, mas colaborando para uma visão unificada e abrangente do estado da rede. Neste cenário, cada região conta com um processo monitor e um conjunto de dispositivos IoT. O monitor de cada região monitora os dispositivos de sua própria região e um subconjunto dos monitores e dispositivos em outras regiões.

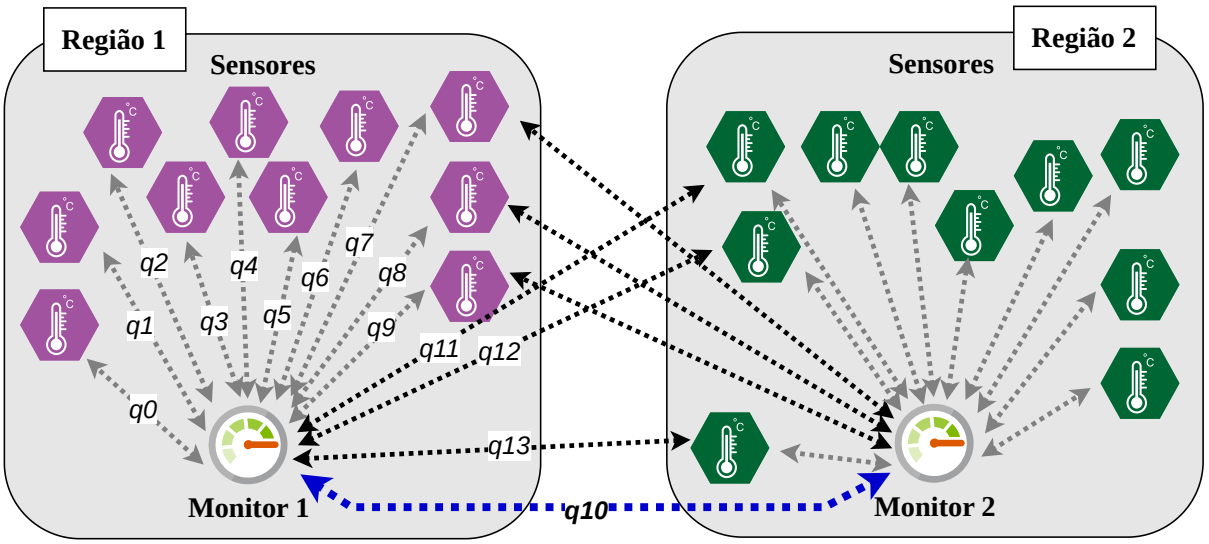

Figura 10 – Exemplo de um cenário de monitoramento.

Isso permite que o sistema não apenas detecte falhas dentro de uma região específica, mas também aproveite informações de outras regiões para melhorar a precisão da detecção e a resposta a desastres. Ao compartilhar dados entre regiões federadas, o sistema pode responder de forma mais eficaz a eventos adversos, melhorando a resiliência e a continuidade das operações em ambientes propensos a desastres. Tal arranjo permite incrementar a tolerância a desastres, impedindo que a falha completa de uma região passe desapercebida.

Neste contexto, cada processo q dentro do conjunto $S \subset \Pi$ recebe um fator de impacto atribuído. Esse fator, um valor real positivo, reflete a importância relativa do processo dentro do sistema. No cenário da Figura 10, por exemplo, pode-se atribuir um fator de impacto maior aos monitores em comparação ao fator de impacto dos sensores, expressando que a falha de um monitor tem um peso maior que a falha de um sensor.

Cada processo monitor p invoca o *Disaster-FD* e realiza o cálculo do nível de confiança para o conjunto S, estabelecendo assim a confiabilidade geral do sistema. Esse nível de confiança é determinado pela soma dos fatores de impacto dos processos que, naquele instante, não são considerados falhos.

4.1.3 Formalização

O *Disaster-FD* mantém a consistência estrutural apresentada no *Impact-FD* (ROSSETTO et al., 2018), ao mesmo tempo em que amplia essa metodologia para contextos onde o monitoramento é realizado de maneira múltipla ou em regime de cooperação. Essa extensão aprimora o desempenho, a acurácia e a flexibilidade do sistema, atributos essenciais para uma resposta efetiva às incertezas e mudanças imprevisíveis presentes em situações de emergência causadas por desastres.

O **Fator de Impacto** atribuído a cada processo corresponde a um valor inteiro positivo que indica sua importância relativa no sistema. O fator de impacto de cada processo monitorado i, (I_i), juntamente com o identificador único do processo, compõem o conjunto $S \subset \Pi$ monitorado. Deste modo, os valores no conjunto S correspondem ao conjunto $\{\langle id_1, I_1\rangle, \langle id_2, I_2\rangle, \ldots, \langle id_k, I_k\rangle\}$ para cada processo $i \in S, 1 \leq i \leq k$.

Para cada conjunto S monitorado, o subconjunto $T_p^S(t)$ representa os processos que não são suspeitos pelo monitor p no instante t. Complementarmente, o conjunto $F_p^S(t)$ representa os processos considerados faltosos pelo monitor p no instante t.

O **Nível de Confiança**, ou ***Trust Level***, indica o nível de confiança do processo monitor p no conjunto de processos em S em um determinado instante, calculado por $TL_p^S(t)$. Representa a soma dos fatores de impacto dos processos não falhos, ou seja, $TL_p^S(t) = \sum_i (I_i), \forall i \in T_p^S(t)$

Cada monitor pode acompanhar subconjuntos de diversos processos, com diferentes níveis de confiança e fatores de impacto individuais. O conjunto S^* abrange os m subconjuntos únicos monitorados, indicado por $S^* = \{S_1, S_2, S_3, \ldots, S_m\}$.

Por fim, o **Limiar**, ou ***Threshold***, define o limite mínimo de confiabilidade para cada conjunto em S^*, matematicamente representado por $\{Th_1, Th_2, \ldots, Th_m\}$, no qual cada Th_i está relacionado ao nível mínimo de confiança necessário para um subconjunto de processos S_i.

O Th_S é usado pelo monitor para verificar a confiança nos processos dos subconjuntos em S^*. Se, para cada um dos m subconjuntos de $S^*, 1 \leq i \leq m$, o $TL_p^i(t) > Th_i$, então S^* é considerado confiável (*trusted*) no tempo t por p; caso contrário, S^* é considerado não confiável (*not trusted*).

Essa classe de algoritmo introduz o conceito de **Propriedade de Flexibilidade**, que denota a capacidade do detector de falhas de tolerar uma certa margem de falhas ou falsas suspeitas, ou seja, a sua capacidade de considerar diferentes conjuntos de respostas que levam o sistema a estados de confiança.

Tabela 2 – Conjunto S_1 com processos q_i e seus respectivos fatores de impacto.

Conjunto S_1 monitorado pelo processo monitor da Região 1
$\langle q_0, 10\rangle, \langle q_1, 10\rangle, \langle q_2, 10\rangle, \langle q_3, 10\rangle, \langle q_4, 10\rangle, \langle q_5, 10\rangle, \langle q_6, 10\rangle,$ $\langle q_7, 10\rangle, \langle q_8, 10\rangle, \langle q_9, 10\rangle, \langle q_{10}, 60\rangle, \langle q_{11}, 20\rangle, \langle q_{12}, 20\rangle, \langle q_{13}, 20\rangle$

A Tabela 2 apresenta um exemplo de conjunto de processos monitorados semelhante ao cenário da Figura 10. O conjunto S_1 do monitor da Região 1 compreende os processos q_0 a q_9 representando os sensores localizados na própria Região 1 (roxo), os processos q_{11} a q_{13} representando sensores remotos em monitoramento (verde), e q_{10} como o monitor da Região 2. O valor máximo de $TL_p^{S_1}(t)$ é 220, $\forall t > 0$, sendo 100 para os sensores locais e 120 para os sensores e monitor remotos. Nesta situação, o limiar escolhido deve refletir o objetivo do monitor. Por exemplo, para garantir que ao menos um processo de cada região sempre responda, tem-se $120 < Th_1 \leq 220$ e $TL_p^{S_1}(t) > Th_1, \forall t > 0$. O monitor da Região 2 pode adotar uma estratégia equivalente.

4.1.4 Estimativa de Chegada de Heartbeats - EA_{k+1}

No contexto do *Disaster-FD*, a aplicação da fórmula para calcular a estimativa de chegada (EA - *Estimated Arrival*) de *heartbeats* - (HB) é adaptada para melhorar a detecção de falhas em ambientes propensos a desastres, especificamente em redes IoT. Essa abordagem se baseia no histórico de tempos de chegada dos *heartbeats* anteriores e incorpora uma margem de segurança (β) para calcular o tempo estimado de chegada do próximo *heartbeat* (EA_{k+1}).

O cálculo de EA_{k+1}, que é a estimativa para a chegada do próximo *heartbeat*, utiliza uma abordagem que leva em conta a janela deslizante dos ω *heartbeats* mais recentes recebidos de um processo q

representadas por $m_1, m_2, ..., m_w$. Os valores $T_1, T_2, ..., T_w$ são os respectivos tempos de recepção dessas mensagens, de acordo com o relógio local de p.

Assim, conforme definido em (CHEN; TOUEG; AGUILERA, 2002), temos:

$$EA_{k+1} = \frac{1}{w} \sum_{i=k-w}^{k} (T_i - \Delta_i \times i) + (k+1) \times \Delta_i \quad (2)$$

Em que Δ_i corresponde ao intervalo de envio de dois *heartbeats* consecutivos.

4.1.4.1 Margem de Segurança e Definição do Timeout

A adição de uma margem de segurança constante (β) ao calcular o próximo *timeout* ($\tau(k+1)$) ajuda a acomodar incertezas na rede, reduzindo falsas suspeitas sem comprometer a QoS. Este método permite um ajuste fino dos parâmetros do detector de falhas com base no comportamento observado, melhorando a precisão na detecção de falhas e a adaptabilidade do sistema.

Deste modo, o tempo esperado de chegada do *heartbeat* $k+1$, denominado τ_{k+1}, é definido como $\tau_{k+1} = EA_{k+1} + \beta$, e o não recebimento de um *heartbeat* até o tempo τ_{k+1} caracteriza o processo q como suspeito.

4.1.5 Diferença do Impact-FD

O detector de falhas *Disaster-FD* foi projetado para ambientes sujeitos a desastres, com foco em monitoramento ativo e constante, adaptando-se a situações onde a infraestrutura de comunicação pode falhar. Emprega monitoramento federado para melhorar a tolerância a desastres. Por outro lado, *Impact-FD* analisa o nível de confiança da rede a partir de uma única leitura de um arquivo contendo o trace da rede, sem realizar monitoramento em tempo real, refletindo uma abordagem menos reativa e mais baseada em análise pontual.

Embora tanto o *Disaster-FD* quando o *Impact-FD* utilizem a equação da Seção 4.1.4 para calcular o tempo estimado de chegada do próximo *heartbeat*, denominado EA_{k+1}, as duas proposta utilizam abordagens ligeiramente distintas na interpretação e implementação da equação.

A implementação do protocolo *Disaster-FD* utiliza o identificador de número de sequência da mensagem de *heartbeat* para calcular a diferença entre o tempo real de chegada e um tempo de chegada teórico, o qual definem como produto do identificador pelo intervalo fixo entre *heartbeats* consecutivos (Δ_i). O protocolo *Impact-FD*, por sua vez, não utiliza diretamente o número de sequência do *heartbeat*, empregando um índice incremental para calcular a diferença entre o tempo de chegada de cada *heartbeat* e o tempo de chegada esperado, baseado no Δ_i.

A metodologia proposta por Chen baseia-se no ajuste da estimativa de tempo de chegada usando o histórico dos tempos de chegada das w mensagens anteriores, considerando a diferença entre os tempos de chegada reais e os esperados, com base no intervalo regular entre os *heartbeats*.

Sendo assim, o *Disaster-FD* está mais alinhado com a teoria de Chen, pois ele incorpora diretamente o conceito de sequencialidade dos *heartbeats* (através do número de sequência), refletindo a abordagem de Chen de ajustar as estimativas com base nas diferenças entre os tempos de chegada reais e os esperados. Já o *Impact-FD*, embora similar em estrutura, não captura a sequencialidade de forma tão direta, utilizando um índice incremental que pode não representar com precisão a sequência real dos *heartbeats*.

Na prática, conforme observado ao analisar os registros de experimentos do *Impact-FD*, isto significa que o *Impact-FD* tem mais dificuldade em lidar com "buracos" na sequência de *heartbeats*, o que ocorre quando algumas mensagens são perdidas.

O efeito disso é que a implementação do *Impact-FD* necessita de um tempo superior para detectar um falso positivo, ou seja, perceber que suspeitou erroneamente de um processo correto.

4.2 Implementação

O *Disaster-FD*, implementado em Java, foi desenvolvido nessa linguagem para garantir compatibilidade e integração eficiente com o *Impact-FD* (ROSSETTO et al., 2018), que também foi criado em Java. Essa escolha permite reutilizar componentes e bibliotecas, facilitando a manutenção e expansão do sistema. A implementação utiliza a biblioteca Californium (**??**) para suportar o protocolo CoAP, essencial para o monitoramento de dispositivos IoT, e está disponível online, com o código-fonte acessível no GitHub [1].

O *Disaster-FD* se destaca por sua abordagem de monitoramento federado e multi-protocolo, ele emprega requisições CoAP para monitorar dispositivos IoT e o protocolo ICMP para supervisionar nós monitores de regiões federadas vizinhas. Essa abordagem dual proporciona flexibilidade e uma análise mais abrangente do estado da rede.

4.2.1 Visão Geral da Arquitetura

A arquitetura do sistema *Disaster-FD* é composto por vários módulos fundamentais que interagem entre si para monitorar dispositivos, detectar falhas e assegurar a confiabilidade da rede, o diagrama de classes apresentado na figura 11 ilustra a arquitetura geral do *Disaster-FD* e as relações entre seus módulo, com ênfase nas associações de composição que destacam a relação estreita entre os módulos.

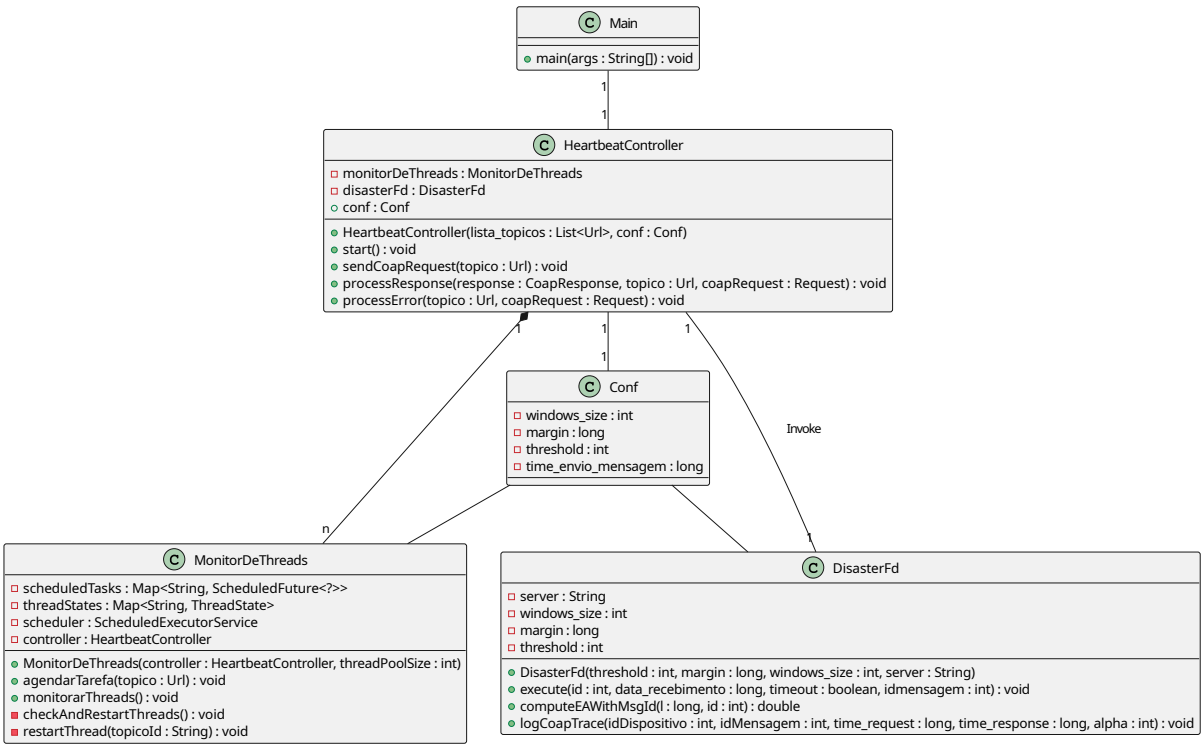

Figura 11 – Visão Geral da Arquitetura Disaster-FD.

O módulo `Main` serve como ponto de entrada para o sistema, inicializando o controlador principal (`HeartbeatController`). O módulo `HeartbeatController` é o núcleo do sistema de monitoramento,

[1] Código Disponível em https://github.com/abadiopaulo/disasterFd.git

gerenciando a comunicação CoAP ou ICMP com os dispositivos, enviando requisições e processando respostas, coordenando as tarefas de monitoramento e detecção de falhas.

Este módulo instancia e gerencia o módulo `MonitorDeThreads`, o módulo `DisasterFd`, que implementa a lógica do detector de falhas e o módulo `Conf`, responsável por armazenar as configurações globais e compartilhadas da execução do algoritmo no processo monitor.

O módulo `HeartbeatController` é essencial no sistema de monitoramento, responsável por iniciar e gerenciar o monitoramento da rede. Ele cuida da comunicação com os dispositivos usando os protocolos CoAP e ICMP, processa as respostas recebidas e trata erros de comunicação, assegurando a robustez e a continuidade do monitoramento. Suas funcionalidades centrais coordenam as operações críticas para detectar falhas e manter a confiabilidade da rede.

O módulo `MonitorDeThreads` gerencia e monitora as *threads* que enviam requisições periódicas aos dispositivos, assegurando que estejam funcionando corretamente e reiniciando aquelas que falham. Ele é responsável por agendar e monitorar tarefas, garantindo que todas as *threads* operem de maneira eficiente e contínua.

Além de supervisionar o estado das *threads*, este módulo verifica e reinicia as *threads* conforme necessário, assegurando a estabilidade e a continuidade das operações de monitoramento na rede.

O módulo `Conf` armazena todas as configurações necessárias para o funcionamento do sistema, incluindo parâmetros essenciais para a operação eficaz e segura. Ele garante que todas as partes do sistema tenham acesso aos parâmetros necessários, como tamanho da janela de monitoramento, margens de segurança, limiares de detecção e intervalos de envio de mensagens. Este módulo é fundamental para assegurar que todas as configurações sejam centralizadas e acessíveis, permitindo que o sistema funcione de maneira consistente e eficiente.

O módulo `DisasterFd` implementa a lógica de detecção de falhas, monitorando os dispositivos, calculando tempos de detecção e verificando a confiabilidade da rede. Ele é responsável por assegurar que o sistema identifique falhas de maneira eficiente e atualize os estados dos dispositivos conforme necessário. Além disso, o módulo calcula o tempo estimado de chegada dos sinais de vida (*heartbeats*) e registra as comunicações CoAP e ICMP, garantindo um monitoramento contínuo e detalhado da rede.

As associações de composição entre esses módulo são bem definidas, destacando a relação estreita e dependente entre eles. O módulo `Main` inicializa e instancia o `HeartbeatController`, que por sua vez compõe as instâncias de `MonitorDeThreads`, `DisasterFd` e `Conf`.

O `HeartbeatController` depende diretamente de `MonitorDeThreads` para gerenciar as *threads* de monitoramento. Essa relação de composição indica que `MonitorDeThreads` não pode existir sem o `HeartbeatController`, pois é o `HeartbeatController` que o inicializa e controla seu ciclo de vida. Da mesma forma, `HeartbeatController` compõe `DisasterFd`, indicando que a lógica de detecção de falhas é uma parte integral do controlador e não pode funcionar de forma independente.

O módulo `Conf` é utilizada por `HeartbeatController`, `MonitorDeThreads` e `DisasterFd`, fornecendo os parâmetros de configuração necessários para o funcionamento do sistema. Essas associações indicam que a configuração do sistema é crucial e deve ser consistentemente acessível por todas as partes do sistema.

A arquitetura do `Disaster-FD` é estruturada, com cada módulo desempenhando um papel específico no monitoramento e detecção de falhas na rede. O `HeartbeatController` atua como o controlador principal, coordenando as atividades de monitoramento e interagindo com os módulos `MonitorDeThreads` e `DisasterFd` para assegurar a confiabilidade da rede. O módulo `Conf` centraliza as configurações do sistema, garantindo que todas as partes do sistema tenham acesso aos parâmetros necessários para funcionar corretamente.

4.2.2 Sistema de Gerenciamento de Threads

O *Disaster-FD* usa uma arquitetura que segue o modelo de concorrência muitos-para-muitos, onde tarefas são mapeadas para um conjunto de *threads* gerenciadas por um `ScheduledExecutorService`, que é parte do *framework* de concorrência Java, fornecido no pacote *java.util.concurrent*. Este serviço é configurado com um pool de *threads*, cujo tamanho é ajustável, permitindo que o sistema escale conforme a demanda.

Cada tarefa corresponde ao envio de uma requisição CoAP a um tópico específico, sendo identificada por um identificador único (ID do tópico) e gerenciada individualmente. Este modelo é caracterizado pelo agendamento e execução de tarefas em *threads* separadas, que podem ser executadas uma única vez ou repetidamente com intervalos fixos.

O controle de estado para cada *thread* agendada é realizado através do (`ThreadState`), o que permite monitorar a atividade da *thread* (por exemplo, a última vez que foi executada) e decidir se uma *thread* precisa ser reiniciada. Isso é crucial para sistemas que necessitam de alta disponibilidade e confiabilidade, garantindo que as tarefas continuem sendo executadas mesmo em caso de falhas.

4.2.2.1 ScheduledExecutorService, ScheduledFuture<?> e ThreadState

A escolha do `ScheduledExecutorService` *(interface que estende ExecutorService)*, permite a criação de um *pool* de *threads*. Este serviço facilita a execução de múltiplas tarefas de maneira concorrente, sem sobrecarregar o sistema com a criação excessiva de *threads*, alinhando-se ao modelo de concorrência muitos-para-muitos. Ele oferece métodos como `scheduleAtFixedRate` para o agendamento de tarefas que precisam ser executadas repetidamente em intervalos específicos.

É importante ressaltar que o emprego de `ScheduledFuture` pertencente ao pacote *java.util.concurrent* representa um componente crítico no modelo de gerenciamento de *threads* adotado, o `ScheduledFuture<?>` é uma interface que representa o resultado de uma tarefa assíncrona e pode ser usada para controlar ou consultar o estado dessa tarefa após o seu agendamento. Essa interface é especialmente relevante em contextos onde tarefas são agendadas para execução futura, permitindo ações como cancelamento, verificação de conclusão e obtenção de resultados da execução.

O sistema implementa um mecanismo de monitoramento de *threads* através do mapa `threadStates`, que registra o estado de cada *thread* (ou tarefa) agendada. Este componente é vital para a manutenção da integridade e da confiabilidade do sistema.

A subinterface `ScheduledExecutorService` é especializada no agendamento de tarefas que devem ser executadas após um período de tempo determinado ou de forma periódica. Essa funcionalidade é particularmente importante em ambientes IoT, onde muitas operações, como o envio de requisições CoAP, precisam ser realizadas em intervalos regulares. A interface permite um controle preciso sobre a execução de tarefas, incluindo a capacidade de executá-las em intervalos fixos ou após um atraso específico.

A interface `ScheduledFuture<?>`, por sua vez, é uma interface que estende `Future<?>` e representa o resultado de uma tarefa assíncrona agendada. Ela fornece métodos para o controle e monitoramento dessas tarefas, como verificar a conclusão, cancelar a execução e obter resultados. Além disso, permite consultar o tempo restante até a próxima execução da tarefa, oferecendo um controle detalhado sobre o agendamento e a execução em sistemas de alta complexidade.

`ThreadState` é uma enumeração que representa os diferentes estados de uma *thread* ao longo de seu ciclo de vida, incluindo estados como *NEW, RUNNABLE, BLOCKED, WAITING e TERMINATED.* O monitoramento do estado de *threads* é crucial para diagnóstico, controle de execução e otimização do uso de recursos em aplicações concorrentes.

4.2.2.2 Estrutura do Gerenciador de Threads no Sistema Disaster-FD

A arquitetura do *Disaster-FD* é sustentada por uma série de classes e interfaces que facilitam o gerenciamento eficiente de *threads* e tarefas assíncronas, especialmente no que diz respeito ao envio de requisições CoAP.

A Figura 12 ilustra o Diagrama de Classe do `MonitorDeThreads`, que é uma representação da estrutura de classes projetada para o gerenciamento eficiente de *threads* e tarefas assíncronas no sistema *Disaster-FD*, com foco especial no envio de requisições CoAP.

O diagrama de classes reflete a seguinte estrutura:

- **HeartbeatController**: Esta classe centraliza as operações relacionadas ao protocolo CoAP, ela é encarregada de iniciar o sistema, enviar requisições CoAP e processar tanto as respostas quanto os erros resultantes. A classe também fornece métodos para a gestão do tempo e recuperação de tópicos por seus identificadores.
- **MonitorDeThreads**: Atua como um sistema de vigilância para as *threads* em execução. Possui métodos para agendar a verificação das *threads*, monitorá-las e, se necessário, reiniciar *threads* individuais. Esta classe é crucial para garantir que as tarefas do sistema permaneçam operacionais.
- **ScheduledExecutorService**: Uma interface que o MonitorDeThreads implementa para agendar tarefas com precisão. Ela é parte integrante do mecanismo que permite ao sistema executar tarefas assíncronas de forma programada.
- **ScheduledFuture<?>**: Interface que representa o resultado futuro de tarefas assíncronas, fornecendo controle sobre a execução dessas tarefas após serem agendadas.
- **ThreadState**: Uma classe que registra o estado das *threads*, mantendo informações sobre a última execução de cada *thread* e permitindo um controle de estado detalhado para o sistema.

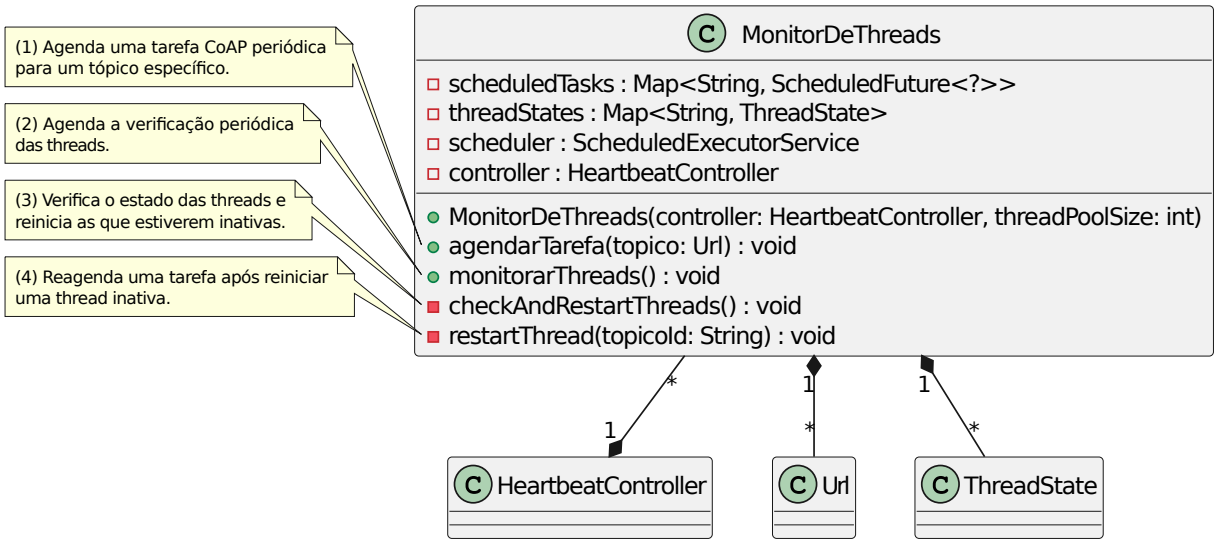

Figura 12 – Diagrama de Classe MonitorDeThreads.

4.2.2.3 Monitor(ICMP) e Dispositivo(CoAP)

O *Disaster-FD* abrangendo as regiões federadas de Grenoble e Strasbourg, permite a gestão e monitoramento eficazes de dispositivos IoT, designados como “nós”, distribuídos por essas regiões.

As solicitações CoAP do tipo "CON" (*Confirmable*) aumentam a confiabilidade na comunicação, pois cada mensagem prevê uma resposta do dispositivo receptor. Além disso, o sistema implementa um mecanismo de *timeout* para estas solicitações, assegurando que o monitoramento continue eficiente mesmo quando um dispositivo não responde dentro do tempo esperado.

A escolha do protocolo CoAP para comunicação com os dispositivos é motivada por sua eficiência em ambientes com recursos limitados, característicos das redes IoT. Adicionalmente, o sistema emprega um método de monitoramento dual, utilizando o protocolo ICMP para verificar a disponibilidade entre monitores de regiões federadas vizinhas. Este mecanismo de segurança adicional melhora a detecção e resposta rápidas a falhas. A capacidade de discernir a indisponibilidade de um monitor e, consequentemente, avaliar a confiabilidade da região afetada é crucial para identificar a falha de uma região e agir.

4.3 Classe MonitorDeThreads

4.3.1 Diagrama de Atividades da Classe MonitorDeThreads

A Figura 13 ilustra o diagrama de atividades da classe `MonitorDeThreads`, que atua como um guardião, mantendo um olhar atento sobre as *threads* que realizam requisições CoAP, assegurando, assim, a integridade e a operacionalidade da rede de monitoramento.

No início do fluxo, o processo é deflagrado pela inicialização do controlador e do agendador pertencentes à classe mencionada, este estágio é crítico, pois configura o ambiente para que as tarefas subsequentes sejam executadas com precisão. A fase de agendamento, onde as tarefas CoAP são programadas para cada tópico de interesse é essencial a obtenção do identificador único para cada tópico, que serve como um ponto de referência para as operações futuras. A inicialização do estado da *thread* é realizada para cada tópico, e é seguida pelo agendamento do envio periódico de requisições CoAP.

Essas tarefas agendadas são acompanhadas de perto, armazenando o estado de cada *thread* após a execução da atividade agendada, o que facilita a gestão e a recuperação do estado em momentos subsequentes.

O processo iterativo é aplicado a todos os tópicos configurados, formando um ciclo de preparação que antecede a fase de monitoramento, a medida que o sistema transita para a vigilância ativa, o monitoramento de *threads* é instaurado através do agendamento de verificações periódicas, este ciclo de vigilância é continuo e as *threads* são verificadas em intervalos definidos.

A cada ciclo, há uma pausa de 1 minuto correspondente ao tempo de verificação estipulado, o que reflete a periodicidade com que o estado do sistema é inspecionado. A fase seguinte é a de checagem e, se necessário, de reinicialização das *threads*. Nesta fase, as *threads* são inspecionadas individualmente para determinar a sua condição operacional.

Caso uma *threads* esteja inoperante, um procedimento de reinício é iniciado, o que envolve a suspensão da tarefa previamente agendada e a subsequente reagendamento da mesma. O mecanismo garante que tópicos desatualizados sejam identificados e reativados, e os erros de *threads* Inativas e tópicos não encontrados são devidamente registrados, assegurando que o sistema mantenha um registro fiel do seu estado.

A verificação ocorre nas seguintes condições é verificado o tempo em que a *Thread* foi executada pela última vez e caso a mesma tenha sido executada a mais de 5 segundos ela será considerada inativa e será reiniciada, para que ocorra o reagendamento da *thread* e extraído o ID de identificação da URL e caso ele exista dentro da lista de nós a sua respectiva *thread* será reiniciada, em resumo, o diagrama de atividades descreve um sistema que assegura a operacionalidade, das *threads* através da classe `MonitorDeThreads`.

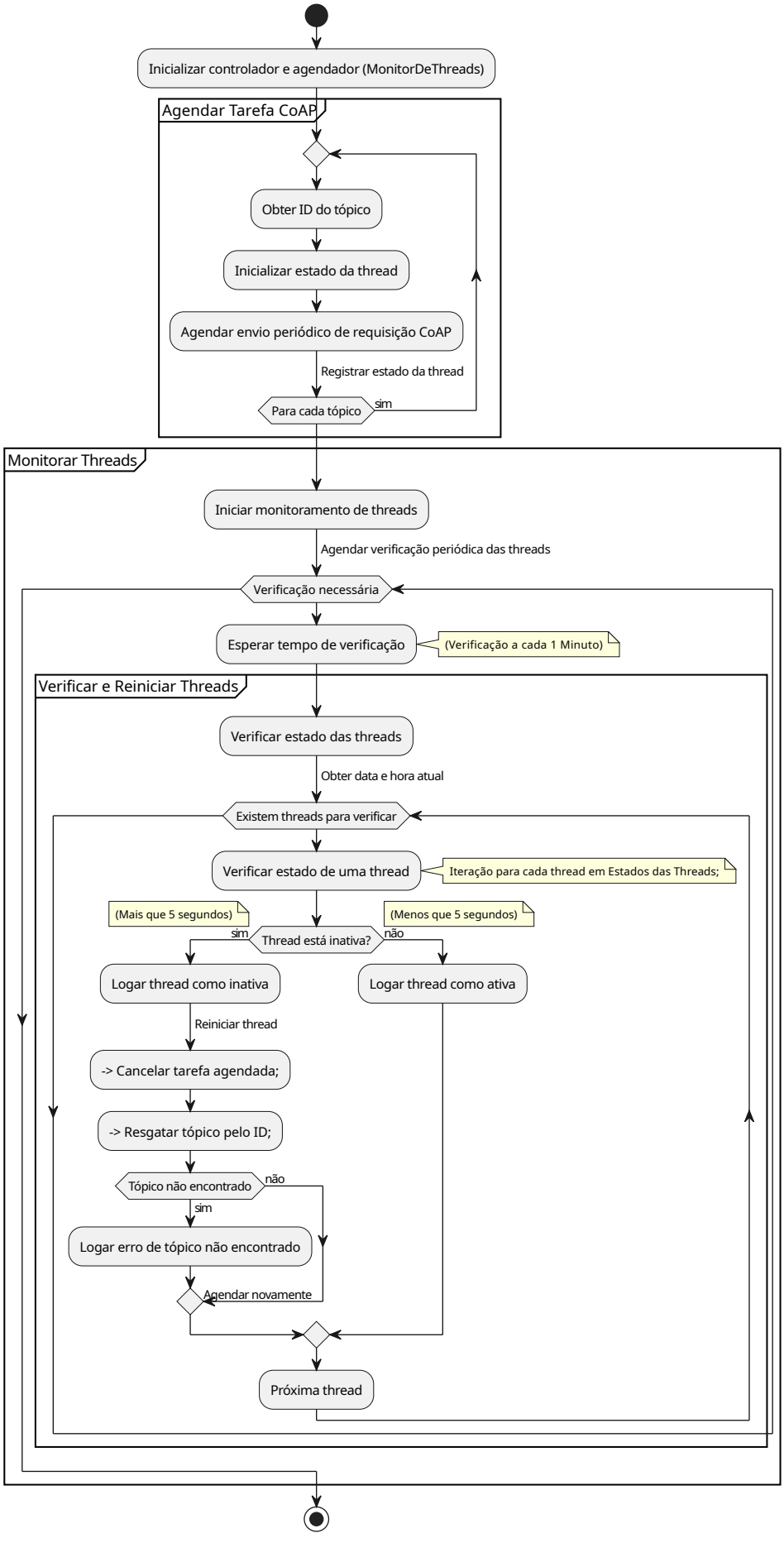

Figura 13 – Diagrama de Atividades da Classe MonitorDeThreads.

4.3.2 Construtor da Classe MonitorDeThreads

O propósito deste construtor é duplo: primeiro, ele associa a classe de monitoramento ao controlador de comunicações CoAP, permitindo que as *threads* enviem e gerenciem as requisições para os dispositivos IoT. Segundo, ele configura o serviço de agendamento, que será o coração do sistema de monitoramento, responsável por iniciar e manter a execução periódica das tarefas. Através da integração com o a classe `HeartbeatController`, o sistema adquire a capacidade de interagir com a rede de dispositivos IoT, enquanto a especificação do tamanho do pool de *threads* é definido como um para um, cada dispositivo IoT possui sua *thread*. A instância de `ScheduledExecutorService` criada pelo construtor é que possibilita a execução regular das *threads*, permitindo o agendamento de tarefas que verificarão o estado das comunicações CoAP e atuarão prontamente caso alguma inconsistência ou falha seja detectada, este serviço de agendamento é crucial para a robustez do sistema de detecção de falhas. Nesse panorama, utilizar o `ScheduledFuture<?>` fornece uma gestão refinada para as tarefas agendadas, pois cada tarefa possui um `ScheduledFuture<?>` associado, o que concede ao sistema o poder de supervisionar e intervir no ciclo de vida das operações programadas, viabilizando o cancelamento ou a reinicialização conforme necessário, esse aspecto é vital para a robustez do sistema de detecção de falhas, garantindo uma resposta ágil frente a incidentes ou instabilidades na rede.

4.3.3 Métodos da Classe MonitorDeThreads

A classe `MonitorDeThreads` contém métodos que estabelecem uma estrutura para o agendamento e o monitoramento contínuo das *threads*. A implementação começa com o construtor da classe, que recebe um controlador CoAP e um tamanho de pool de *threads*, indicando o número de *threads* de trabalho que o executor terá disponíveis. Este construtor é o ponto de partida para o agendamento e o monitoramento das tarefas.

O **método agendarTarefa** é utilizado para programar as *threads* para envio periódico de requisições CoAP. Ao receber um objeto `URL` que representa um tópico específico, o método extrai o identificador do tópico, inicializa o estado da *thread* com um valor de "last run" que indica uma execução anterior e coloca essa informação em um mapa de estados de *threads*. A função de agendamento do executor é então chamada, onde uma tarefa anônima é definida para atualizar o "last run" e enviar a requisição CoAP. Essa ação é agendada para execução em uma taxa fixa.

Por sua vez, o **método monitorarThreads** se encarrega de iniciar o processo de monitoramento. Ele agenda, em uma base periódica, a execução do método `checkAndRestartThreads`, que realiza a verificação do estado de cada *thread*.

O **método checkAndRestartThreads** analisa os tempos de execução das *threads*, comparando-os com um limite pré-definido. No caso de uma *thread* não ter executado dentro do limite de tempo esperado, ela é considerada inativa, e ações de reinício são tomadas.

O **método restartThread** é invocado quando uma *thread* é determinada como inativa. Ele cancela a tarefa agendada, se ainda estiver ativa, e tenta resgatar o tópico pelo ID. Se o tópico for encontrado, a tarefa é agendada novamente, promovendo a recuperação da *thread* e mantendo a continuidade do monitoramento.

4.4 Classe HeartbeatController

4.4.1 Diagrama de Atividades da Classe HeartbeatController

A classe `HeartbeatController` foi construída para orquestrar as comunicações utilizando o protocolo CoAP com dispositivos IoT e protocolo ICMP para comunicação com o monitor adjacente.

No cerne do processo de inicialização, o construtor recebe uma lista de tópicos, que são *endpoints* de dispositivos na rede IoT. Essa lista é atribuída à variável estática `lista_topicos`, que fornece um acesso universal dentro da classe às URLs destinadas ao monitoramento. Este mecanismo garante que os tópicos designados estejam disponíveis para as operações de monitoramento e interação do controlador com a rede. A variável estática `conf` é igualmente definida para incorporar um conjunto de configurações globais cruciais para as operações subsequentes da classe. Estas configurações abrangem desde limiares de detecção de falhas e temporizadores fornecendo uma base sólida para todas as funcionalidades do `HeartbeatController`.

A integração do sistema de detecção de falhas é realizada por meio da instanciação do objeto `DisasterFd`, parametrizado por elementos estratégicos retirados da configuração `conf`, segundo argumentos do construtor.

Por meio desse construtor, o `HeartbeatController` é habilitado com todos os recursos essenciais para dar início ao processo de monitoramento, estabelecendo uma ponte entre as diretrizes operacionais e a execução prática das comunicações CoAP. O construtor assegura que a classe esteja completamente preparada para responder de maneira dinâmica e confiável a uma gama de condições operacionais, cumprindo assim o papel vital de manter a resiliência e a eficiência da rede IoT em situações de desastres, a Figura 14 apresenta o diagrama de classes `HeartbeatController` com seu respectivo relacionamento com as demais classes `DisasterFd, MonitorDeThreads, Conf, Url e Tarefas`.

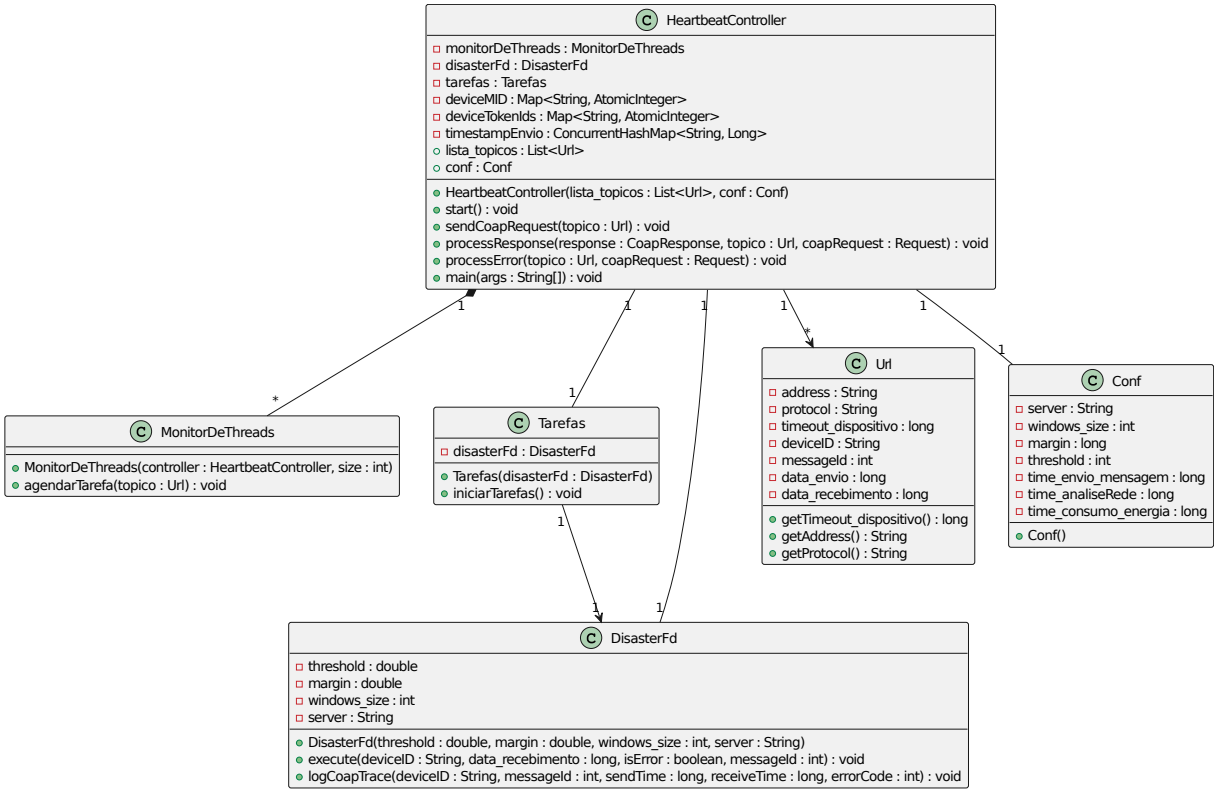

Figura 14 – Diagrama de Classe HeartbeatController.

Além disso, a lista de URLs é atribuída a uma variável estática, `lista_topicos`, disponibilizando globalmente, na classe, os tópicos designados para monitoramento. Quanto ao parâmetro de configuração, denominado `conf`, este serve como um repositório centralizado para todas as configurações essenciais ao funcionamento do `HeartbeatController`. Esse conjunto de configurações pode abarcar desde parâmetros de segurança e temporizadores até limiares para a detecção de falhas. A principal vantagem de encapsular essas definições em um único objeto é a facilidade com que ele pode ser distribuído pelo código, facultando às diversas seções do controlador o acesso às configurações requeridas.

O objeto `disasterFd` é inicializado no construtor com vários parâmetros provenientes do objeto `conf`:

- **conf.threshold**: refere-se a um valor que define um limite inferior para o nível de confiança no conjunto de processos monitorados, permitindo determinar se o sistema é considerado "confiável" ou "não confiável".
- **conf.margin**: inclui uma margem de segurança, denotada por β, usado para o cálculo do tempo de espera para a próxima mensagem.
- **conf.windows_size**: refere-se aos η sinais de vida (heartbeats) mais recentes usados para estimar a chegada da próxima mensagem.
- **conf.server**: refere-se ao endereço do servidor ou identificador que é usado para monitorar a saúde da rede ou dos dispositivos.

O objeto "Tarefas" é instanciado e integrado ao `DisasterFd`, refletindo a estreita relação entre as responsabilidades de monitoramento do consumo de energia e a análise do nível de confiança da rede. A Figura 15 apresenta o diagrama de atividades que descreve o fluxo de operações dos componentes `HeartbeatController` dentro de um sistema de detecção de falhas destinado a ambientes sujeitos a desastres, como parte do sistema *Disaster-FD*.

O processo começa com a inicialização do `HeartbeatController`, que serve como um ponto de partida para o controle de tarefas e monitoramento de *threads*.

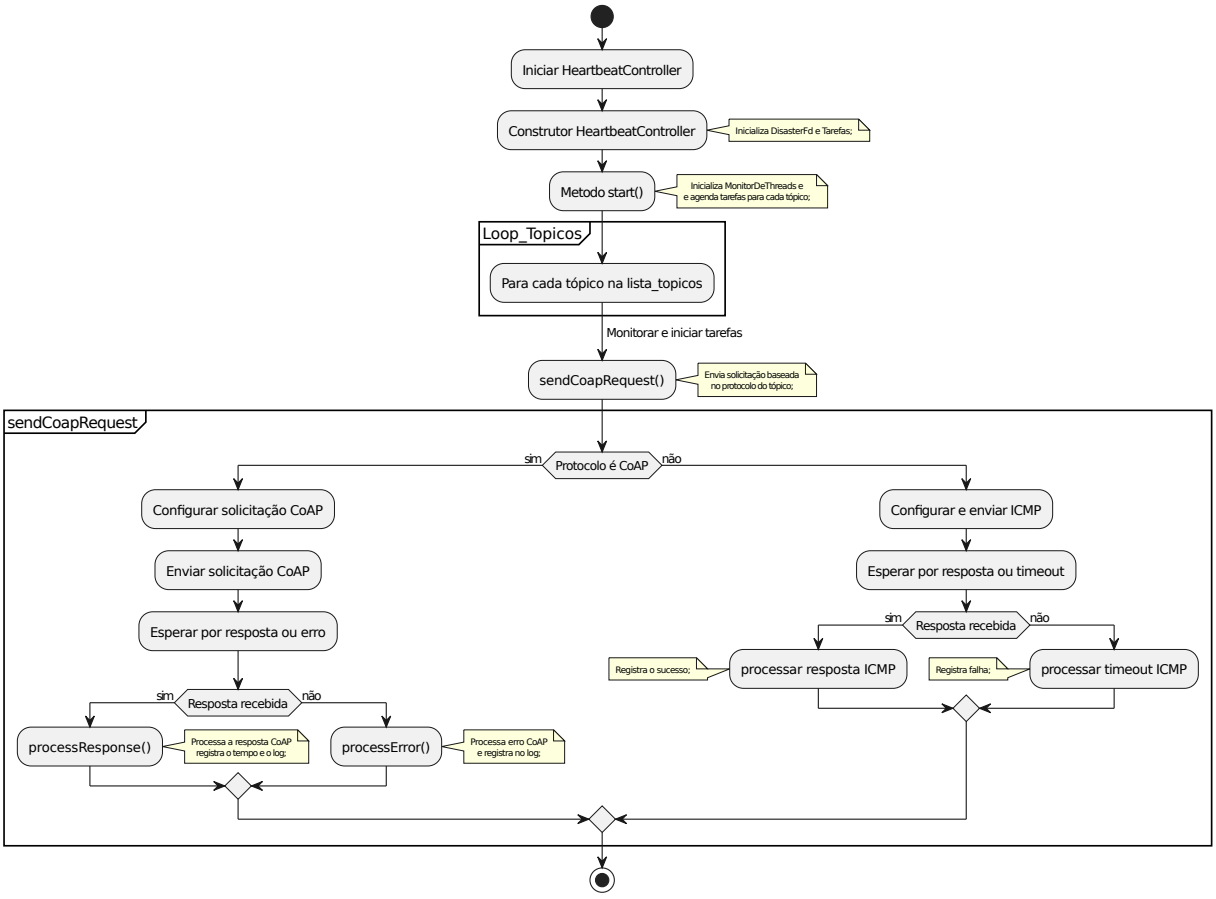

Figura 15 – Diagrama de Atividades da Classe HeartbeatController.

Após a inicialização, entra-se em um *loop* no qual o sistema, para cada tópico na lista, começa a monitorar e iniciar tarefas associadas.

Isso sugere um mecanismo de escuta que aguarda dados ou eventos que precisam ser tratados. Em seguida, há uma tentativa de enviar solicitações com base no protocolo do tópico em questão.

Na atividade `sendCoapRequest`, o diagrama se divide em dois caminhos distintos, dependendo do protocolo utilizado, caso o protocolo seja CoAP, um subfluxo específico é iniciado, este subfluxo envolve a configuração e o envio de solicitações CoAP, e aguarda respostas ou erros. No caso de uma resposta bem-sucedida, a resposta é processada adequadamente, contudo se ocorrer um erro ou não havendo resposta, ocorre o processamento adequado para esse cenário.

Por outro lado, se o protocolo não for CoAP, o sistema configura e envia solicitações ICMP, uma alternativa para comunicação entre dispositivos na rede. Da mesma forma, o sistema aguarda por uma resposta ou *timeout*. Em caso de resposta, a informação é processada e detalhes da resposta são registrados. Caso ocorra um *timeout*, indicando uma falha na obtenção de resposta, este evento também é processado e registrado esse fluxo de operações demonstra uma abordagem robusta para a comunicação em um sistema de IoT onde a detecção de falhas é crítica.

O mecanismo de decisão entre CoAP e ICMP sugere que o sistema foi projetado para ser flexível e compatível com múltiplos protocolos de rede, o que é fundamental em cenários de desastres onde diferentes tecnologias podem ser necessárias para garantir uma comunicação confiável. A parte inferior do diagrama detalha as operações de tratamento de respostas e erros, que são vitais para manter a integridade do sistema de monitoramento. Este processo de resposta e erro é crítico, pois assegura que cada evento ou dado recebido seja adequadamente registrado e tratado, mantendo a confiabilidade e eficiência do sistema em condições potencialmente instáveis e propensas a falhas, como as encontradas em cenários de desastres naturais, sendo assim os respectivos métodos `processResponse()` e `processError()` serão apresentados mais adiante neste mesmo trabalho.

Método ProcessResponse

A Figura 16 ilustra o diagrama de atividades associado ao método `processResponse()` detalhando o fluxo lógico essencial para o gerenciamento de mensagens dentro do sistema Disaster-FD. Este método é um componente crítico da classe `HeartbeatController`, encarregado de processar as respostas de requisições enviadas para dispositivos IoT em uma rede distribuída.

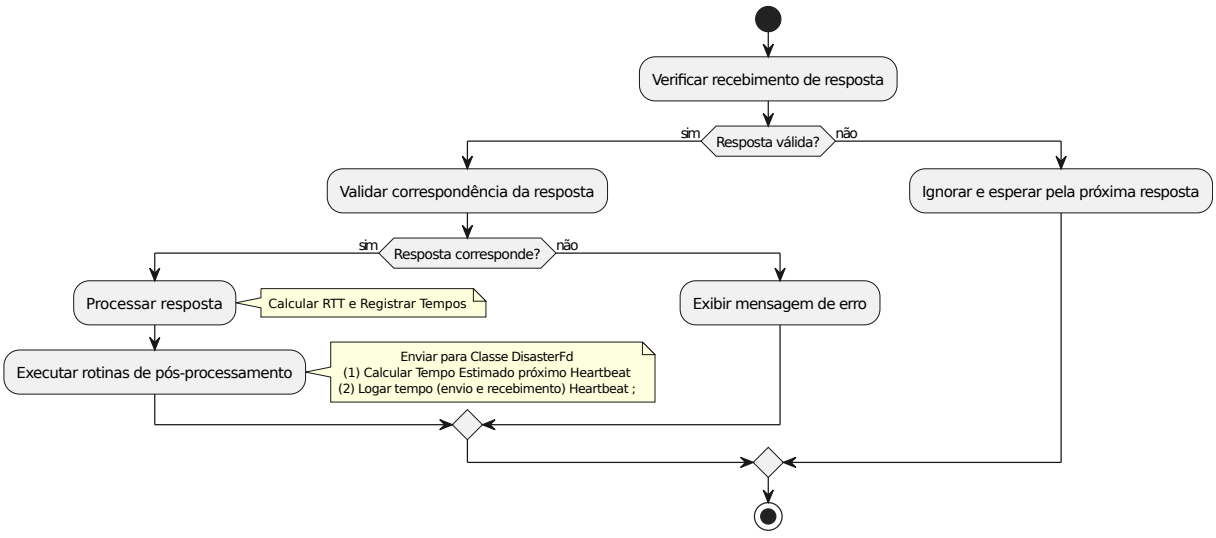

Figura 16 – Diagrama de Atividade ProcessResponse.

O método inicia sua execução com uma checagem da resposta obtida, caso constatada a presença de uma resposta válida ou seja, a resposta não é nula, ele avança para a inspeção do identificador da

mensagem, conhecido como MID. Este passo é imprescindível para assegurar que há uma correspondência exata entre a resposta e a requisição que a originou, a correspondência correta desses identificadores é fundamental para a integridade da comunicação assíncrona, associando corretamente a mensagem enviada com a mensagem recebida.

Após a confirmação de que a resposta está corretamente alinhada com a requisição, o método registra métricas importantes, como os tempos de envio e recebimento, remove tempos de envio mais antigos, para limpar o método, além disso avalia a latência da rede por meio do cálculo do Round Trip Time (RTT), entre o tempo de envio e recebimento da mensagem, está métrica é um indicador chave da performance da comunicação do dispositivo.

Concluindo o processamento da resposta, o método realiza ações de pós-processamento, como o envio desse tempo de recebimento e os respectivos IDs da mensagem e do dispositivo para a classe `DisasterFd` responsável por calcular o tempo estimado de chegada da próxima mensagem, o nível de confiança da rede, além manter um registro dos *heartbeats* recebidos, facilitando futuras análises de desempenho e diagnósticos. O processo descrito é executado de maneira assíncrona, integrando-se ao sistema *Disaster-FD* para garantir uma resposta contínua e eficiente diante das dinâmicas de uma rede IoT distribuída e, muitas vezes, operando sob condições críticas.

Método ProcessError

O método `processError` é uma funcionalidade essencial no `HeartbeatController`, desempenhando uma função no gerenciamento em situações onde as requisições CoAP não recebem resposta, um indicativo de falha de comunicação ou de perda de mensagem.

A Figura 17 ilustra o diagrama de atividades que detalha o fluxo de procedimentos quando uma requisição falha é detectada, garantindo que as falhas sejam adequadamente registradas e tratadas, Em termos gerais, o processo começa com a identificação do identificador da mensagem sem resposta. A chave correspondente é localizada no registro do sistema para confirmar o envio anterior da solicitação, contudo se essa confirmação falhar, um erro é informado “Registro de envio não encontrado”.

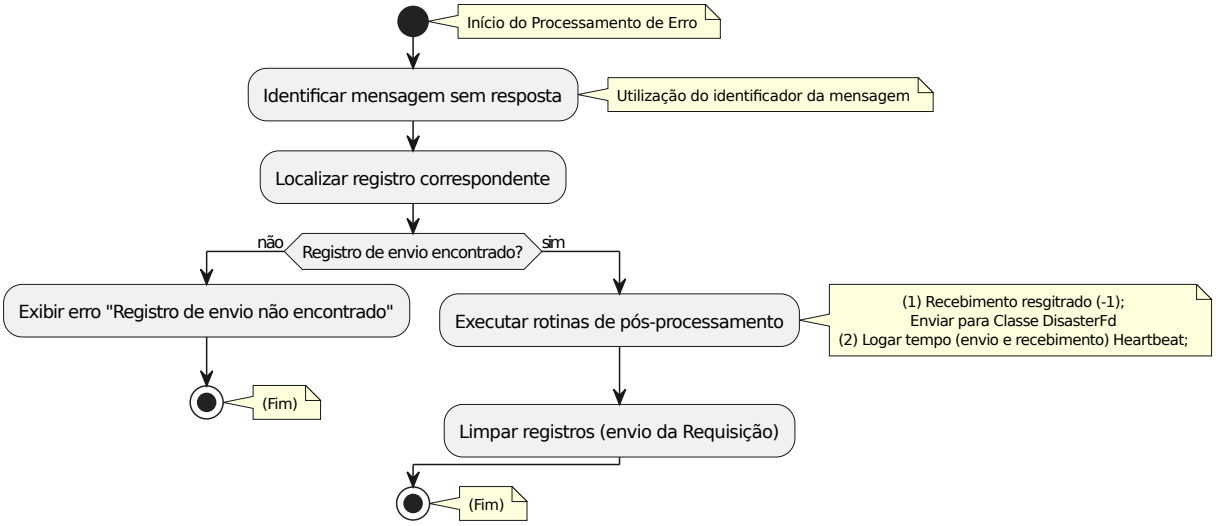

Figura 17 – Diagrama de Atividade ProcessError.

Caso a requisição tenha sido enviada , o sistema ajusta os indicadores para refletir a não recepção da resposta, definido o tempo de recebimento como -1 (**valor um negativo**), simbolizando a ausência de resposta. Em seguida, são executadas ações de pós-processamento registrando no arquivo de *log* que a requisição enviada não retornou uma resposta, mantendo um *log* mais completo acompanhamento tanto

mensagens bem sucedidas quanto as falhas dos respectivos dispositivos IoT. Além disso o método remove as requisições falhas para evitando a retenção de informações obsoletas.

4.5 Classe Disaster-FD

A classe *Disaster-FD* é uma implementação dentro do sistema de detecção de falhas projetado para ambientes IoT sujeitos a desastres. Sua principal função é monitorar ativamente a rede e verificar a confiabilidade de cada dispositivo conectado empregando a comunicação de sinais conhecidos como "heartbeats", desta forma estabelecendo o nível de confiança da rede. A classe foi projetada para ser capaz de operar em condições onde a comunicação pode ser irregular e os recursos limitados, esta classe é responsável pela implementação de um algoritmo que não somente monitora a conectividade dos dispositivos IoT através da recepção de *heartbeats*, mas também estabelece e mantém um modelo de confiança para a rede como um todo, característica herdada do projeto *Impact-FD* (ROSSETTO et al., 2018), incorporando o uso de elementos como limiar de confiabilidade, nível de confiança e fator de impacto, que são fundamentais para a análise de confiança em redes IoT.

4.5.1 Funcionalidades da Classe Disaster-FD

1. **Monitoramento de Heartbeats**: Gerencia a recepção de *heartbeats* dos dispositivos, que indicam sua operacionalidade. Além disso, registra o momento exato em que cada *heartbeat* é recebido para avaliar a confiabilidade desses dispositivos na rede.

2. **Determinação de Confiabilidade Individual e da Rede**: Avalia a confiabilidade de cada dispositivo individualmente e da rede como um todo, considerando o recebimento contínuo de *heartbeats* e utilizando um algoritmo para estabelecer o nível de confiança.

3. **Registro e Gerenciamento de Erros**: Documenta eventos críticos como a detecção de falhas, contribuindo para o diagnóstico e a resposta a incidentes.

4. **Estimativa Proativa**: Projeta os tempos esperados de recebimento de *heartbeats* futuros, possibilitando a identificação antecipada de possíveis falhas.

A Figura 18 ilustra o diagrama de classes da entidade *Disaster-FD*, ilustrando sua estrutura o diagrama destaca a classe `DisasterFd` e a classe auxiliar `DeviceReply`, revelando os atributos e métodos responsáveis pelo monitoramento ativo da rede. A seta indicativa de "contém" reflete a relação entre as classes, demonstrando como a classe `DisasterFd` armazena e gerencia as respostas dos dispositivos, ou "*heartbeats*", encapsulando a lógica de dependência entre as duas classes.

Métodos da Classe

- **Execute**: Este método é chamado para processar cada *heartbeat* recebido de um dispositivo, ele utiliza o identificador do dispositivo (id), o *timestamp* do recebimento (data_recebimento), um `booleano` que indica se houve *timeout* e o identificador da mensagem do *heartbeat* (idmensagem). Este método é o núcleo da lógica de monitoramento, lidando com a atualização da confiabilidade dos dispositivos e da rede como um todo.

- **sumTrusted**: Retorna o peso total dos fatores de impacto dos dispositivos que são considerados confiáveis na rede. É um método que ajuda a calcular o nível de confiança da rede.

Figura 18 – Diagrama de Classes Disaster-FD.

- **ComputeEAWithMsgId**: Similar ao `computeEA`, mas calcula a Estimativa de Chegada levando em consideração o ID da mensagem do *heartbeat*, isso permite uma estimativa mais precisa, que pode ser crucial para sistemas onde a precisão temporal é vital.
- **logCoapTrace**: Registra as informações de rastreamento dos *heartbeats* protocolos CoAP/ICMP, incluindo o identificador do dispositivo (idDispositivo), o identificador da mensagem (idMensagem), e os tempos de requisição (time_request) e de resposta (time_response). Este método pode ser utilizado para fins de depuração e para analisar a latência da rede.
- **RealizarEstatistica**: Executa a análise estatística da rede. Este método pode ser utilizado para processar os dados coletados, fornecendo *insights* sobre o desempenho da rede e a detecção de tendências anormais que podem sinalizar falhas.

4.5.2 Estimativa de Chegada (*Estimated Arrival*)

O termo "EA" na classe Java `DisasterFd` se refere à "Estimativa de Chegada" (*Estimated Arrival*) de *heartbeats* subsequentes de dispositivos em uma rede IoT, este conceito é fundamental para a detecção de falhas, uma vez que permite ao sistema prever quando o próximo sinal de vida (*heartbeat*) é esperado de cada dispositivo monitorado, com esta previsão o sistema pode identificar desvios que possam indicar problemas operacionais ou falhas.

O cálculo de EA é efetuado pelo método `computeEAWithMsgId`. Este método analisa a sequência de *heartbeats* recebidos, combinando dados históricos com um fator ajustável que representa o intervalo de tempo entre os envios dos *heartbeats*. Com base nesta estimativa de chegada (EA), a classe define o tempo previsto para a chegada da próxima mensagem ao dispositivo. Adicionalmente, é incluída uma "margem

de segurança", um valor de tempo extra destinado a absorver atrasos não relacionados ao funcionamento intrínseco do dispositivo, mas que podem ocorrer devido a variações na rede ou outros fatores externos, caso um *heartbeat* não seja recebido até este momento pré-estipulado, o dispositivo é considerado suspeito de falha.

O projeto *Impact-FD* (ROSSETTO et al., 2018) utiliza o método `computeEA` para calcular a Estimativa de Chegada (EA) do próximo *heartbeat*, contudo existem algumas diferenças na maneira como cada método aborda o cálculo, detalhadas na seção seguinte.

4.5.2.1 ComputeEA vs. ComputeEAWithMsgId

O método `computeEA` implantado no *Impact-FD* de (ROSSETTO et al., 2018), foi projetado para calcular a estimativa de chegada de *heartbeats* em uma sequência onde cada mensagem é assumida como parte de uma série contínua. Ele incrementa um contador cada vez que um *heartbeat* é recebido, aplicando um cálculo baseado no número sequencial de *heartbeats*.

Esta abordagem é direta e eficaz em cenários onde as mensagens são recebidas sem interrupções ou perdas significativas de mensagens. Entretanto, em ambientes onde as mensagens podem não chegar de forma sequencial ou podem ser perdidas, este método pode enfrentar desafios para ajustar suas estimativas de tempo, uma vez que baseia-se na premissa de uma sequência contínua e ininterrupta.

Por outro lado, o método `computeEAWithMsgId` introduz uma abordagem adicional ao incorporar o identificador único de cada mensagem, o MID, no cálculo. Este detalhe permite que o método trate cada *heartbeat* como uma entidade distinta, ajustando o tempo de chegada com base em seu identificador específico.

Esta abordagem é particularmente útil em redes onde as mensagens podem experimentar diferentes latências ou até mesmo em casos onde a ordem de chegada pode não refletir a ordem de envio. Utilizando o MID para ajustar o cálculo, o método `computeEAWithMsgId` pode fornecer uma representação mais precisa do comportamento da rede, adaptando-se a variações que podem ocorrer devido a condições de rede dinâmicas ou instáveis.

A aplicabilidade de `computeEA` é ideal em redes com alta confiabilidade e baixa taxa de perda de mensagens, por outro lado, o método `computeEAWithMsgId` é recomendado para ambientes de rede voláteis como ecossistemas IoT ou para aplicações onde a precisão na estimativa de tempo e a rápida detecção de falhas são críticas.

A utilização do MID no `computeEAWithMsgId` permite uma detecção de falhas mais responsiva e confiável, pois a ausência de uma mensagem específica é rapidamente identificada, ao contrário do `computeEA`, que pode não detectar prontamente a falha ou perda de mensagens, especialmente se várias mensagens consecutivas forem perdidas.

A escolha entre esses métodos deve considerar as características específicas do ambiente de rede e as necessidades do sistema monitorado. Ambos os métodos apresentam particularidades que destacam a importância de uma estratégia de cálculo bem ajustada ao contexto operacional.

4.5.3 Método Execute - Monitor de Heartbeats

O método `execute` no contexto do sistema *Disaster-FD* desempenha um papel no monitoramento e gestão da confiabilidade dos dispositivos em uma rede de IoT, além de definir o nível de confiança do sistema como um todo, particularmente em cenários onde a estabilidade da comunicação é crítica. Este método integra diversas funcionalidades que juntas contribuem para a manutenção da integridade e eficiência da rede, tais como fator de impacto dos dispositivos, margem de segurança, limiar da rede e o calculo estimado de chegada do próximo *heartbeats*.

Inicialmente, o método trata da recepção de *heartbeats* dos dispositivos, iniciando o processo ao registrar o tempo de chegada do primeiro *heartbeat*. Essa ação estabelece um marco inicial para análises subsequentes e ajustes necessários no monitoramento do sistema.

Em seguida, verifica-se a presença de *timeout*, que indica se um dispositivo falhou em enviar seu *heartbeat* dentro do intervalo esperado, caso um *timeout* ocorrá, o dispositivo será considerado não confiável, o que ativa procedimentos para documentar a falha e ajustar o nível de confiança da rede.

A confiabilidade de cada dispositivo em uma rede de IoT é monitorada tanto individualmente quanto em um contexto coletivo, utilizando um modelo que avalia o nível de confiança da rede baseado em fatores de impacto atribuídos a cada dispositivo. Esses fatores são essencialmente pesos numéricos que representam a importância relativa de cada dispositivo para a funcionalidade e segurança da rede como um todo.

Uma vez estabelecidos os fatores de impacto, o nível de confiança da rede é calculado somando-se os fatores de impacto de todos os dispositivos considerados confiáveis em um determinado momento. Esse total é então comparado com um limiar predefinido, que é determinado com base em critérios específicos, como requisitos de segurança ou operacionais. Se o nível de confiança cair abaixo desse limiar, isso indica uma redução na confiabilidade geral da rede, levando a ações como registrar o inicio do erro ou seja quando a rede não é mais confiável e logo esse cenário de não confiabilidade da rede se reverta o método `execute` calcula o tempo de erro da rede e a soma de erros ocorridos na rede.

Além disso, o método execute é responsável por calcular o tempo estimado para a chegada do próximo *heartbeat* de cada dispositivo, utilizando uma janela das 100 (cem) últimas mensagens recebidas do dispositivo especifico, calculo este mencionado na seção 4.5.2, com base nesse cálculo não só teremos uma previsão de quando o próximo *heartbeat* deve chegar, mas também ajusta os intervalos de *timeout* do dispositivos IoT dentro do sistema.

As decisões e ações dentro deste método são continuamente registrados e analisados, permitindo ajustes dinâmicos ao comportamento da rede, garantindo que o sistema mantenha uma operação robusta e resiliente. A Figura 19 apresenta o diagrama de atividades do método execute implementado na classe `DisasterFd`, que é responsável por monitorar a confiabilidade de dispositivos e o nível de confiança da rede como um todo, além de calcular o EA da próxima mensagem.

4.5.3.1 Evolução da Classe DisasterFd

O *Disaster-FD* emerge como um sistema projetado para o monitoramento das redes IoT, construído sobre o legado robusto do Impact-FD (ROSSETTO et al., 2018), ele não só adota, mas também utilizas boas práticas para enfrentar os desafios emergentes do ambiente IoT. Distanciando-se da abordagem estática de seu predecessor, que confiava na análise de um arquivo de trace, o *Disaster-FD* realiza o calculo do nível de confiança da rede de maneira contínua e em tempo real.

O *Disaster-FD*, influenciado pela base sólida do *Impact-FD*, avança com uma série de funcionalidades focados para o monitoramento em tempo real. Ele processa *heartbeats* conforme são recebidos, permitindo que cada evento de comunicação contribua imediatamente para a avaliação da confiabilidade da rede. Essa análise em tempo real é crítica em cenários onde a prontidão para responder a falhas ou anomalias é crucial.

Adicionalmente, o *Disaster-FD* introduz uma gestão *heartbeats* e *timeouts*, monitorando não apenas os sinais recebidos, mas também os ausentes, cada um afetando o estado de confiança da rede IoT. Isso permite que o sistema adapte dinamicamente seu entendimento e reação aos estados de confiança dos dispositivos individualmente e da rede como um todo.

Enquanto o *Impact-FD* realiza a leitura única de um arquivo predefinido, o *Disaster-FD* emprega uma fila dinâmica que mantém um registro contínuo e atualizado dos tempos de resposta dos dispositivos. Essa

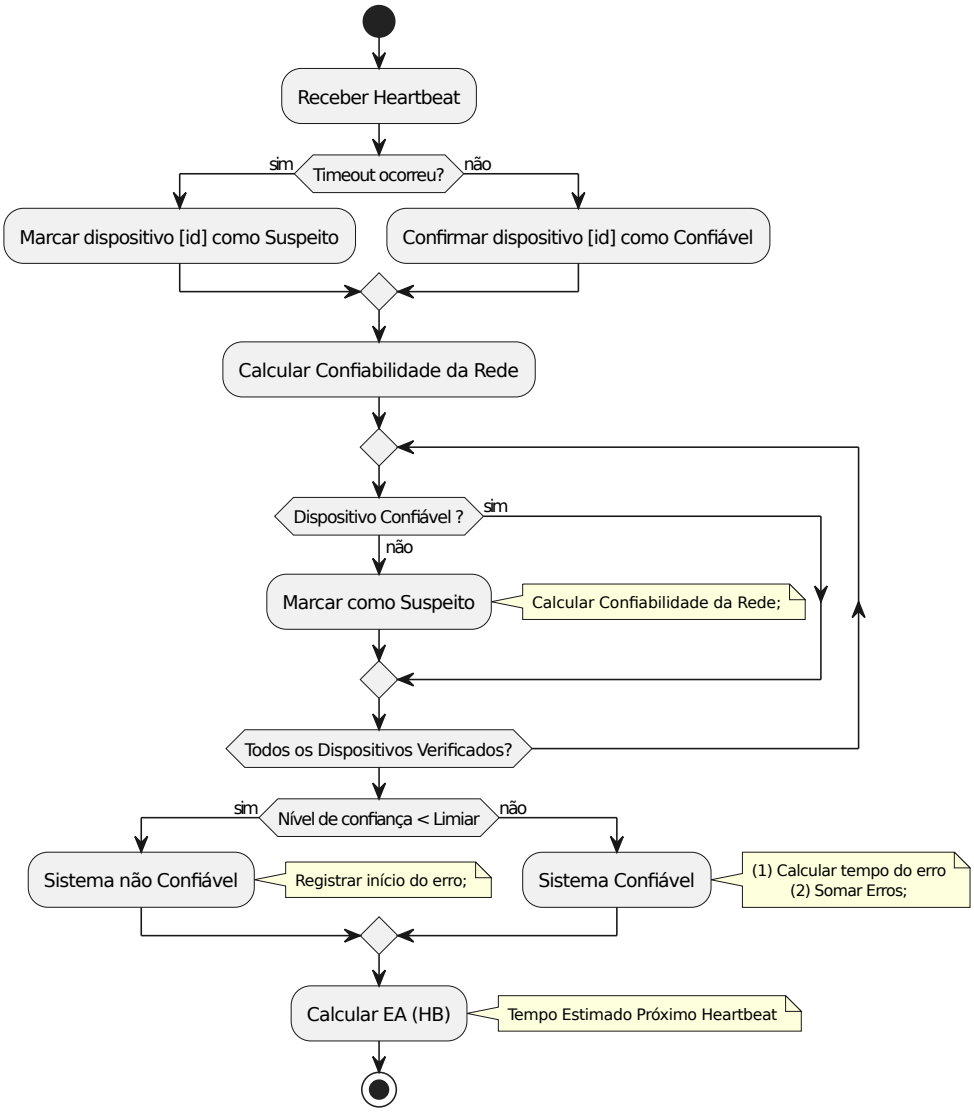

Figura 19 – Diagrama de Atividades Método Execute.

estratégia cria um histórico vivo que serve para diagnósticos mais profundos e análises de tendências ao longo do tempo, assim como seu predecessor.

Com a integração dessas funcionalidades o *Disaster-FD* honra as fundações estabelecidas pelo *Impact-FD*, além de focar no monitoramento em tempo real, o que é indispensável para redes IoT operando em ambientes que requerem vigilância constante e capacidade de resposta rápida diante de imprevistos e desastres.

4.6 Preparação para Testes e Influências do Impact-FD

Conforme delineado no presente capítulo desta dissertação, os próximos passos incluem a realização de testes experimentais na plataforma IoT-LAB, esses testes serão essenciais para validar a abordagem proposta e permitirão avaliar a confiabilidade e eficiência do sistema *Disaster-FD* sob condições similares de uso real em uma rede IoT.

Este trabalho tem como objetivo aderir às diretrizes do *Impact-FD*, conforme descrito por (ROSSETTO et al., 2018). O *Impact-FD* é um modelo reconhecido e já foi utilizado como referência em diversos estudos na área de sistemas de monitoramento e previsão de desastres, incluindo os trabalhos

de (PASQUINI et al., 2020) e (SILVEIRA et al., 2022). Essas referências destacam a importância e a relevância do *Impact-FD* no desenvolvimento de soluções para detecção de falhas e gestão de desastres em ambientes IoT.

A fase de experimentação com o *Disaster-FD* não só visa validar as funcionalidades e métricas propostas, mas também contribuir para o corpus de conhecimento estabelecido pelo *Impact-FD*, oferecendo percepções que podem aprimorar ainda mais as práticas de monitoramento e resposta em cenários de desastres. Através desses estudos e testes, espera-se consolidar o *Disaster-FD* como um sistema promissor para o monitoramento em ambientes de ecossistemas IoT.

Capítulo 5

Experimentos e Análise dos Resultados

Neste capítulo, são apresentados e analisados os resultados de uma série de experimentos realizados na plataforma FIT IoT-LAB. Esses experimentos são projetados para avaliar a eficácia do *Disaster-FD*, um sistema de detecção de falhas para redes IoT em ambientes suscetíveis a desastres, por meio de envio de mensagens a diversos dispositivos IoT em ambientes que replicam condições reais. Durante o experimento de 24 horas, o *Disaster-FD* é testado quanto à sua capacidade de monitorar continuamente dispositivos IoT e detectar falhas, como perda de comunicação e anomalias no tráfego de rede, utilizando os protocolos CoAP e ICMP. Os experimentos abordam questões críticas, como intervalos de envio de requisições, comunicação entre regiões federadas e a resposta do sistema a falhas. Foca-se na avaliação da margem de segurança, no consumo de energia dos dispositivos e no nível de confiança da rede.

5.1 Estrutura Geral do Experimento utilizando a Plataforma IoT-LAB

A plataforma IoT-LAB fornece um ambiente de experimentação heterogêneo, que permite a integração de diferentes dispositivos e tecnologias. Os experimentos podem ser configurados através de uma interface web intuitiva ou por meio de ferramentas de linha de comando (CLI) para usos mais avançados e automatização de tarefas. A infraestrutura da plataforma é suportada por uma API RESTful aberta, que gerencia as solicitações dos usuários e distribui comandos entre os diferentes sites de experimentação.

A IoT-LAB está implantada em seis locais na França: Grenoble, Lille, Paris, Saclay , Strasbourg e Toulouse, até o presente momento, cada uma oferecendo uma combinação única de sensores e robôs móveis, adequados para experimentação diversificada, incluindo sincronia GPS e mobilidade controlada.

5.1.1 Arquitetura de Nós

Para atender a diferentes necessidades experimentais, a IoT-LAB organiza sua infraestrutura de nós em três componentes principais: *Open Nodes (ON)*, *Gateways (GW)*, e *Control Nodes (CN)*. Esses componentes trabalham juntos para fornecer um controle completo sobre a configuração do nó, monitoramento de desempenho, e coleta de dados. Os nós são conectados a uma rede *backbone* que proporciona conectividade segura e alimenta remotamente os dispositivos.

- **Open Node (ON):** Um dispositivo de baixo consumo de energia que pode ser reprogramado pelo usuário. É o componente principal para experimentação direta e inclui sensores e interfaces de comunicação.

- **Gateway (GW):** Um pequeno computador Linux que se conecta ao ON e ao CN. O GW é responsável pelo monitoramento, reprogramação e transmissão de dados do ON para os servidores *backend*.
- **Control Node (CN):** Controla o ON, incluindo a seleção da fonte de energia (bateria ou rede elétrica), e monitora o consumo de energia, além de atuar como *sniffer* de pacotes de rede.

Hardware IoT-LAB e Tipos de Nós

A IoT-LAB suporta uma variedade de hardware, como os nós M3 e A8, cada um representando diferentes níveis de capacidade e complexidade de dispositivos IoT. Os M3 representam dispositivos IoT modernos com múltiplos sensores. Os A8 são os nós mais avançados, suportando sistemas operacionais completos como Linux ou Android e oferecendo poder de processamento superior(ADJIH et al., 2015).

A figura 20 ilustra um microcontrolador utilizado na plataforma IoT-Lab para realização de experimentos.

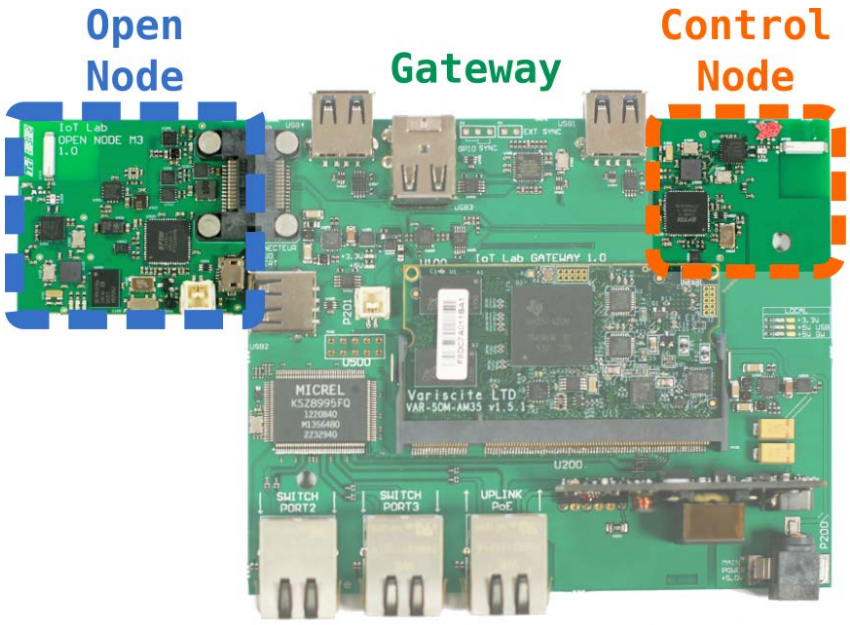

Figura 20 – Nó instalado na plataforma Iot-Lab.

Fonte: https://iot-lab.github.io/docs/getting-started/design/

Sensores Utilizados na IoT-LAB

Os nós de experimentação da IoT-LAB(ADJIH et al., 2015) são equipados com uma variedade de sensores que permitem a coleta de diferentes tipos de dados para os experimentos:

❑ **Nós M3:**

- **Acelerômetro e Magnetômetro Triaxial:** Utilizados para detectar vibrações e variações no campo magnético, capazes de identificar a presença de pessoas ou o movimento de robôs móveis.
- **Sensores de Luz, Pressão e Temperatura:** Coletam dados ambientais variados, essenciais para simulações de condições específicas.

- **Sensores de Localização por RF:** Utilizados para experimentos de localização, permitindo triangulação e rastreamento de dispositivos em movimento.

❑ **Nós A8:**

- **Acelerômetro, Magnetômetro e Giroscópio Triaxial:** Utilizados para monitoramento avançado de movimento e orientação, ideais para análise de movimentos complexos.
- **Módulo GPS:** Proporciona sincronização precisa e localização geográfica para experimentos distribuídos e que requerem precisão de tempo e espaço.
- **Sensores Ambientais:** Como sensores de temperatura, umidade e pressão, para monitoramento ambiental detalhado em experimentos complexos.

Na plataforma FIT IoT-LAB, a configuração do *firmware* do servidor CoAP para os nós A8 envolve uma série de etapas que garantem a comunicação eficiente por meio do protocolo CoAP em uma rede IPv6. Um dos motivos para a escolha dos nós A8 foi a capacidade desses dispositivos de fornecerem sensores ambientais, como sensores de temperatura, umidade e pressão para a consulta e coleta de dados em redes IoT. Inicialmente, um nó A8 é configurado como Roteador de Borda (*Border Router - BR*), que atua como ponte entre a rede IoT e a rede IPv6 pública. O *Border Router - BR* é compilado e carregado nesse nó específico. Em seguida, define-se um prefixo IPv6 exclusivo, como exemplo 2001:660:5307:3100::1/64, garantindo a comunicação correta entre os dispositivos. Posteriormente, o *firmware* do servidor CoAP é carregado nos outros nós A8, utilizando as ferramentas da plataforma IoT-LAB para programar os dispositivos remotamente.

Esses nós, além de fornecerem dados ambientais, estão conectados à rede 6LoWPAN, processando requisições do protocolo CoAP (GET, POST, PUT, DELETE), simulando cenários reais de comunicação em redes de baixa potência e largura de banda. Os testes realizados garantem que o servidor CoAP opera corretamente, assegurando que os nós A8 respondam de maneira eficiente, mesmo em ambientes com recursos limitados. Essa configuração, com o uso dos sensores ambientais dos nós A8, resulta em uma comunicação estável e eficiente, essencial para redes IoT em cenários críticos, como áreas remotas ou propensas a desastres. Assim, o sistema demonstra a capacidade dos nós A8 de integrar-se com sucesso em uma rede IoT federada, contribuindo tanto para o monitoramento contínuo quanto para a detecção de falhas.

5.1.2 Conectividade IPv6 na IoT-LAB

A plataforma IoT-LAB fornece uma infraestrutura avançada de conectividade baseada no protocolo IPv6, que é essencial para a comunicação direta entre dispositivos de IoT de baixa potência e a Internet. Essa conectividade é crucial para o desenvolvimento e teste de novos protocolos e aplicações IoT, assegurando que os dispositivos possam interagir de maneira eficiente tanto entre si quanto com serviços baseados na web.

A Figura 21 ilustra o pool de prefixos IPv6 públicos é utilizado para configurar redes IPv6 que permitem a conectividade global dos dispositivos de teste. Cada site do IoT-LAB, como Grenoble, Lille, Saclay ou Strasbourg, possui um intervalo específico de prefixos /64. Esses prefixos são atribuídos a sub-redes que fornecem endereços IPv6 globais unicast para os dispositivos. Por exemplo, em Grenoble, o intervalo de prefixos vai de 2001:660:5307:3100::/64 até 2001:660:5307:317f::/64, permitindo conectividade direta dos dispositivos com a internet.

Site	Número de sub-redes	Prefixos, de	para
Grenoble	128	2001:660:5307:3100::/64	2001:660:5307:317f::/64
Lille	128	2001:660:4403:0480::/64	2001:660:4403:04ff::/64
Saclay	64	2001:660:3207:04c0::/64	2001:660:3207:04ff::/64
Strasbourg	32	2a07:2e40:fffe:00e0::/64	2a07:2e40:fffe:00ff::/64

Figura 21 – Pool de prefixos IPv6/64 públicos por site - Iot-Lab.
https://www.iot-lab.info/docs/getting-started/ipv6/

Para adaptar o IPv6 às limitações de redes de sensores sem fio de baixa potência, a IoT-LAB utiliza o protocolo 6LoWPAN (*IPv6 over Low Power Wireless Personal Area Networks*). Este protocolo permite que pacotes IPv6 sejam compactados em um formato adequado para transmissão em redes conforme o padrão IEEE 802.15.4, que possuem restrições de tamanho de quadro e capacidade de energia. A adaptação dos pacotes IPv6 para 6LoWPAN é fundamental para manter a interoperabilidade entre dispositivos com diferentes capacidades, permitindo que até mesmo dispositivos com recursos limitados participem de uma rede IP completa, comunicando-se de maneira eficiente e segura através da Internet (ADJIH et al., 2015).

A conectividade IPv6 na IoT-LAB é facilitada pelo uso do protocolo 6LoWPAN, que adapta pacotes IPv6 para um formato que pode ser manejado por redes de sensores sem fio de baixa potência. Esta adaptação é essencial devido à limitação do tamanho dos quadros de dados que podem ser transmitidos por essas redes, os quais são significativamente menores que os pacotes IPv6 convencionais. O 6LoWPAN permite que dispositivos IoT funcionem de maneira eficiente em redes com largura de banda limitada e baixa capacidade de processamento, mantendo a compatibilidade com o protocolo IPv6 (ADJIH et al., 2015).

- **Roteador de Borda (*Border Router - LBR*):** O *Low-power Border Router (LBR)* desempenha um papel crucial na arquitetura de rede da IoT-LAB. Ele funciona como um ponto de transição entre a rede de dispositivos IoT e a Internet, executando a compactação dos cabeçalhos IPv6 para o formato 6LoWPAN. Além disso, o LBR é responsável pelo encaminhamento de pacotes de entrada e saída, garantindo que os dados transmitidos entre os dispositivos IoT e a Internet sejam compactados e descompactados conforme necessário para uma comunicação eficiente. Este processo permite que dispositivos de IoT se comuniquem diretamente com servidores na Internet, mantendo uma comunicação segura e eficaz.

- **Interfaces Virtuais (tun e tap):** Para integrar a rede IoT com a Internet, o LBR utiliza interfaces de rede virtuais, como tun e tap. Essas interfaces criam conexões de rede virtualizadas, que permitem capturar e manipular pacotes antes de serem encaminhados para os dispositivos de destino ou para a Internet. No caso do OpenWSN, o LBR é implementado em um computador que se comunica com o nó raiz do DAG através de uma porta serial. O software OpenVisualizer é usado para gerenciar esta conexão, realizando a compactação de pacotes IPv6 para 6LoWPAN e transmitindo-os pela rede. Esse mecanismo garante que a comunicação ocorra de forma fluida e sem interrupções.

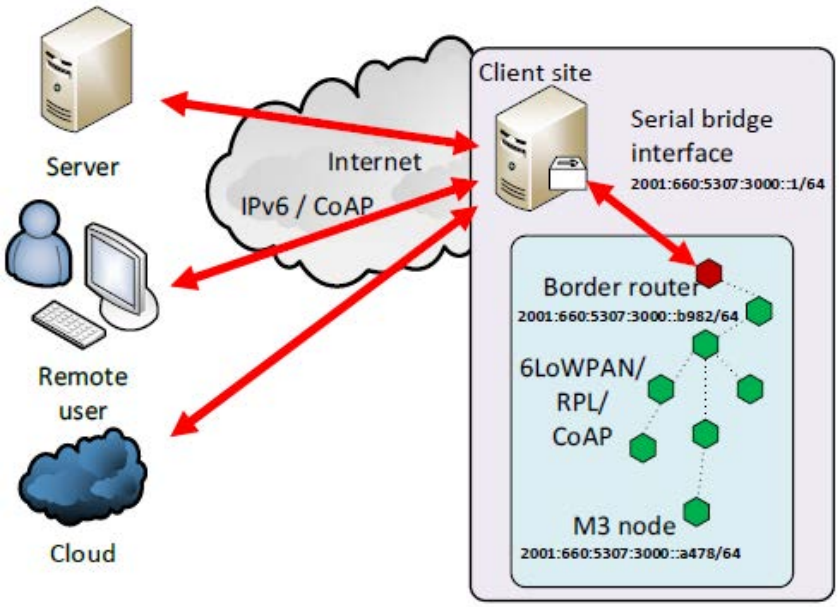

Figura 22 – Conectividade Ipv6 - Iot-Lab.

Fonte: (ADJIH et al., 2015)

A Figura 22 ilustra a arquitetura de conectividade IPv6 na IoT-LAB, destacando como dispositivos IoT, como os nós M3, se conectam à Internet e se comunicam entre si. O roteador de borda (LBR) é fundamental nessa arquitetura, atuando como um ponto de interconexão que não só conecta a rede IoT à Internet, mas também facilita a comunicação direta entre os dispositivos IoT, convertendo os cabeçalhos IPv6 para o formato 6LoWPAN. A figura também demonstra o fluxo de dados entre os nós M3 e a infraestrutura de *backend* da IoT-LAB, utilizando interfaces de rede virtuais (tun e tap) para criar conexões virtualizadas, assegurando uma comunicação eficiente e integrada.

5.1.3 Dispositivos por Região

Até o término deste trabalho, a quantidade de dispositivos disponíveis para testes é apresentada na Tabela 3. As regiões de Grenoble, Lille, Saclay e Strasbourg eram as que possuíam uma faixa de *pools* disponíveis para testes, conforme ilustrado na Figura 21. Essas regiões foram selecionadas devido à sua disponibilidade e adequação às necessidades de testes, enquanto outras regiões, como Paris e Toulouse, não apresentavam *pools* suficientes ou disponíveis no momento.

Site	Número de Nós
Grenoble	620
Lille	270
Paris	131
Saclay	199
Strasbourg	260
Toulouse	43

Tabela 3 – Tabela de números de nós por site.

Dessa forma, a escolha dos locais foi baseada na acessibilidade e no suporte oferecido para a execução dos testes necessários. No entanto, mesmo com as regiões de Grenoble, Lille, Saclay e Strasbourg disponíveis para experimentos, conforme apresentado na seção 5.2, apenas Grenoble e Strasbourg possuíam conectividade de mão dupla para realizar o experimento com monitoramento federado entre regiões, conforme tabela 4 (resumo de comunicação entre regiões).

5.2 Definição das Regiões Monitoradas

Os experimentos conduzidos na plataforma IoT-Lab visavam estabelecer uma infraestrutura de monitoramento federado entre as regiões de Grenoble, Strasbourg, Lille e Saclay. O objetivo central era garantir uma comunicação bidirecional entre estas regiões para criar um sistema de monitoramento robusto e colaborativo. Entretanto, barreiras inerentes à arquitetura da rede IoT-Lab, especificamente as políticas de roteamento e configurações de *firewall*, emergiram como desafios críticos para a realização dessa meta. A Figura 23 ilustra uma falha de comunicação entre as regiões de Grenoble e Lille, esta representação visual destaca os pontos específicos onde as interrupções ocorreram.

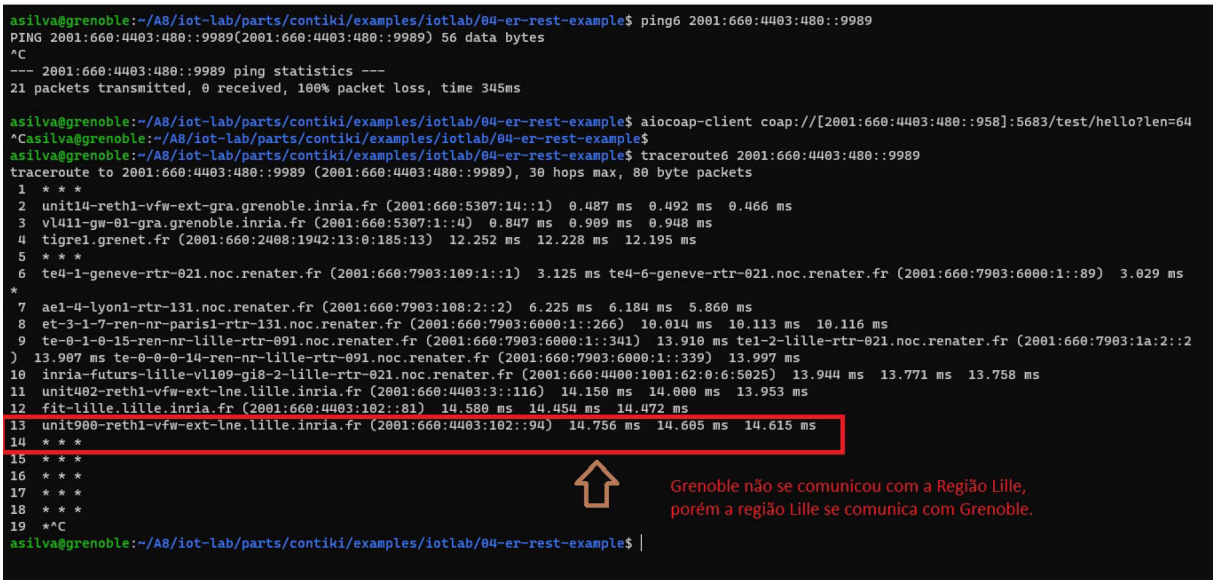

Figura 23 – Exemplo de Falha de Comunicação entre Regiões (Grenoble e Lille).

Tentativas de estabelecer uma conexão usando o protocolo CoAP falharam, como demonstrado pelos testes de `ping` e `traceroute`. O *ping* não recebeu resposta após enviar 21 pacotes, indicando uma perda total. O *traceroute*, por sua vez, começou a mapear a rota dos pacotes, mas não conseguiu completar o caminho para o destino, conforme mostram os asteriscos na saída. Esses resultados evidenciam uma ruptura na comunicação entre as duas regiões.

Os testes de conectividade realizados entre as regiões de Grenoble, Strasbourg, Lille e Saclay demonstraram que apenas Grenoble e Strasbourg mantiveram uma comunicação bidirecional bem-sucedida, um elemento crítico para a implementação de um sistema de monitoramento federado. Para evitar redundância e manter a clareza, optou-se por não incluir as representações visuais de todos os testes, dado que os resultados foram consistentes com aqueles ilustrados na Figura 23.

Em vez disso, será apresentada a Tabela 4 que ilustra o resumo destacando a conectividade entre as regiões mencionadas, simplificando a visualização do desempenho da rede no contexto do nosso estudo.

A distinção clara das regiões com capacidades de comunicação bem-sucedida é crucial para a continuidade dos testes e a evolução contínua do algoritmo de monitoramento. Tal discernimento é indispensável para concentrar os esforços de desenvolvimento de maneira direcionada e eficiente, evitando a dispersão de recursos em tentativas de solucionar problemas de envio de requisições CoAP que, em análise mais aprofundada, poderiam ser incorretamente atribuídos a falhas operacionais ao invés de limitações estruturais da rede.

Tabela 4 – Resumo de Comunicação entre Regiões.

Regiões	Status da Comunicação
Grenoble ↔ Strasbourg	Bidirecional
Lille → Grenoble	Unidirecional (Lille para Grenoble)
Lille → Strasbourg	Unidirecional (Lille para Strasbourg)
Saclay → Grenoble	Unidirecional (Saclay para Grenoble)
Saclay → Strasbourg	Unidirecional (Saclay para Strasbourg)
Lille × Saclay	Falha em ambas regiões
Grenoble ← Lille	Unidirecional (apenas Lille para Grenoble)
Grenoble ← Saclay	Unidirecional (apenas Saclay para Grenoble)
Strasbourg ← Lille	Unidirecional (apenas Lille para Strasbourg)
Strasbourg ← Saclay	Unidirecional (apenas Saclay para Strasbourg)

Por conseguinte, a identificação precisa das áreas de comunicação eficaz e as restrições impostas pela infraestrutura da plataforma IoT-Lab são componentes essenciais para a otimização do sistema. Estes *insights* permitem a implementação de estratégias focadas na melhoria da confiabilidade e da eficiência do monitoramento, assegurando que o sistema opere dentro das capacidades e limitações conhecidas da rede.

5.3 Definição da Margem de Segurança (β)

Neste estudo efetuado na plataforma IoT-Lab (ADJIH et al., 2015), o principal objetivo foi a definição de uma margem de segurança robusta para o cálculo do tempo de chegada previsto dos *heartbeats*. Este parâmetro é essencial para assegurar que haja uma tolerância apropriada para as variações inerentes nas comunicações de rede.

Portanto, foram realizados experimentos nas regiões de Grenoble e Strasbourg, cobrindo um período contínuo de 24 horas. Os testes incluíram quatorze dispositivos: nós de 0 a 9 representando os sensores localizados na própria Região; o nó 10, atuando como o monitor da Região vizinha; e os nós 11, 12 e 13, representando sensores remotos ou seja da região adjacente.

Esta pesquisa concentra-se na mensuração do tempo de *ping*, que são os intervalos medidos desde o momento de envio de uma requisição pelo sistema de monitoramento, designado *Disaster-FD*, até a recepção de uma resposta do dispositivo em análise.

Os resultados revelaram uma diversidade nos tempos de resposta, ilustrando as variações e as flutuações na performance da rede de dispositivos IoT. A discrepância observada nos tempos de resposta salientou a necessidade de estabelecer uma margem de segurança que pudesse compensar as incertezas e assegurar a detecção fiel do estado operacional dos dispositivos sem incorrer em alarmes falsos.

Os experimentos conduzidos nas regiões de Grenoble e Strasbourg resultaram em desvios padrão dos tempos de *ping* de 1.233 milissegundos e 1.187 milissegundos, respectivamente, conforme ilustrado na Figura 24. Com base nestes resultados, optou-se por definir uma margem de segurança de 1.500 milissegundos para os procedimentos de teste futuros.

Esta margem foi considerada suficientemente conservadora para absorver as variações notadas nos tempos de *ping*, evitando assim a classificação antecipada de atrasos nos *heartbeats* como falhas no sistema. Esta medida preventiva é de significativa importância no domínio dos dispositivos IoT, onde a integridade da comunicação está sujeita a influências externas, como flutuações na carga da rede e diversidades nas condições ambientais.

Região Grenoble		
Dispositivos	**Média de Ping (ms)**	**DesvPad de Ping (ms)**
Node - 0	70	124
Node - 1	159	184
Node - 2	399	719
Node - 3	1218	1211
Node - 5	1602	1206
Node - 6	2527	499
Node - 7	2530	489
Node - 8	2557	442
Node - 9	2465	697
Node - 10	85	87
Node - 11	383	731
Node - 12	275	566
Node - 13	968	1097
Valores	**1157**	**1233**

Região Strasbourg		
Dispositivos	**Média de Ping (ms)**	**DesvPad de Ping (ms)**
Node - 0	98	101
Node - 1	194	139
Node - 2	599	821
Node - 4	910	1094
Node - 5	1883	1102
Node - 6	1783	1108
Node - 7	2258	863
Node - 8	2473	562
Node - 9	2517	482
Node - 10	90	45
Node - 11	118	260
Node - 12	331	622
Node - 13	578	907
Valores	**1050**	**1187**

Figura 24 – Média e Desvio Padrão (*Pings dos Dispositivos*).

5.4 Consumo de Energia

5.4.1 Energia IoT: Variação por Tamanho e Intervalo de Envio das mensagens

Para explorar o impacto do consumo de energia em dispositivos IoT, o estudo analisou duas variáveis críticas: a frequência de envio das mensagens e o tamanho delas. Utilizando a plataforma IoT-Lab, foram conduzidos experimentos com três dispositivos que enviavam requisições GET via protocolo CoAP. As mensagens tinham tamanhos variando entre 10, 100 e 1.000 caracteres, enquanto as frequências de envio oscilavam entre 100 e 1.000 milissegundos. O objetivo era compreender como esses fatores influenciam o consumo energético dos dispositivos.

A Figura 25 ilustra o consumo de energia dos dispositivos em relação ao tamanho das mensagens e intervalo de envio, é observado uma tendência clara: o consumo de energia se eleva com o aumento do tamanho das mensagens. No entanto, o intervalo de envio surge como o fator mais determinante nesse consumo. Para intervalos de 100 milissegundos, independentemente do tamanho da mensagem, o consumo foi consistentemente mais alto comparado aos intervalos de 1.000 milissegundos. Isso sugere que a frequência de envio das mensagens exerce uma influência mais acentuada no consumo de energia do que o tamanho da mensagem.

De forma mais detalhada, as mensagens de 10 caracteres enviadas a cada 100 milissegundos consumiram aproximadamente uma média de 0,16793 de energia, enquanto aquelas enviadas a cada 1.000 milissegundos consumiram em torno de 0,16585. Este padrão se mantém quando foi aumentado o tamanho da mensagem para 100 e 1.000 caracteres. Esses dados destacam que, embora exista um incremento no consumo proporcional ao tamanho das mensagens, o impacto da frequência de envio é mais significativo, principalmente quando se considera a eficiência energética em dispositivos IoT.

A partir dos resultados obtidos, infere-se que a otimização da frequência de envio de mensagens pode resultar em uma gestão de energia mais eficiente. Isso é especialmente relevante para a implementação em larga escala de dispositivos IoT, onde a economia de energia é imprescindível.

5.4.2 Energia IoT: Variação pelo Intervalo de Envio das mensagens

Em uma investigação sobre o impacto do consumo de energia em dispositivos IoT, foi adotada uma metodologia de avaliação centrada no intervalo de envio de requisições, um fator determinante para o consumo de energia. Utilizando a plataforma IoT-Lab, os experimentos foram conduzidos em três dispositivos distintos, cada um configurado para emitir requisições GET via protocolo CoAP.

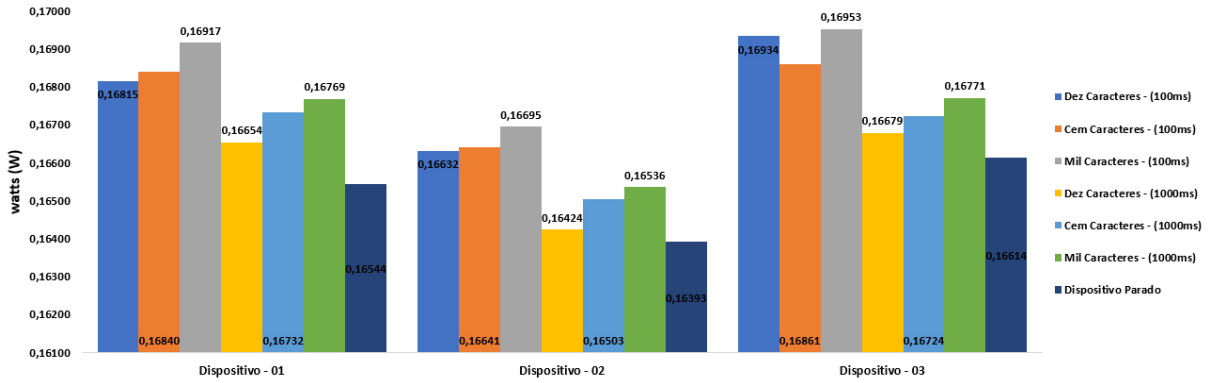

Figura 25 – Consumo Energético por Tamanho e Intervalo das Mensagens.

A Figura 26 ilustra o consumo de energia em diferentes intervalos de envio, onde foram observados os seguintes resultados. A figura ilustra que o consumo de energia varia significativamente em intervalos que vão de 10 milissegundos a 60.000 milissegundos. Os dados apontam para um consumo de energia marcadamente maior nos intervalos mais curtos, notadamente entre 10 milissegundos e 1.000 milissegundos. Em contrapartida, observa-se que, a partir de 5.000 milissegundos, o consumo de energia apresenta tendência à estabilização, assemelhando-se aos valores registrados quando os dispositivos se encontram em estado de repouso.

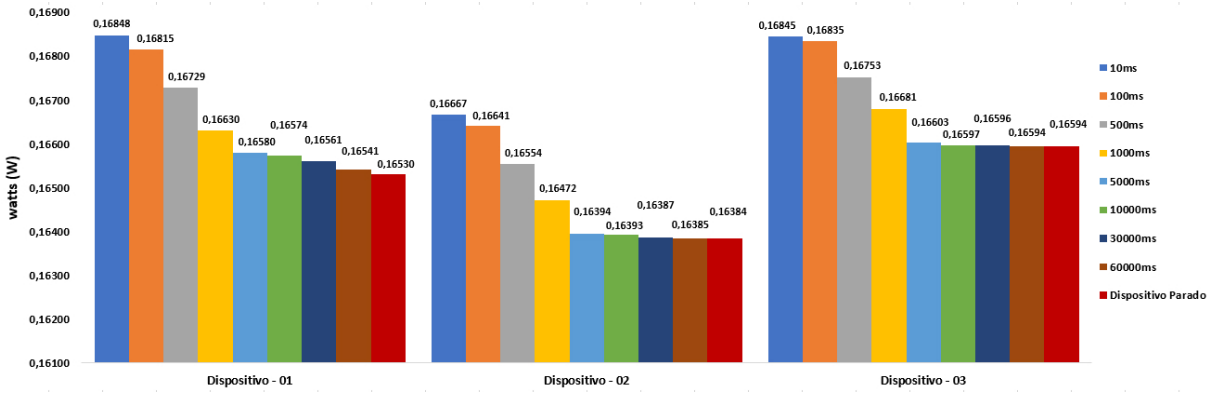

Figura 26 – Consumo Energético por Intervalo de Envio das Mensagens.

Este padrão indica que um intervalo de envio de 5.000 milissegundos pode representar um equilíbrio adequado, atenuando o consumo de energia sem comprometer a frequência de atualizações, sinalizando um elemento vital para dispositivos operando em ambientes onde o acesso à energia é restrito ou a troca de baterias é inviável em intervalos curtos, ao expandirmos a análise para intervalos mais extensos, como 10.000 milissegundos, 30.000 milissegundos e 60.000 milissegundos, observa-se que o consumo de energia se mantém próximo ao cenário de 5.000 milissegundos. Entretanto, esses intervalos maiores podem ser contraproducentes em contextos que requerem respostas ágeis e atualizações de dados frequentes.

Conclui-se, portanto, que o intervalo de envio de 5.000 milissegundos se destaca como a escolha mais equilibrada para os dispositivos avaliados na plataforma IoT-Lab. Este intervalo não apenas favorece a redução do consumo energético mas também mantém uma eficácia operacional adequada. Tal equilíbrio é fundamental, permitindo o controle do gasto energético enquanto se preserva a frequência necessária para as comunicações, aspecto crucial para a maioria das implementações práticas em cenários de energia limitada, ajustar o intervalo de envio das mensagens é, portanto, imperativo para otimizar a eficiência e a sustentabilidade dos dispositivos IoT, garantindo a longevidade e a confiabilidade necessárias em aplicações do mundo real.

5.5 Problemas Identificados na Implementação

5.5.1 Problema na contagem de erros em dispositivos IoT

Durante o experimento de 24 horas com o sistema de detecção de falhas *Disaster-FD*, foram detectadas anomalias no registro de erros e acertos dos dispositivos IoT monitorados. Observou-se que, em certos momentos, a falta de resposta às requisições CoAP não era registrada como um erro nem confirmada por uma resposta bem-sucedida. Essa ausência de retorno induzia os dispositivos a um estado indeterminado, referido neste estudo como "estado de limbo". Nesse estado, não havia registro de recebimento no arquivo `Log_CoAP.csv` nem de falhas no arquivo `contadorErrosDispositivo.csv`, como seria tipicamente esperado.

Nesse estado, os dispositivos não apresentavam sinais claros de operação normal ou de falha, contradizendo a lógica do *Disaster-FD*, que prevê um retorno explícito **(sucesso ou erro)**, para cada requisição CoAP do tipo GET. Aprofundando a análise no código do *Disaster-FD*, foi identificado que o problema poderia estar relacionado à inicialização das *threads* atribuídas a cada dispositivo, foi constatado que em algumas situações as *threads* não eram corretamente ativadas, afetando a eficácia do monitoramento.

Como ação corretiva, adaptou-se o algoritmo do sistema *Disaster-FD*, introduzindo um mecanismo de supervisão ativa das *threads* esta implementação tem o objetivo de realizar verificações periódicas do estado de cada *thread*, garantindo sua ativação correta e evitando paralisações inesperadas no processo.

A Tabela 5, apresenta um resumo dos *logs* de monitoramento das *threads* em Strasbourg. As informações confirmam interrupções súbitas nas atividades das *threads*. Com o aprimoramento do monitoramento, foi possível eliminar o fenômeno anteriormente observado de *threads* em "estado de limbo". Esta medida revelou que o cerne do problema residia na robustez do monitoramento das *threads*, e não na funcionalidade dos próprios dispositivos IoT.

A implementação da estratégia de monitoramento de *threads* confirmou sua eficácia, garantindo a operacionalidade contínua das *threads* e o correto funcionamento do sistema *Disaster-FD*. Essa melhoria eliminou a incerteza sobre o estado dos dispositivos, aumentando a confiabilidade do processo de monitoramento.

Tabela 5 – Estado das Threads em Strasbourg: Extrato dos Logs.

Data e Hora da Verificação	Thread ID	Estado
03/01/2024 01:32:51	coap://[2a07:2e40:fffe:e0::a984]:5683/sensors/gyros	Ativa
03/01/2024 01:32:51	coap://[2a07:2e40:fffe:e0::a488]:5683/sensors/gyros	Ativa
03/01/2024 01:33:51	coap://[2a07:2e40:fffe:e0::a984]:5683/sensors/gyros	**Inativa**
03/01/2024 01:33:51	coap://[2a07:2e40:fffe:e0::a488]:5683/sensors/gyros	**Inativa**
03/01/2024 01:34:51	coap://[2a07:2e40:fffe:e0::a984]:5683/sensors/gyros	Ativa
03/01/2024 01:34:51	coap://[2a07:2e40:fffe:e0::a488]:5683/sensors/gyros	Ativa
03/01/2024 15:10:51	2001:660:5307:3100::9181	**Inativa**
03/01/2024 15:11:51	2001:660:5307:3100::9181	Ativa
03/01/2024 15:58:51	2001:660:5307:3100::9181	**Inativa**
03/01/2024 15:59:51	2001:660:5307:3100::9181	Ativa

5.5.2 Problema no Intervalo de Envio das Requisições

Durante o aprimoramento do algoritmo para o sistema de detecção de falhas *Disaster-FD*, com o objetivo de otimizar o envio de mensagens para os dispositivos IoT, enfrentou-se desafios. Um deles foi o envio de requisições GET utilizando o protocolo CoAP em intervalos fixos de 5.000 milissegundos.

Entretanto, a análise da Tabela 6 a seguir, ilustra a ocorrência de inconsistências temporais durante o processo de envio de requisições, observa-se que as mensagens com ID (2 e 3) e (5 e 6) foram expedidas de maneira concomitante para o dispositivo 1, desconsiderando o intervalo predefinido de 5.000

milissegundos. Em contraste, as mensagens (2, 4 e 5) mostraram intervalos aparentemente arbitrários. Adicionalmente, a mensagem de ID 1 está sem um intervalo de envio especificado, dado que esta representa a requisição inicial do conjunto de dados. A conformidade esperada no intervalo de envio de mensagens não foi observada, conforme detalhado na tabela mencionada, exigindo uma investigação adicional para discernir as causas dessas anomalias.

Tabela 6 – Erro no Intervalo de Envio das Requisições.

Dispositivo	Mensagem	*Timestamp* Envio(ns)	*Timestamp* Recebimento(ns)	Envio(ms)
1	1	23537899039209092	23537899950775277	*****
1	2	23537899967261588	23537899998478815	928
1	3	23537899967261588	23537902638222651	0
1	4	23537906019306704	23537906510133414	6052
1	5	23537907185614502	23537907725877106	1166
1	6	23537907185614502	23537917885450369	0

Este desafio ocasionou uma necessária revisão estrutural do sistema, inicialmente o código-fonte estava integralmente alocado dentro do método `main`, configurando uma estrutura monolítica e inflexível. Tal organização comprometia diretamente a clareza, a manutenibilidade e a testabilidade do sistema. Visando superar essas limitações, foi realizado uma refatoração do código, o processo concentrou-se, primordialmente, no trecho de código responsável pelas requisições CoAP, que foi cuidadosamente separado do método principal e realocado em unidades modulares específicas.

Esta reorganização não apenas favoreceu a modularidade e a eficiência do sistema, mas também emergiu como um passo decisivo para a identificação e resolução de um problema crítico: a utilização do método `ping()` da biblioteca Californium. Este método, inicialmente empregado para verificar a disponibilidade dos dispositivos alvo antes da transmissão das requisições, introduzia uma latência, comprometendo assim a temporalidade essencial das requisições CoAP.

A análise e ajuste desta funcionalidade foi, portanto, fundamental para a otimização do desempenho do sistema, assegurando o cumprimento adequado dos prazos estipulados para envio das requisições.

A abordagem de *threading* implementada, embora eficaz ao alocar *threads* distintas para o envio das requisições de cada dispositivo, não isolava completamente as requisições individuais dentro de cada *thread* dedicada. Isso resultava em uma latência cumulativa, já que a espera por uma resposta do *ping* dentro de uma *thread* específica provocava atrasos em toda a sequência de emissão de requisições.

A solução emergiu com a eliminação do fragmento de código vinculado ao método `ping()`, removendo assim a dependência da confirmação de disponibilidade antes da transmissão das requisições. Tal ajuste erradicou a latência e permitiu que as requisições CoAP fossem enviadas dentro do intervalo estipulado. Além disso, percebeu-se que o método `client.setTimeout()` da biblioteca Californium, já estabelecia um limite temporal para aguardar as respostas das requisições CoAP. Este método se mostrou um recurso eficaz para administrar as respostas perdidas ou atrasadas, descartando a necessidade da verificação preliminar pelo *ping*.

Consequentemente, a introdução do método `client.setTimeout()`, conforme resultado apresentado na Tabela 7, ilustra as requisições CoAP que passaram a ser enviadas dentro do intervalo de tempo pré-determinado de 5.000 milissegundos, assegurando a operacionalidade do sistema *Disaster-FD* dentro dos parâmetros temporais projetados.

5.5.3 Problema com Valores Negativos na Métrica P.A.

Nos testes realizados para coletar e analisar o tempo de chegada dos *heartbeats* emitidos pelos dispositivos IoT, essenciais para determinar a confiabilidade da rede monitorada pelo sistema de detecção de

Tabela 7 – Intervalo de Envio das Requisições CoAp após Correção.

Dispositivo	Mensagem	*Timestamp* Envio(ns)	*Timestamp* Recebimento(ns)	Envio(ms)
1	1	22854346795459578	22854347373210896	*****
1	2	22854351711986909	22854352572718459	4917
1	3	22854356712006264	22854357164815839	5000
1	4	22854361711868379	22854362844168546	5000
1	5	22854366711861127	22854367260440847	5000
1	6	22854371711858370	22854372522808483	5000
1	7	22854376711836970	22854377898414791	5000
1	8	22854381711709226	22854382489989902	5000
1	9	22854386711835480	22854388105909452	5000

falhas *Disaster-FD*, foi identificado um comportamento atípico nos resultados estatísticos.

A métrica de Probabilidade Acurácia (P.A.), que mensura a precisão na identificação de falhas de rede, registrou valores negativos conforme ilustrado na Tabela 8 a seguir, esta situação é conceitualmente inconsistente, uma vez que a P.A. deveria variar estritamente de zero a um. Tal discrepância foi notada somente na região de Strasbourg, não sendo evidenciada em Grenoble, apesar de ambas as regiões utilizarem o mesmo algoritmo de detecção de falhas e terem sido iniciadas simultaneamente.

Tabela 8 – Análise Estatística da Região Strasbourg.

Janela	Margem (β)	Limiar	Nível de Confiança	Erro	Tempo do Erro	P.A.
100	1500ms	50	70	1	558985883	0.9905
100	1500ms	50	60	2	3101972596	0.9983
100	1500ms	50	60	6	3095521805214	0.1542
100	1500ms	50	60	7	3097197816048	0.4327
100	1500ms	50	60	15	14460445480456	**-0.5961**
100	1500ms	50	50	39	31367774326042	**-1.8884**
100	1500ms	50	70	43	31373676759629	**-1.4783**
100	1500ms	50	40	51	31379649826290	**-1.1701**
100	1500ms	50	50	296	32867148946217	**-1.0214**

Após análise do código-fonte do *Disaster-FD*, foi diagnosticado que a raiz do problema era um caso clássico de condição de corrida, onde múltiplas *threads* acessavam e modificavam simultaneamente um recurso compartilhado sem a devida sincronização. Em sistemas paralelos, o gerenciamento do acesso concorrente a objetos compartilhados é vital para manter a integridade dos dados. A falta de sincronização pode resultar em leituras e escritas inconsistentes, levando a estados indesejáveis no programa.

O mecanismo de sincronização em Java, implementado através do modificador *synchronized*, fornece uma solução para esse desafio de concorrência. Quando um método é declarado como *synchronized*, ele estabelece um bloqueio (*lock*) sobre o objeto que controla o acesso à seção crítica do código, assegurando que, em um dado momento, somente uma *thread* possa executar o método sincronizado. Se múltiplas *threads* tentam invocar métodos sincronizados no mesmo objeto, as tentativas subsequentes são forçadas a esperar, criando uma fila que respeita a ordem de chamada. Isso impede que duas operações entrem em conflito, preservando a consistência dos valores e evitando resultados errôneos como os observados.

A aplicação do modificador *synchronized* ao método `execute()` emergiu como uma solução, conferindo atomicidade às operações de atualização e prevenindo o acesso concorrente a recursos compartilhados.

A Tabela 9 ilustra a eficácia desta intervenção, pois com a sincronização efetiva, a integridade dos dados foi assegurada, resultando em valores positivos consistentes para a métrica P.A. na região Strasbourg. Estes resultados positivos corroboram a adequação do algoritmo em refletir com precisão a confiabilidade dos dispositivos IoT nas regiões de Grenoble e Strasbourg, como pretendido inicialmente, a implementação

do *synchronized* trouxe estabilidade e confiabilidade ao sistema, garantindo que as métricas refletissem o verdadeiro estado operacional da rede monitorada.

Tabela 9 – Análise Estatística da Região Strasbourg após Correção.

Janela	Margem (β)	Limiar	Nível de Confiança	Erro	Tempo do Erro	P.A.
100	1500ms	50	60	1	187234780	0.9968
100	1500ms	50	60	4	21005797961	0.9887
100	1500ms	50	60	7	33216079114	0.9909
100	1500ms	50	50	24	56441152142	0.9896
100	1500ms	50	60	34	65944450088	0.9909
100	1500ms	50	50	39	79963786486	0.9911
100	1500ms	50	50	57	116733802108	0.9892
100	1500ms	50	50	63	149587833338	0.9881
100	1500ms	50	50	70	153364776104	0.9893
100	1500ms	50	50	83	169872314578	0.9895

5.6 Experimentos

Os experimentos realizados têm como objetivo testar o sistema de detecção de falhas *Disaster-FD* no contexto da Internet das Coisas (IoT), mais especificamente no ambiente FIT IoT-LAB (ADJIH et al., 2015). A configuração das regiões de Grenoble e Strasbourg é similar ao exemplo da Figura 10. Além disso, a decisão de monitorar somente as regiões de Grenoble e Strasbourg baseou-se em limitações de conectividade observadas com outras áreas da infraestrutura, provavelmente devido a configurações de *firewall*, conforme explorado na Seção 5.2.

O protocolo CoAP foi empregado para comunicação com os sensores e ICMP para monitoramento de outros dispositivos. As requisições CoAP foram definidas para um intervalo de 5.000 milissegundos, baseado nos resultados sobre consumo de energia na seção 5.4.2, e a margem de segurança (β) de 1.500 milissegundos foi estabelecida com base em testes de latência descritos na seção 5.3. A distribuição dos fatores de impacto foi estrategicamente configurada para refletir a importância relativa dos dispositivos e processos monitorados. Dessa forma o experimento envolveu o monitoramento de 14 processos em cada região, incluindo 10 sensores locais, 3 em regiões adjacentes e um monitor na região vizinha. Os sensores locais receberam um fator de impacto de 10, enquanto os adjacentes, 20, e o monitor na região vizinha, considerado crítico, recebeu um fator de 60.

Assim, o nível de confiança teórico máximo foi estipulado em 220 ($10 \times 10 + 3 \times 20 + 60$). Este estudo não apenas verificou a eficiência do *Disaster-FD* na detecção de falhas em tempo real em um ambiente IoT, mas também investigou o comportamento das interações entre dispositivos distribuídos por regiões federadas.

A avaliação destacou a assertividade do sistema em contextos federados de IoT, realçando sua capacidade de monitoramento e eficiência operacional, mesmo frente a desafios de infraestrutura de rede. O estudo ressaltou a importância de um ecossistema IoT interconectado e resiliente, capaz de se adaptar e responder prontamente a falhas e interrupções inesperadas.

É importante destacar que, durante o experimento, não foram inseridas falhas artificiais nos dispositivos ou na rede. O experimento foi executado de forma ininterrupta durante 24 horas, garantindo a observação dos comportamentos reais do sistema e sua eficiência em condições normais de operação.

5.6.1 Erros Acumulados por Dispositivo em Strasbourg

A Figura 27 ilustra o perfil de erros acumulados por cada dispositivo ao longo de um ciclo completo de 24 horas na região de Strasbourg. Os dispositivos enumerados de 0 a 9 são identificados como sensores

nativos dessa região e, explicitamente, os dispositivos 5 e 7 mantiveram um registro isento de falhas, onde o detector não registrou falhas para esses dispositivos, sublinhando a robustez da infraestrutura local de Strasbourg.

Os demais dispositivos IoT (0, 1, 2, 3, 4, 6, 8 e 9) registraram poucos erros, reforçando a percepção de uma rede local estável. Por outro lado, a análise da região adjacente de Grenoble, representada pelos dispositivos 11 a 13, além do dispositivo 10, revela uma realidade oposta.

O padrão de erros significativos observado nesses dispositivos indicam uma instabilidade nessa área, especificamente, o dispositivo 10, que desempenha um papel crucial como o nó central ou roteador de borda (monitor), sendo fundamental para a gestão do tráfego de rede em Grenoble.

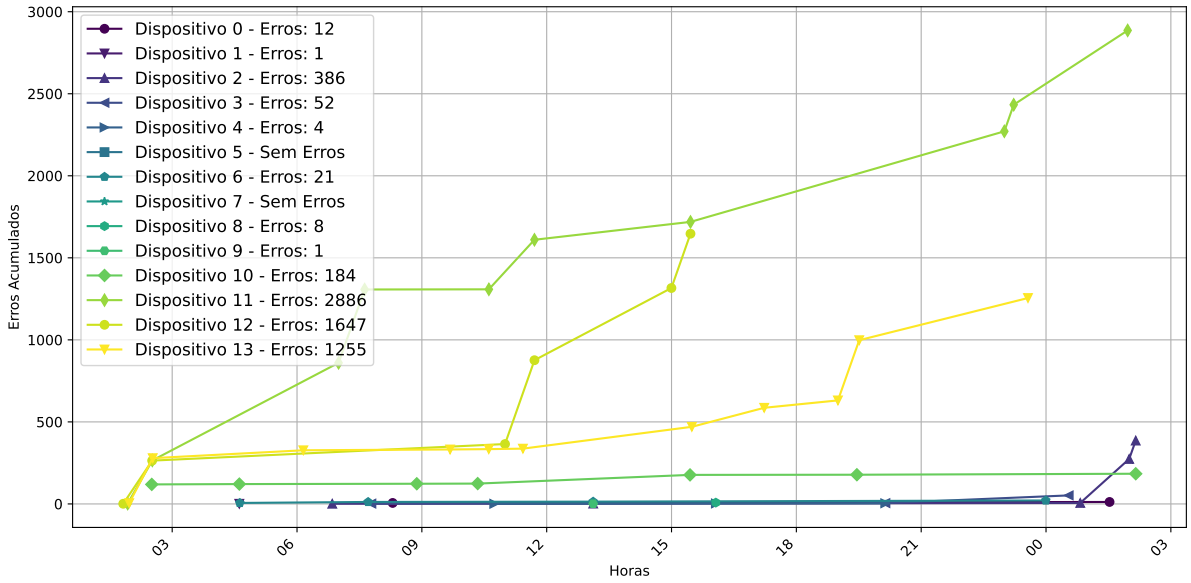

Figura 27 – Erros Acumulados por dispositivo na Região de Strasbourg.

A incidência elevada de erros no dispositivo 10 não apenas captura a atenção para a vulnerabilidade da região, mas também demostra preocupações sobre o funcionamento ininterrupto da rede e de atender devidamente às solicitações processadas pelo sistema de monitoramento *Disaster-FD*.

Dessa forma, a estabilidade comprometida do dispositivo 10 emerge como um fator determinante, potencialmente catalisador de interrupções e degradação da performance da rede IoT em Grenoble, causando interrupções no funcionamento e falhas na resposta às requisições enviadas pelo monitor *Disaster-FD*.

5.6.2 Erros Acumulados por Dispositivo em Grenoble

De forma similar ao observado para Strasbourg, a Figura 28 exibe a distribuição de erros acumulados por dispositivo durante o período de monitoramento de 24 horas na região de Grenoble. A análise dos dispositivos, identificados de 0 a 9, mostra um cenário com um aumento significativo na frequência de erros, evidenciando uma instabilidade na rede local de Grenoble.

Os dispositivos 11 a 13, monitorados pelo detector de falhas *Disaster-FD* e correspondentes aos sensores localizados em Strasbourg, registraram um número reduzido de erros. Em contrapartida, o dispositivo 10, que serve como o monitor na região de Strasbourg, refletiu uma robustez comparativamente maior, com um registro mínimo de falhas. Este contraste nos resultados reforça as descobertas previamente discutidas na Seção 5.6.1 onde abordamos a região Strasbourg, apontando para um desempenho assimétrico entre as duas regiões.

O monitoramento cruzado entre Strasbourg e Grenoble capturado nas Figuras 27 e 28, respectivamente, evidencia uma disparidade significativa entre as condições da rede IoT em cada localidade.

Especificamente, o monitor de Strasbourg identificou uma propensão à instabilidade em Grenoble, enquanto o monitoramento de Grenoble constatou alta incidência de erros em sua própria rede (dispositivos de 0 a 9) e estabilidade na região de Strasbourg (dispositivos de 10 a 13).

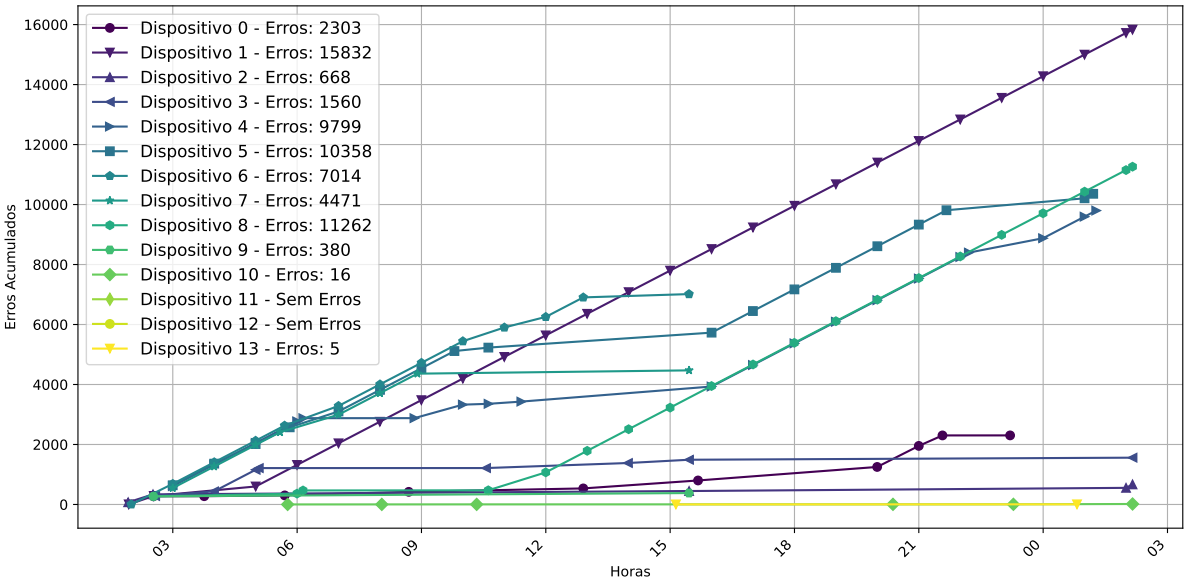

Figura 28 – Erros Acumulados por dispositivo na Região de Grenoble.

Os resultados obtidos indicam que o *Disaster-FD* é capaz de capturar com eficácia as variações entre regiões federadas, fornecendo uma base sólida para futuras investigações com o objetivo de otimizar a detecção e a resiliência das redes frente às falhas.

5.6.3 Estatística da Rede em Strasbourg

A Figura 29 apresenta uma visão estatística da rede em Strasbourg, onde a interação entre a Probabilidade de Acurácia (P.A.) e o Nível de Confiança é monitorada ao longo de 24 horas. O estabelecimento de um limiar de segurança ou *Threshold* em 160, debatido na Seção 4.1.3 e explicitado na Tabela 2, foi pensado para representar um ideal operacional para Strasbourg e respostas adequadas de Grenoble. Observa-se que a Probabilidade Acurácia (P.A.), simbolizada pela linha azul, permanece consistentemente acima de 95% após as 8:50, evidenciando a precisão do sistema *Disaster-FD* na vigilância do estado da rede. A Figura 27 complementa essa análise ao ilustrar os Erros Acumulados por dispositivo em Strasbourg, com um foco especial nos dispositivos 10 a 13 da região de Grenoble, com fatores de impacto de 60, 20, 20 e 20, respectivamente.

Em paralelo, a linha vermelha que retrata o Nível de Confiança demonstra uma volatilidade que reflete os eventos na rede, identificando os momentos de estabilidade e instabilidade. Quando essa linha vermelha cruza para baixo do limiar verde, é indicativo de uma diminuição na confiança na rede de Strasbourg, uma observação corroborada pelo registro de erros nos dispositivos 10 a 13, localizados na região de Grenoble, conforme relatado na Figura 27. O declínio na linha vermelha abaixo do Nível de Confiança ocorreu entre 15:50 e 16:50, sugerindo um evento adverso que afeta a percepção da confiabilidade da rede.

Isso implica que o pico de erros nos dispositivos adjacentes está diretamente ligado à percepção de confiabilidade da rede e pode ser um indicador precoce de um evento adverso na região de Strasbourg, esse cenário corrobora a interdependência entre as redes de Strasbourg e Grenoble. Essa correspondência nos dados revela a utilidade do *Disaster-FD* em integrar informações de fontes cruzadas para proporcionar uma leitura mais abrangente da saúde da rede, ao vincular essas observações da Tabela 10, identifica-se

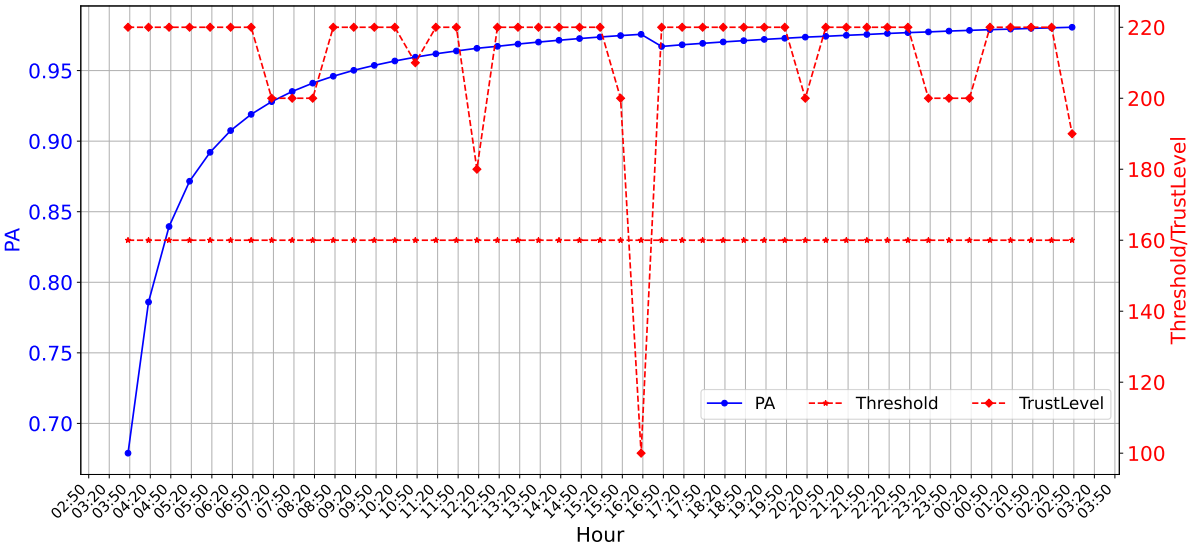

Figura 29 – Estatística da rede em Strasbourg.

momentos críticos onde o Nível de Confiança caiu acentuadamente para 100. Isso se deve, em parte, à inatividade simultânea dos dispositivos em Grenoble, refletida na escalada abrupta de erros mostrado na Figura 27, culminou em uma detecção aprimorada de falhas pela infraestrutura do *Disaster-FD*.

De forma significativa, a sequência do "Vetor de dispositivos" `TTTTTTTTTTFFFF` indica uma falha **(valor "F")** nos quatro dispositivos da rede vizinha em Grenoble, assim sendo a sequência de Vetor dos dispositivos, indicando falhas nos dispositivos adjacentes, que válida ainda mais o *Disaster-FD* como uma ferramenta no gerenciamento e na resposta a incidentes na rede IoT. Esses resultados não apenas atestam a funcionalidade do sistema, mas também fornecem *insights* críticos para o aprimoramento de estratégias de resposta a desastres, elevando o patamar de monitoramento e segurança em redes IoT.

A Tabela 10 fornece um registro adicional dos "Vetores de dispositivos" e dos níveis de confiança em momentos específicos.

Tabela 10 – Extrato dos *logs* para os monitores em Strasbourg e Grenoble.

Região	Nível de Confiança	Dispositivos	P.A.	Hora
Strasbourg	100	`TTTTTTTTTTFFFF`	0.8061	04-01-2024 03:17
Strasbourg	100	`TTTTTTTTTTFFFF`	0.9756	04-01-2024 16:17
Grenoble	120	`FFFFFFFFFFTTTT`	0.9963	04-01-2024 03:17
Grenoble	120	`FFFFFFFFFFTTTT`	0.9676	04-01-2024 16:17

5.6.4 Estatística da Rede em Grenoble

A Figura 30 apresenta a estatística da rede IoT de Grenoble, onde as linhas coloridas no gráfico representam indicadores-chave do desempenho da rede. A linha azul, que denota a Probabilidade de Acurácia (P.A.), mostra uma recuperação inicial e subsequente estabilidade acima do limiar verde, o que indica uma precisão consistente do sistema *Disaster-FD* na detecção do estado operacional dos dispositivos monitorados. Esta linha azul acima do limiar demonstra a capacidade do sistema de manter uma avaliação precisa a despeito das oscilações iniciais.

Por outro lado, o Nível de Confiança, retratado pela linha vermelha, exibe uma tendência mais variável, com momentos em que desce abaixo da linha verde que representa o limiar de segurança ou *Threshold*. Este comportamento indica uma variabilidade na percepção da confiabilidade da rede, com pontos onde a

confiança é restaurada e outros em que diminui significativamente, refletindo uma dinâmica de alterações na rede de Grenoble.

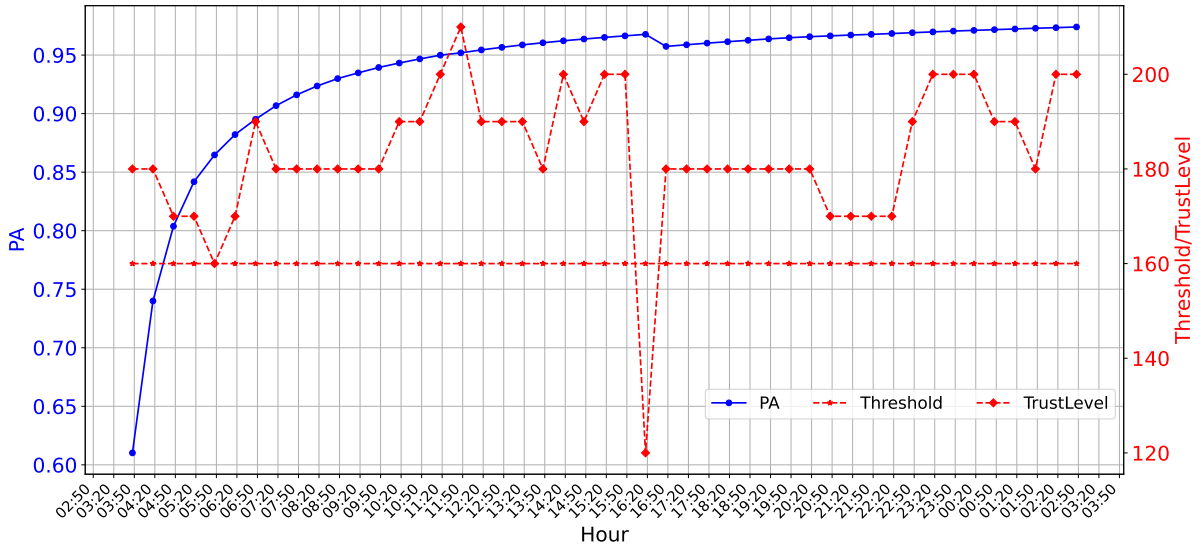

Figura 30 – Estatística da rede Grenoble.

Em ambas as ocasiões registradas, às 03:17 e 16:17, o Nível de Confiança de Grenoble é evidenciado abaixo do limiar de segurança, corroborando momentos de instabilidade da rede sugeridos pela linha vermelha do gráfico, ademais as incursões da linha vermelha próxima do limiar verde são momentos significativos, indicando uma redução na confiabilidade da rede de Grenoble, demostrando sua instabilidade. Além disso, a correlação entre as Figuras 27 e 28 é evidente, pois elas mostram os Erros Acumulados por dispositivo, onde alguns apresentam um número significativo de erros. Esta tendência nos erros acumulados para a maioria dos dispositivos acompanha as flutuações no Nível de Confiança e coincide com as marcações de **(“F”)** na Tabela 10, ilustrando uma relação entre os erros dos dispositivos e as variações no Nível de Confiança da rede. A consistência entre os picos de erros acumulados e os pontos baixos no Nível de Confiança reforça a precisão do sistema *Disaster-FD* em refletir o estado atual da rede IoT de Grenoble, revelando uma sincronização nos dados estatísticos e nos erros acumulados que também é observada na região de Strasbourg. Esta sincronia demonstra uma integração efetiva de monitoramento entre as duas regiões federadas, com implicações importantes para a avaliação interconectada da saúde da rede IoT.

Conclusão entre as duas estatísticas

Os momentos identificados de baixa confiança na rede de Grenoble, especificamente às 03:17 e 16:17, refletem diretamente os intervalos críticos observados na rede de Strasbourg. Essa sincronia nos episódios de instabilidade entre as duas regiões federadas, evidenciada pela coincidência temporal nos dados estatísticos e nos erros acumulados, destaca um monitoramento inter-regional eficaz proporcionado pelo sistema *Disaster-FD*.

As métricas de confiança em ambas as regiões, influenciadas pelos “Vetores de dispositivos” revelam falhas significativas nos dispositivos locais e adjacentes, indicam um ponto de convergência nas tendências de falha, ressaltando a interconectividade e a interdependência das redes. A correspondência entre os períodos de baixo nível de confiança e os vetores de dispositivo **(“F”)** aponta para uma capacidade do sistema em detectar e reagir de maneira coordenada a incidentes em uma configuração federada de rede IoT.

A detecção simultânea de discrepâncias nas regiões de Grenoble e Strasbourg reforça a premissa de que uma estratégia de monitoramento centralizado pode ser um mecanismo eficiente para o gerenciamento da integridade da rede em larga escala. A harmonização nos dados e nos momentos de alerta entre as regiões reafirma a necessidade de uma abordagem unificada e colaborativa no monitoramento de redes IoT complexas, onde a robustez do sistema é fundamental para uma resposta rápida e eficiente a eventos adversos.

5.6.5 Comparação com Impact-FD em Strasbourg

A estatística da rede em Strasbourg, usando a implementação *Impact-FD*, mostrou que a Probabilidade Acurácia P.A. variou entre aproximadamente 0.40 e 0.90 ao longo do tempo. A P.A., média permaneceu acima do limite de 0.40, inicialmente inferior à do *Disaster-FD*, mas indicando uma melhoria na precisão da detecção de falhas durante o experimento.

O limiar de segurança da rede foi mantido em 160, o nível de confiança variou entre 80 e 220, mostrando momentos de instabilidade durante o período, especialmente entre 15:50 e 16:50, quando houve uma queda significativa na confiança. Em um cenário real, isso poderia indicar um possível desastre, desencadeando ações de emergência para as regiões afetadas, um cenário também apresentado pelo detector de falhas *Disaster-FD*.

Comparando as estatísticas da rede para Strasbourg usando as fórmulas *Impact-FD* e *Disaster-FD*, observa-se que o *Disaster-FD* demonstrou uma probabilidade acurácia variando entre 0.70 e 0.95, atingindo 0.95 no início. Em contraste, o *Impact-FD* mostrou maior variabilidade, com a P.A. variando entre 0.40 e 0.90, atingindo 0.90 apenas no final do experimento. Assim, o *Disaster-FD* foi mais estável e se adaptou rapidamente às instabilidades da rede, enquanto o *Impact-FD* foi menos estável e se adaptou mais lentamente.

O nível de confiança na fórmula *Disaster-FD* foi mais estável, variando menos em comparação com o *Impact-FD*, o *Disaster-FD* considera o ID da mensagem em vez de um contador simples para calcular o tempo estimado de chegada da próxima mensagem, contribuindo para maior precisão e estabilidade na detecção de falhas. A propriedade Acurácia P.A. no *Disaster-FD* mostrou uma vantagem clara, resultando em menos falsos positivos e maior confiabilidade na detecção de falhas. A figura 31 ilustra a estatística para a região de Strasbourg usando a fórmula *Impact-FD*.

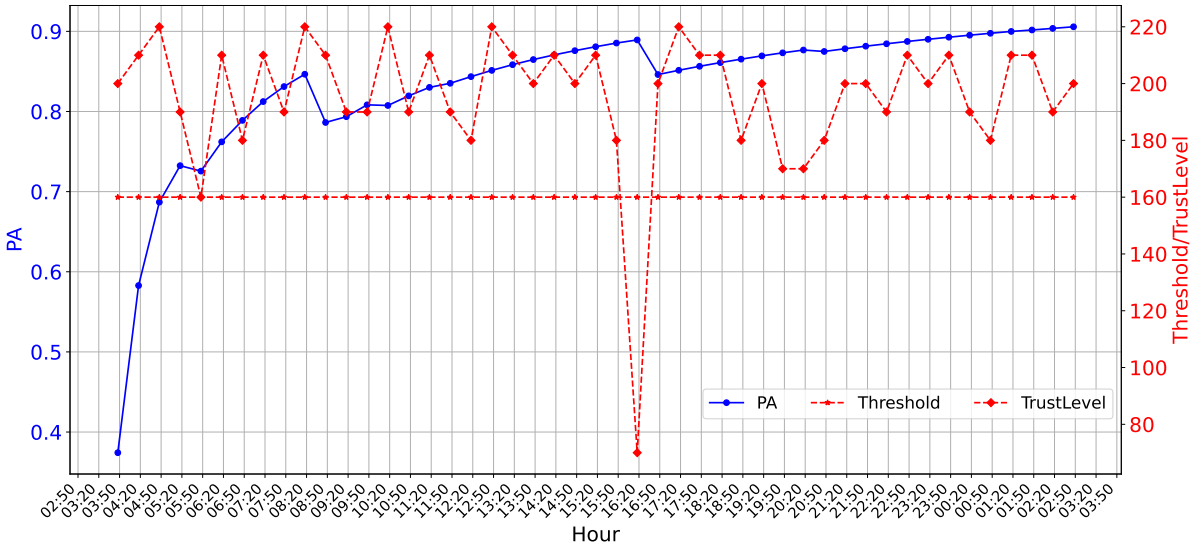

Figura 31 – Estatísticas da rede em Strasbourg usando Impact-FD.

5.6.6 Comparação com Impact-FD em Grenoble

Para a rede de Grenoble, usando a fórmula *Impact-FD*, a P.A. variou de aproximadamente 0.02 a 0.80. O limiar de segurança da rede foi mantido em 160. O nível de confiança variou entre 120 e 200, com instabilidades mais acentuadas ao longo do experimento, o que era esperado devido à maior instabilidade da região. A P.A. mais baixa durante momentos críticos indicou menor confiabilidade na detecção de falhas, resultando em uma maior ocorrência de falsos positivos.

Ao comparar os gráficos para a região de Grenoble, observou-se que o *Disaster-FD* manteve uma acurácia P.A. alta e estável, variando de 0.60 a 0.95, enquanto o *Impact-FD* mostrou maior variabilidade, com P.A. variando de 0.02 a 0.80. Grenoble experimentou mais instabilidades na rede durante o experimento em comparação com Strasbourg. O *Disaster-FD* adaptou-se rapidamente a essas instabilidades, ao contrário do *Impact-FD*, que teve uma adaptação mais lenta e inconsistências na detecção, o limiar de segurança foi mantido em 160 para ambas as fórmulas. A figura 32 ilustra as estatísticas para a região de Grenoble usando a fórmula *Impact-FD*.

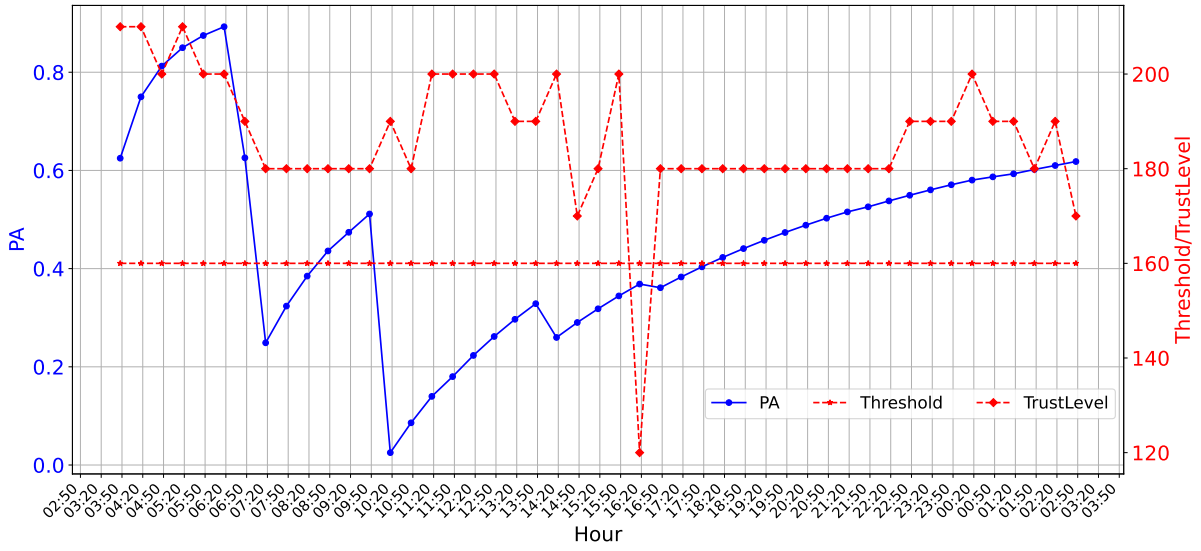

Figura 32 – Estatísticas da rede em Grenoble usando Impact-FD.

O nível de confiança na fórmula *Disaster-FD* mostrou uma clara vantagem em termos de estabilidade e menor variabilidade, o uso do ID da mensagem na fórmula *Disaster-FD* resultou em uma estimativa de tempo de chegada mais precisa, melhorando a acurácia P.A. e reduzindo significativamente a probabilidade de falsos positivos.

Portanto, a comparação das estatísticas entre as regiões de Strasbourg e Grenoble, usando as fórmulas *Impact-FD* e *Disaster-FD*, revela que ambas as fórmulas são eficazes em manter um nível de confiança acima do limite crítico, garantindo a confiabilidade da rede. No entanto, a fórmula *Disaster-FD* mostrou uma clara vantagem em termos de estabilidade, com menor variabilidade nos níveis de confiança, especialmente em redes mais instáveis como Grenoble. Essa estabilidade adicional pode ser crucial em cenários de desastres naturais, onde a confiabilidade contínua da rede é essencial para a detecção precoce de falhas e resposta rápida a eventos adversos.

A vantagem do *Disaster-FD* em usar o ID da mensagem em vez de um contador simples para calcular o tempo estimado de chegada da próxima mensagem contribui para maior precisão e estabilidade na rede. A propriedade Acurácia P.A. foi significativamente maior no *Disaster-FD*, resultando em menos falsos positivos e maior confiança na detecção de falhas. Logo, usar a fórmula *Disaster-FD* pode oferecer melhor resiliência e consistência para monitoramento em tempo real em ambientes suscetíveis a desastres.

5.6.7 Tempo Estimado e Efetivo dos Heartbeats

A Figura 33 apresenta a análise do comportamento do Dispositivo 5, localizado em Strasbourg, ao longo de um período monitorado de 24 horas.

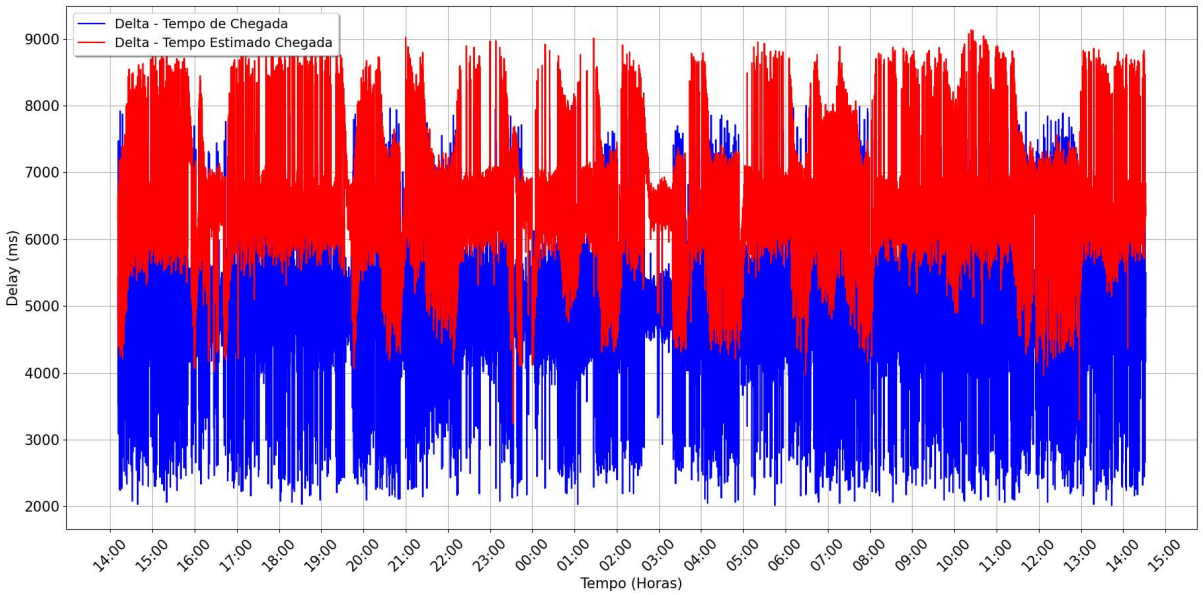

Figura 33 – Tempo de chegada e Tempo Estimado para Dispositivo 5 (Strasbourg).

A escolha desse dispositivo específico para a análise foi baseada em sua estabilidade operacional, uma característica destacada pela sua performance consistente, como evidenciado anteriormente na Figura 27.

Esta análise busca avaliar a eficiência do sistema *Disaster-FD* em contextos de monitoramento em tempo real de redes IoT em ambientes propensos a desastres. O sistema *Disaster-FD* utiliza uma margem de segurança de 1.500 milissegundos, conforme explicado na Seção 5.3, com intervalos de 5.000 milissegundos para o envio de *heartbeats*, a definição desse intervalo para o envio de *heartbeats*, adotada pelo sistema *Disaster-FD*, baseia-se nos resultados dos testes conduzidos com sensores na plataforma IoT-Lab, conforme descrito na Seção 5.4.2, nestes testes investigamos a relação direta entre a frequência de envio dos *heartbeats* para os sensores e o consumo de energia.

A Figura 33 compara os tempos estimados de chegada dos *heartbeats*, marcados em vermelho, contra os tempos reais de chegada, indicados em azul. A análise desses dados visa avaliar a precisão das estimativas de chegada em relação às observações reais, servindo como um indicador da capacidade do sistema em prever e ajustar-se às condições variáveis da rede. A observação dos dados revela uma concordância entre os tempos estimados e os reais, indicando uma baixa variação entre as previsões do sistema e os tempos efetivos de chegada dos *heartbeats*.

A predominância de pontos vermelhos sobre os azuis em certos intervalos sugere que, na maioria dessas instâncias, os *heartbeats* chegaram ao monitor antes do tempo previsto. Este padrão de chegada antecipada dos *heartbeats* realça a eficácia do sistema *Disaster-FD* em manter a comunicação dentro dos limites operacionais estabelecidos, mesmo em face de variações inesperadas na rede.

A adequação da margem de segurança e dos intervalos de envio, como implementados no *Disaster-FD*, é crítica para o equilíbrio entre a detecção adequada de falhas e a minimização de alarmes falsos, portanto, a Figura 33 demonstra não somente a precisão das estimativas de chegada dos *heartbeats* geradas pelo sistema *Disaster-FD*, mas também sublinha a importância da configuração de parâmetros operacionais, como a margem de segurança e os intervalos de envio, na otimização do monitoramento de redes IoT em cenários de risco.

A Tabela 11 ilustra a sequência de recebimento das mensagens do dispositivo 5, situado na região de Strasbourg, destacando o funcionamento do sistema *Disaster-FD* em um cenário real de monitoramento na plataforma IoT-Lab.

Tabela 11 – Dados das mensagens recebidas para o dispositivo 5.

Dev	Msg	Tempo Chegada(ns)	Estimado Próxima Mensagem(ns)	Δ_Chegada(ms)	Δ_Estimado(ms)
5	1	25020640357701306	25020646857701304	-	-
5	2	25020645810193498	25020652083947400	5452	1048
5	3	25020650688915212	25020657118936672	4879	1395
5	4	25020655472370782	25020662082295200	4783	1647
5	5	25020660383644543	25020667042565068	4911	1699
5	6	25020665519198660	25020672038670668	5136	1523
5	7	25020670558745554	25020677041538504	5040	1480
5	8	25020675742067343	25020682066604608	5183	1299
5	9	25020680734232563	25020687085229936	4992	1332
5	10	25020685389719591	25020692065678904	4655	1696

1. **Coluna Dev**: Identifica o nó ou dispositivo sob monitoramento. Cada entrada nesta coluna é uma referência direta ao dispositivo específico, neste caso, o dispositivo 5.

2. **Coluna Msg**: Esta coluna representa a "mensagem", o número em cada linha da coluna indica a sequência ou o ID da mensagem que está sendo recebida, especialmente útil para rastrear e ordenar as mensagens, garantindo que elas sejam processadas na ordem correta ou para identificar possíveis perdas de mensagens.

3. **Coluna Tempo de Chegada**: Está coluna indica o "tempo de chegada" de uma mensagem, o tempo de recebimento é registrado em nanossegundos.

4. **Coluna Tempo Estimado para Próxima Mensagem**: Esta coluna representa o tempo de estimado chegada da próxima mensagem subsequente, também expresso em nanossegundos.

5. **Coluna Δ_Chegada**: Esta coluna calcula o tempo que passou entre a chegada de uma mensagem e a chegada da mensagem imediatamente anterior. Isso é feito subtraindo o tempo de chegada atual pelo tempo de chegada da mensagem anterior, transformada em milissegundos, esta coluna é útil para entendermos se o intervalo de tempo entre mensagens está aumentando, diminuindo ou permanecendo consistente.

6. **Coluna Δ_Estimado**: Confronta o tempo estimado para a chegada da próxima mensagem com o instante real de recebimento da mensagem. A subtração fornece a diferença temporal entre o que foi estimado anteriormente e o que realmente ocorreu. O resultado é a variação entre a estimativa e a realidade, isso pode mostrar se as estimativas estão próximas da realidade ou se há uma tendência de adiantamento ou atraso nas mensagens. Valores positivos indicam que a mensagem chegou antes do tempo estimado, refletindo uma previsão conservadora. Por outro lado, valores negativos são indicativos de atrasos no recebimento das mensagens.

Tabela 12 – Médias de Tempo Δ_Chegada e Δ_Estimado.

Dev	Média Δ_Chegada(ms)	Média Δ_Estimado(ms)
0	5008	1492
1	5007	1493
2	5128	1368
3	5050	1449
4	5127	1374
5	5117	1379
6	5126	1373
7	5106	1393
8	5120	1380
9	5121	1380
11	6024	478
12	5535	966
13	5414	1085

A Tabela 12 apresenta as médias dos tempos de Δ_Chegada e Δ_Estimado, valores estes derivados dos dados obtidos durante os experimentos. Essas médias refletem o comportamento dos dispositivos IoT dentro do contexto para avaliar o intervalo de chegada e estimativa de chegada das mensagens.

1. **Média de Δ_Chegada**: Esta coluna representa o tempo médio, em milissegundos, entre a chegada de *heartbeats* (ou mensagens) consecutivas para cada dispositivo. Por exemplo, para o dispositivo 0, o intervalo médio entre as chegadas é de 5.008 milissegundos. Valores mais próximos do intervalo de envio das mensagens estipulado em 5.000 milissegundos sugere que o dispositivo está estável, enquanto grandes variações poderiam indicar instabilidade ou possíveis falhas na rede.

2. **Média de Δ_Estimado**: Esta coluna quantifica a média das diferenças entre o tempo estimado para a próxima mensagem, já incrementado pela margem de segurança, e o tempo real de chegada da mensagem atual. Isso reflete a acurácia do sistema em prever quando uma mensagem chegará, por exemplo, para o dispositivo 0, a previsão estava, em média, 1.492 milissegundos adiantada em relação ao tempo real. Logo quando os valores da coluna "Δ_Estimado" são positivos e próximos da margem de segurança, isso indica que as mensagens estão chegando ligeiramente antes do tempo previsto, mostrando que as estimativas do sistema são bastante precisas. Entretanto, valores negativos ou valores positivos que estão próximos de zero apontam para atrasos, significando que o tempo real de chegada foi maior do que o tempo estimado mais a margem de segurança. Portanto, quanto mais próximo ou abaixo de zero for o valor de "Δ_Estimado", maior o atraso observado em relação à estimativa ajustada.

Observa-se uma diferença significativa na coluna de média "Δ_Estimado" entre os dispositivos 0 a 9 e os dispositivos 11 a 13. Para os dispositivos 0 a 9, a média "Δ_Estimado" é de aproximadamente 1400 milissegundos, indicando uma boa precisão na estimativa dos tempos de chegada das mensagens. Em contraste, para os dispositivos 11 a 13, que são dispositivos vizinhos situados em Grenoble, a média "Δ_Estimado" é consideravelmente mais baixa (478 ms, 966 ms e 1085 ms, respectivamente). Essa menor precisão nas estimativas para os dispositivos em Grenoble está alinhada com as observações dos gráficos anteriores, que indicaram instabilidades na rede dessa região em comparação com a região de Strasbourg. Tais instabilidades podem impactar a acurácia das estimativas de tempo de chegada, evidenciando diferenças nas condições de rede entre as duas regiões.

5.6.8 Comparando Tempos Reais e Estimados de Chegada em Strasbourg

A Figura 34 ilustra duas variáveis relacionadas ao ambiente de teste no ecossistema de rede IoT da plataforma Iot-Lab: “Tempo de Chegada” e “Tempo Estimado de Chegada”. O eixo horizontal (X) exibe os valores de milissegundos, enquanto o eixo vertical (Y) mostra a probabilidade cumulativa de que os tempos registrados sejam menores ou iguais aos valores correspondentes no eixo X.

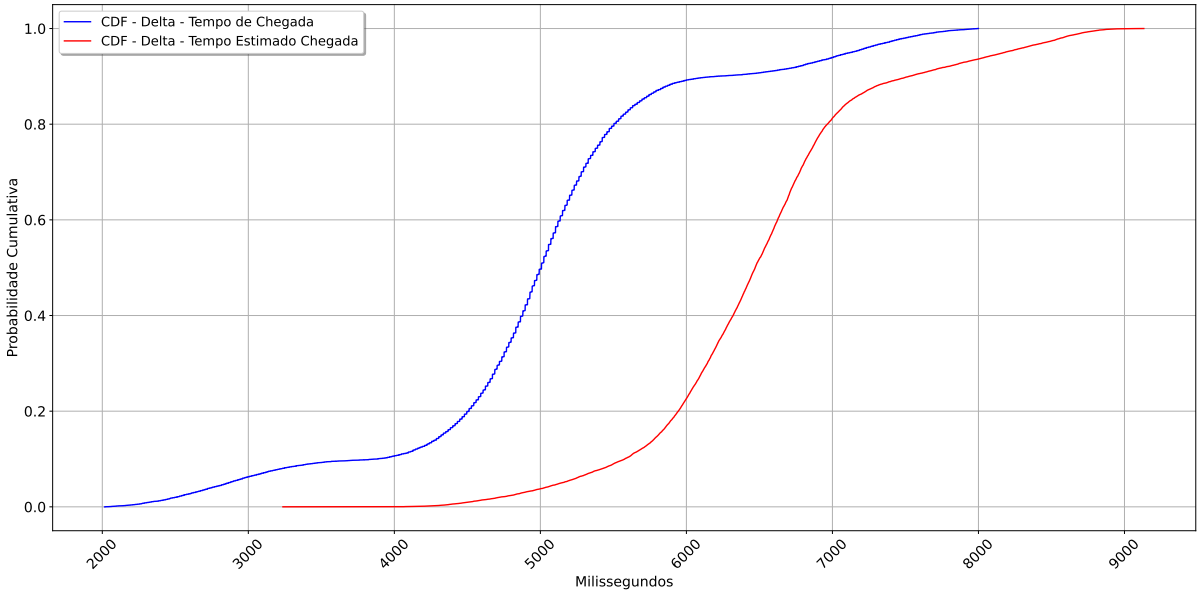

Figura 34 – Comparação CDF de Tempos Reais e Estimados de Chegada - Strasbourg.

O gráfico *Cumulative Distribution Function* - CDF, apresenta duas funções que comparam o desempenho de tempos de chegada observados contra tempos de chegada estimados em um sistema em análise. No eixo abscissa (X), a unidade temporal é expressa em milissegundos, denotando a métrica de tempo sob avaliação. No eixo ordenada (Y), tem-se a probabilidade cumulativa, cuja escala varia de 0 a 1, correspondendo a 0% a 100% da amostra observada, as curvas CDF traçadas no gráfico permitem a comparação entre os tempos reais de chegada e os tempos estimados.

A curva em azul, que representa o “Tempo de Chegada”, mostra como os dados reais foram distribuídos. Por exemplo, pode-se observar que cerca de 50% dos eventos ocorreram antes de 5.000 milissegundos, já que a curva azul atinge a probabilidade de 0,5 (ou 50%) neste valor. Da mesma forma, quase todos os eventos (100% deles) aconteceram antes de 8.000 milissegundos, já que a curva se aproxima da totalidade naquela marca.

A curva em vermelho, correspondente ao “Tempo Estimado de Chegada”, reflete as expectativas ou previsões de quando os eventos deveriam ocorrer. Uma análise direta do gráfico mostra duas curvas CDF: a curva azul indica o Tempo Real de Chegada dos eventos, e a curva vermelha representa o Tempo Estimado de Chegada, com uma margem de segurança de 1.500 milissegundos. A proximidade entre as duas curvas sugere que os tempos reais de chegada estão alinhados com os tempos estimados, incluindo a margem de segurança.

A diferença no eixo x entre as duas curvas para um mesmo valor de probabilidade cumulativa corresponde à margem de segurança adotada de 1.500 milissegundos na grande maioria dos casos, reforçando a acurácia dos cálculos de estimativa de chegada de mensagens. Por conseguinte a posição da curva vermelha à direita da azul indica que o “Tempo Estimado de Chegada” foi consistentemente maior que o tempo real de chegada, isso reflete um conservadorismo na estimativa, conforme detalhado na seção 5.3 sobre a Definição da Margem de Segurança (β).

5.7 Considerações Finais

5.7.1 Cenário ideal para avaliar o Disaster-FD

O cenário ideal para avaliar o *Disaster-FD* envolve um ambiente de rede IoT distribuído, onde múltiplas regiões são monitoradas simultaneamente. Este ambiente deve ser propenso a condições adversas, como desastres naturais (inundações, deslizamentos, etc.), que afetam a estabilidade da rede e exigem uma resposta rápida e precisa.

A avaliação deve incluir variações de conectividade e energia, simulando perdas intermitentes de sinal e flutuações no fornecimento de energia para validar a capacidade do *Disaster-FD* em manter a integridade da rede e a eficácia na detecção de falhas. A plataforma FIT IoT-LAB mencionada no estudo foi um excelente exemplo de ambiente de teste, oferecendo as condições necessárias para avaliar a eficiência do *Disaster-FD* em cenários de uso real, como demonstrado nos testes realizados.

5.7.2 Retomando as hipóteses estabelecidas na seção 1.3

As hipóteses foram testadas e confirmadas ao longo da pesquisa, sendo que a primeira hipótese, de que os detectores de falhas empregados em redes tradicionais poderiam ser adaptados para redes de sensores IoT, foi validada. O *Disaster-FD* demonstrou a capacidade de monitorar a rede como um todo, permitindo uma detecção de falhas mais integrada e abrangente em cenários de desastres.

A segunda hipótese, sugere que a utilização de comunicação federada entre detectores de falhas em diferentes regiões poderia melhorar a eficiência na detecção de desastres naturais, também foi confirmada. Os resultados mostraram que o sistema *Disaster-FD*, ao adotar uma abordagem de monitoramento federado e em tempo real, foi capaz de melhorar a eficiência na detecção de falhas, oferecendo uma visão mais holística das condições da rede e permitindo respostas mais rápidas e coordenadas a eventos adversos. Especificamente a falha na região de Grenoble pôde ser percebida pelo monitor em Strasbourg.

A pesquisa e os experimentos realizados confirmaram a eficácia do *Disaster-FD* como uma ferramenta com potencial para o monitoramento de redes IoT em cenários de desastres naturais. A capacidade do sistema de ajustar-se dinamicamente às condições variáveis da rede, utilizando monitoramento federado contribui para a resiliência da infraestrutura de monitoramento e a mitigação de desastres.

A metodologia desenvolvida prova ser uma abordagem promissora para futuros estudos e implementações em ambientes críticos, destacando-se como uma solução que pode ser explorada e melhorada no campo da detecção de falhas e gestão de desastres. Assim, o trabalho realizado não só alcança os objetivos propostos, como também oferece uma base sólida para futuras pesquisas e desenvolvimentos na área de monitoramento de redes IoT e gestão de desastres.

Capítulo 6

Conclusão

O trabalho apresentado na dissertação "*Disaster-FD*" reflete um esforço no desenvolvimento de uma solução para o monitoramento de redes IoT em cenários de desastres naturais. Utilizando uma combinação de técnicas de monitoramento federado e tempo real, este estudo propõe uma metodologia que integra os protocolos CoAP e ICMP para assegurar uma resposta rápida e eficiente diante de condições adversas, com monitoramento de dispositivos IoT e computadores tradicionais.

Inspirado pelo projeto *Impact-FD* (ROSSETTO et al., 2018) , o *Disaster-FD* foi concebido com o objetivo de melhorar a detecção e resposta a falhas em ambientes IoT propensos a desastres naturais. A motivação por trás deste projeto centra-se na crescente necessidade de sistemas de monitoramento que não apenas detectem falhas de forma eficaz, mas também operem de maneira sustentável e eficiente em termos energéticos em cenários prolongados de crise.

A implementação do sistema *Disaster-FD* incorpora um sistema de gerenciamento de *threads*, otimizando o processamento e a execução de tarefas de monitoramento através de um modelo de concorrência muitos-para-muitos. Este sistema permite a execução de tarefas em *threads* separadas, o que é essencial para manter a alta disponibilidade e confiabilidade requerida em ambientes críticos.

Durante os testes, o sistema demonstrou sua capacidade de identificar de forma efetiva as falhas em dispositivos distribuídos, utilizando o monitoramento de *threads* para garantir que todos os processos funcionem sem interrupções.

Além disso o *Disaster-FD* utilizou o *Message* ID (MID) das respostas das requisições CoAP, a substituição do contador numérico convencional pelo MID para auxiliar no cálculo do tempo estimado de chegada (EA) das mensagens resultou em uma abordagem que permitiu uma previsão mais precisa da chegada de mensagens subsequentes, essencial para a otimização do tempo de resposta em situações de crise.

Ademais, o *Disaster-FD* foi projetado para analisar não somente as mensagens recebidas com sucesso, mas também aquelas que falharam, esta funcionalidade amplia a robustez do sistema, permitindo não apenas a detecção de falhas operacionais, mas também a análise de padrões de falha, o que é crucial para antecipar e mitigar problemas antes que eles possam causar danos extensivos.

Os objetivos delineados na seção 1.2 e as hipóteses propostas na seção 1.3 foram alcançados e confirmados, conforme demonstrado nos testes apresentados no Capítulo 5, o uso do MID para aprimorar o cálculo de Estimativa de Chegada (EA), substituindo contadores numéricos, permitiu uma previsão mais precisa do tempo de chegada das mensagens, melhorando assim a capacidade de resposta do sistema em cenários críticos.

A hipótese de que a utilização de comunicação federada entre detectores de falhas poderia melhorar a eficiência na detecção de desastres foi comprovada pelos resultados. O sistema não só analisou mensagens recebidas com sucesso, mas também aquelas com falhas, o que ampliou consideravelmente a robustez do *Disaster-FD*.

O monitoramento federado permitiu uma visão holística e integrada das condições da rede IoT, fundamental para uma resposta rápida a falhas. Além disso, a integração de técnicas de monitoramento do consumo de energia destacou a preocupação com a sustentabilidade e eficiência energética. A consideração da demanda energética dos dispositivos e os intervalos de envio das requisições permitiram otimizar o uso de recursos, garantindo a longevidade e eficácia do sistema em operações prolongadas.

Esses resultados destacam o *Disaster-FD* como um promissor elemento na detecção de falhas em ambientes IoT, ao integrar análises detalhadas de sucesso e falha de transmissões, o sistema oferece uma visão abrangente e detalhada do comportamento da rede, melhorando a capacidade de resposta em face de desastres naturais e outras condições adversas.

6.1 Trabalhos Futuros

Para promover a evolução contínua do "*Disaster-FD*", considerando os resultados obtidos na detecção de falhas e no monitoramento de ecossistemas IoT, propõem-se as seguintes áreas de pesquisa futura. Essas iniciativas visam melhorar ainda mais a eficácia e a eficiência do sistema, assim como aprimorar a capacidade de monitoramento da rede como um todo.

- ❏ Análise da frequência de falhas: o algoritmo poderia monitorar a frequência de falhas na rede de sensores ao longo do tempo para identificar tendências e determinar se algum tipo de manutenção preventiva é necessária.

- ❏ Notificação baseada em falhas: o algoritmo, ao perceber um número massivo de notificações de falhas da rede de sensores, poderia considerar a hipótese de desastre na rede IoT, e com isso antecipar um possível cenário de desastre natural.

- ❏ Utilização da Função ComputeMargin: inspirado no projeto *Stab-FD* (SENS et al., 2024), o *Disaster-FD* aplicaria uma função adaptativa para calcular a margem de segurança baseada na variação da estabilidade dos links, o que ajudará a ajustar os *timers* de acordo com as condições atuais da rede, promovendo uma detecção de falhas mais precisa e confiável.

- ❏ Integração com Telegram para Notificações de Confiabilidade da Rede: desenvolver uma funcionalidade no *Disaster-FD* que utilize o Java `Telegram Bot API` para enviar alertas automáticos aos administradores de sistema sempre que a rede apresentar indícios de instabilidade ou quando a confiabilidade cair abaixo de um limiar pré-definido.

Os trabalhos futuros propostos para o *Disaster-FD* visam expandir significativamente suas capacidades, introduzindo funcionalidades que não só aprimorarão a detecção de falhas, mas também a eficácia geral na gestão de redes IoT.

6.2 Contribuições em Produção Bibliográfica

O projeto "*Disaster-FD*" foi apresentado inicialmente durante o IX FACOM TechWeek e o XVI Workshop de Teses e Dissertações em Ciência da Computação, um evento de extensão realizado nos dias 10 e 11 de novembro de 2022 pela Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia.

Adicionalmente o artigo "*Disaster-FD*, um detector de falhas para ambientes suscetíveis a desastres com foco no monitoramento em tempo real de redes IoT", foi submetido e aprovado para publicação e apresentado no evento WTF 2024 - XXV Workshop de Testes e Tolerância a Falhas, que ocorreu entre os

dias 20 e 24 de maio de 2024, em Niterói/RJ. O artigo foi reconhecido com uma menção honrosa durante o simpósio.

Autores: Abadio de Paulo Silva, Paulo Coelho e Rafael Pasquini (Universidade Federal de Uberlândia – Minas Gerais – Brasil), Luciana Arantes e Pierre Sens (Université de Sorbonne, CNRS, LIP6 – Paris – França), Anubis Graciela de Moraes Rossetto (Instituto Federal Sul-rio-grandense – Rio Grande do Sul – Brasil).

Cabe destacar que o artigo "*Disaster-FD*" será submetido ao XIII Simpósio Latino-Americano de Computação Confiável e Segura (LADC), que ocorrerá em Recife - Brasil, entre os dias 26 e 29 de novembro de 2024. Esta nova submissão incluirá atualizações, abrangendo o monitoramento do consumo de energia e uma comparação entre o *Disaster-FD* e seu predecessor *Impact-FD*, focando na apresentação do nível de confiança da rede e a precisão nas regiões monitoradas, demonstrada pelo gráfico estatístico da rede nas regiões de Strasbourg e Grenoble.

Referências

ADJIH, C. et al. Fit iot-lab: The largest iot open experimental testbed. 04 2015. Disponível em: <https://doi.org/10.1109/WF-IoT.2015.7389098>.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, Elsevier, v. 54, n. 15, p. 2787–2805, 2010. Disponível em: <https://doi.org/10.1016/j.comnet.2010.05.010>.

BASU, A.; CHARRON-BOST, B.; TOUEG, S. Simulating reliable links with unreliable links in the presence of process crashes. **Distributed algorithms**, Springer, p. 105–122, 1996. Disponível em: <https://doi.org/10.1007/3-540-61769-8_8>.

CAN, Z.; DEMIRBAS, M. Querying on federated sensor networks. **Journal of Sensor and Actuator Networks**, MDPI, v. 5, n. 3, p. 14, 2016. Acessada: 2023-02-14. Disponível em: <https://doi.org/10.3390/jsan5030014>.

CARRION, P.; QUARESMA, M. Internet da coisas (iot): Definições e aplicabilidade aos usuários finais. **Human Factors in Design**, v. 8, n. 15, p. 049–066, mar. 2019. Disponível em: <https://doi.org/10.5965/2316796308152019049>.

CHANDRA, T. D.; HADZILACOS, V.; TOUEG, S. The weakest failure detector for solving consensus. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 43, n. 4, p. 685–722, 1996. Disponível em: <https://doi.org/10.1145/234533.234549>.

CHANDRA, T. D.; TOUEG, S. Unreliable failure detectors for reliable distributed systems. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 43, n. 2, p. 225–267, 1996. Disponível em: <https://doi.org/10.1145/226643.226647>.

CHEN, W.; TOUEG, S.; AGUILERA, M. K. On the quality of service of failure detectors. **IEEE Transactions on computers**, IEEE, v. 51, n. 5, p. 561–580, 2002. Disponível em: <https://doi.org/10.1109/TC.2002.1004595>.

COULOURIS, G. et al. **Sistemas distribuídos: conceitos e projeto**. 5. ed. Porto Alegre: Bookman, 2013. ISBN 978-85-8260-054-2.

CRISTIAN, F.; FETZER, C. The timed asynchronous distributed system model. **IEEE Transactions on Parallel and Distributed Systems**, v. 10, n. 6, p. 642–657, 1999. Disponível em: <https://doi.org/10.1109/71.774912>.

DELPORTE-GALLET, C.; FAUCONNIER, H.; GUERRAOUI, R. **Shared Memory vs Message Passing**. [S.l.], 2003. Disponível em: <https://infoscience.epfl.ch/record/52584>.

DOLEV, D.; DWORK, C.; STOCKMEYER, L. On the minimal synchronism needed for distributed consensus. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 34, n. 1, p. 77–97, 1987. Disponível em: <https://doi.org/10.1145/7531.7533>.

ECLIPSE FOUNDATION. **Californium (Cf) CoAP Framework**. 2024. <https://javadoc.io/doc/org.eclipse.californium/californium-core/latest/index.html>.

FILHO, F. A. M.; FILHO, J. G. P.; GOMES, R. L. Um framework para compartilhamento de objetos inteligentes via redes sociais. In: **Proceedings of the X Brazilian Symposium in Collaborative Systems**. BRA: Sociedade Brasileira de Computação, 2013. (SBSC '13), p. 32–39. ISBN 9788576692805. Disponível em: <https://dl.acm.org/doi/10.5555/2542508.2542515>.

FISCHER, M. J.; LYNCH, N. A.; PATERSON, M. S. Impossibility of distributed consensus with one faulty process. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 32, n. 2, p. 374–382, 1985. Disponível em: <https://doi.org/10.1145/3149.214121>.

GIUNTINI, F. T. **Um modelo de rede de sensores sem fio autoorganizada e tolerante a falhas para detecção de incêndios**. Dissertação (Dissertação de Mestrado) — Universidade Federal de São Carlos, São Carlos,São Paulo, 2016. Disponível em: <https://repositorio.ufscar.br/bitstream/handle/ufscar/8296/DissFTG.pdf?sequence=1&isAllowed=y>.

GRANEMANN, D.; CARNEIRO, G. Monitoramento de focos de incendio e Áreas queimadas com a utilizaÇÃo de imagens de sensoriamento remoto. v. 1, p. 1–8, 05 2009. Acessada: 2024-08-26. Disponível em: <https://revistas.uepg.br/index.php/ret/article/view/11431/209209209404>.

JANONEDA, L. **A cada desastre natural no Brasil, em média, 3,4 mil pessoas são afetadas**. 2022. Acessada: 2022-01-09. Disponível em: <https://www.cnnbrasil.com.br/nacional/a-cada-desastre-natural-no-brasil-em-media-34-mil-pessoas-sao-afetadas>.

KOVATSCH, M.; LANTER, M.; SHELBY, Z. Californium: Scalable cloud services for the internet of things with coap. In: **2014 International Conference on the Internet of Things (IOT)**. [s.n.], 2014. p. 1–6. Disponível em: <https://doi.org/10.1109/IOT.2014.7030106>.

MELLO, G. de; SILVA, L. A.; LEITHARDT, V. R. Cosgp-iot-sg: Uma proposta para integração em smartgrid para dispositivos com capacidade limitada. In: **Anais da XIX Escola Regional de Alto Desempenho da Região Sul**. Porto Alegre, RS, Brasil: SBC, 2019. ISSN 2595-4164. Disponível em: <https://sol.sbc.org.br/index.php/eradrs/article/view/7033>.

PASQUINI, R. et al. Admits: Architecting distributed monitoring and analytics in iot-based disaster scenarios. p. 11–20, 2020. Disponível em: <https://doi.org/10.5753/sbcup.2020.11207>.

RFC Editor. **The Constrained Application Protocol (CoAP)**. 2023. Updated by RFC 7959, 8613, 8974, 9175. Disponível em: <https://www.rfc-editor.org/info/rfc7252>.

ROSSETTO, A. G. d. M. et al. Impact fd: An unreliable failure detector based on process relevance and confidence in the system. **The Computer Journal**, Oxford University Press, v. 61, n. 10, p. 1557–1576, 2018. Disponível em: <https://doi.org/10.1093/comjnl/bxy041>.

SASIKUMAR, M.; NARAYANAN, A. A low power system for wireless sensor networks with highly consistent data collection. **TEST Engineering & Management**, Periyar Maniammai Institute of Science and Technology, v. 83, n. March -April 2020, p. 23546–23553, 2020. Disponível em: <https://testmagzine.biz/index.php/testmagzine/article/view/11545/8866>.

SENS, P. et al. Stab-fd: A cooperative and adaptive failure detector for wide area networks. **Journal of Parallel and Distributed Computing**, v. 186, 2024. Disponível em: <https://doi.org/10.1016/j.jpdc.2023.104803>.

SILVEIRA, R. M. et al. Sistema distribuído de monitoramento de rede para previsão de desastre. **9ª Conferência Ibero Americana de Computação Aplicada**, São Paulo, SP, Brasil, 2022. Disponível em: <https://ciaca-conf.org/wp-content/uploads/2022/11/2_CIAWI2022_PT_S_051.pdf>.

SOUZA, J. C. **Logística Humanitária – Distribuição Espacial de Centrais de Atendimento de Emergência para Populações Atingidas por Desastres Naturais**. [S.l.: s.n.], 2011. 274-285 p.

VERISSIMO, P.; RODRIGUES, L. **Distributed Systems for System Architects**. [S.l.]: Springer Science & Business Media, 2012. v. 1.

VIEIRA, T. P. F. **Gerenciamento inteligente de falhas aplicado a rede de sensores sem fio com integração à internet das coisas**. Dissertação (Mestrado) — Centro Federal de Educação Tecnológica de Minas Gerais, 2017. Disponível em: <https://sig.cefetmg.br/sigaa/verArquivo?idArquivo=2121364&key=80885f3b9a8342c7f3badd437d999eba>.

YANG, R. et al. Medley: A novel distributed failure detector for IoT networks. In: **Proceedings of the 20th International Middleware Conference**. [s.n.], 2019. p. 319–331. Disponível em: <https://doi.org/10.1145/3361525.3361556>.