CESAR SCHOOL

RICARDO NOGUEIRA SANTOS

MESTRADO PROFISSIONAL EM ENGENHARIA DE SOFTWARE
UMA API EXTENSÍVEL E ADAPTÁVEL PARA EXPERIMENTAÇÃO NO TRATAMENTO E TRANSMISSÃO DE IMAGENS EM REDES DE SENSORES SEM FIO.

MANAUS
2022

RICARDO NOGUEIRA SANTOS

MESTRADO PROFISSIONAL EM ENGENHARIA DE SOFTWARE UMA API EXTENSÍVEL E ADAPTÁVEL PARA EXPERIMENTAÇÃO NO TRATAMENTO E TRANSMISSÃO DE IMAGENS EM REDES DE SENSORES SEM FIO.

Dissertação apresentada ao programa de Mestrado em Engenharia de Software do Centro de Estudos e Sistemas Avançados do Recife - CESAR School, como requisito para a obtenção do título de Mestre em Engenharia de Software.

Orientador: Dr. Sylker Teles - Cesar School

Co-Orientador: Dr. Cid A. Nogueira Santos

MANAUS
2022

Catalogação da Publicação na Fonte (CIP)
Centro de Estudos e Sistemas Avançados do Recife – CESAR School em Manaus

S237a Santos, Ricardo Nogueira

Uma API Extensível e Adaptável para Experimentação no Tratamento e Transmissão de Imagens em Redes de Sensores Sem Fio. / Ricardo Nogueira Santos. Manaus: O autor, 2022.

79 p.: il.

Dissertação (Mestrado) - Centro de Estudos e Sistemas Avançados do Recife – CESAR School. Programa de Pós-Graduação em Engenharia de Software em Manaus.

Orientação: Prof. Sylker Teles da Silva
Coorientação: Prof. Cid Adinam Nogueira Santos

1. Internet das Coisas. 2. Redes de Sensores Sem Fio. 3 Sensores de Imagens. I.Silva, Sylker Teles da. (Orientador). II. Santos, Cid Adinam Nogueira Santos (Coorientador). III. Título.

CDD 005.1

RICARDO NOGUEIRA SANTOS

UMA API EXTENSÍVEL E ADAPTÁVEL PARA EXPERIMENTAÇÃO NO TRATAMENTO E TRANSMISSÃO DE IMAGENS EM REDES DE SENSORES SEM FIO

Trabalho aprovado em Manaus: **16/03/2022**

Professor: Sylker Teles da Silva
(CESAR SCHOOL)
Orientador

Professor: Allan Rodrigo dos Santos Araújo
(CESAR SCHOOL)
Avaliador Interno

Professor: Daniel Frazão Luiz
Avaliador Externo
(UFAM)

MANAUS
2022

Agradecimentos

Ao Sidia – Instituto de Tecnologia pelo integral apoio ao desenvolvimento do trabalho, desde o período de disciplinas.

Ao orientador Sylker e ao co-orientador Cid pela orientação e condução da pesquisa.

Aos meus familiares pelo suporte e compreensão durante os anos dedicados a conclusão do trabalho, mãe Antônia, esposa Mônica, filhos Murilo e Sofia e os irmãos Cid e Jorge.

Resumo

As Redes de Sensores sem Fio (Wirelles Sensor Network – WSN) são a infraestrutura tecnológica para dispositivos e aplicações IoT. Tendo usualmente requisitos de projeto tarefas de baixa complexidade e consumo reduzido de energia, tais dispositivos são viabilizados com HW de baixo custo e desempenho computacional limitado, assim como redes sem fio de baixa capacidade de transmissão.

O alto grau de desenvolvimento da indústria de semicondutores permitiu a popularização de dispositivos de captura de imagem, o que desperta o interessa da incorporação de sensores de imagens em aplicações visuais nas WSN. Todavia essa tarefa representa um grande desafio técnico, visto que o tratamento e transmissão de imagens tipicamente envolvem alta necessidade de processamento (e consumo de energia), e gera uma grande quantidade de dados para transmissão.

Várias abordagens têm sido propostas para superar as limitações de transmissão de imagens em redes de sensores, como técnicas de detecção de mudanças, algoritmos de compressão de imagens de baixo processamento e técnicas de transmissão que minimizam o tráfego de dados. Tais técnicas podem apresentar diversas variantes e serem usadas isoladamente ou em conjunto. Deste modo, existe uma grande variedade de cenários possíveis na experimentação de soluções para uma rede específica, dificultando a busca de soluções que apresentem o máximo desempenho.

Este trabalho apresenta uma API extensível e adaptável que unifica todas as possíveis etapas da transmissão de imagens em WSN, facilitando a experimentação de cenários. Um protótipo de tal API é implementada na plataforma Contiki-NG/Cooja para sua validação.

Palavras-chave: internet das coisas (IoT); redes de sensores sem fio (WSN); sensores de imagem; API.

Abstract

Wireless Sensor Networks (WSN) is a technological infrastructure for IoT devices and its applications. Usually having requirements of low complexity tasks and constrained energy consumption, such devices are built over low cost HW with limited computational performance, as well as wireless networks with low transmission capacity.

The progress of semiconductor industry has popularized image sensor based devices, such as cameras, which brings the interest to embed these sensors for WSN applications. However, this task represents a technical challenge since image processing and transmission requires computational(energy consumption) and network performance, due algorithm´s complexity and huge data amount for transmission

Some techniques have been proposed to overcome image transmission limitations in WSN, such as change detection techniques and low complexity compression algorithms to minimize networking data traffic. Such techniques may be used isolated or combined. Thus, there are a variety of possible scenarios for experimentation with specific solutions, making it difficult to find the solution with maximum performance.

This work presents an extensible and flexible API design that encapsulates all possible steps for image processing and transmission in WSN, making easier to experiment the scenarios. A prototype of API is implemented on Contiki-NG/Cooja platform for validation of the design.

Keywords: internet of things (IoT); wirelles sensor networks (WSN); image sensors; API.

Nomenclatura

AODV *Adhoc on-demand Distance vector routing*

API *Application Programming Interface*

BDCT *Binary DCT*

CCD *Charge Coupled Device*

CIF *Common Intermediate Format*

CMOS *Complementary Metal Oxide Semiconductor*

CPU *Central Processing Unit*

CR *CR Compression Ratio*

CSN *Componente Sensor da WSN*

DCT *Discrete Cosine Transform*

DSR *Dynamic Source Routing*

DWT *Discrete Wavelet Transform*

EZW *Embedded Zero tree wavelet*

ISM *Industrial, Médico e Científico (Medical, Scientific and Medical)*

LPWAN ou LPWA *Low Power Wide Area Networks*

MSE *Mean Square Error*

PSNR *Peak Signal to Noise Ratio*

QVGA *Quarter Video Graphics Array*

RFID *Radio-Frequency Identification*

SPIHT *Set Partitioning in Hierarchical Trees*

Sub-GHz *Frequências abaixo de 1 Ghz*

UNB *Ultra-Narrowband*

VGA *Video Graphics Array*

VSN *Visual Sensor Network*

WLAN *Wireless Local Area Network*

WNLAN *Wireless Neighboardhood Area Network*

WPAN *Wireless Personal Area Network*

WSN *Wireless Sensor Network*

WWAN *Wireless Wide Area Network*

Lista de Figuras

Sumário

1 Introdução e Justificativa

A União Internacional de Telecomunicações (ITU, 2018) define a Internet das Coisas (IoT) como sendo uma infraestrutura global para a sociedade da informação, habilitando serviços avançados por meio da interconexão (física e virtual) de objetos, baseado na troca de informação por meio de tecnologias de comunicações existentes e em evolução. A consultoria Gartner estima que 25 bilhões de elementos estavam conectados à internet em 2021 (GARTNER, 2018), evidenciando a importância da área.

As redes de sensores sem fio ou WSNs são arquiteturas que embarcam dispositivos dotados de sensores que estão interconectados através de redes sem fio segundo (LIN et al., 2017). Tais dispositivos desempenham tarefas específicas e agem como interface para meio ambiente. As WSN oferecem uma estrutura física para a implementação de aplicações IoT, representando assim um elemento facilitador para o desenvolvimento de tais aplicações.

De acordo com (ALTAYEB; SHARIF; ABDELLA, 2018) as WSN e os sensores terão uma grande influência no IoT, uma vez que representam a principal forma pela qual os sistemas computacionais podem interagir com o mundo físico.

Aplicações clássicas das WSN embarcam sensores de baixa complexidade, como os de temperatura e pressão. Aplicações visuais com sensores de imagens ainda tem pouca representatividade no contexto IoT baseado em WSN. Tradicionalmente sensores de imagens ou câmeras representam custos consideravelmente maiores que os mencionados anteriormente, em termos de dados para a representação, uma imagem é muito maior que um sensor típico de uma WSN (FELEMBAN; NASEER; AMJAD, 2020).

O custo dos dispositivos tem sido um fator preponderante no desenvolvimento de aplicações IoT visto que são previstos dezenas de bilhões de elementos conectados. Na visão de (MOCNEJ, 2018), o custo do HW representa um desafio para o IoT, visto que se o custo não for baixo, o seu valor agregado não se justifica. Do ponto de vista do HW, (NAUMAN et al., 2020) define um sistema típico como alimentado por baterias ou energia solar com poucos Kb de memória e capacidade computacional limitada em MHz.

O alto grau de desenvolvimento da indústria de semicondutores tem permitido a oferta de sensores de imagem a custos cada vez menores, sendo assim o fator custo torna-se cada vez menos impeditivo para a introdução desta classe de sensores em aplicações das WSN. Tais condições tem permitido o desenvolvimento de aplicações de captura e transmissão de imagens para as WSN (ABDULLAH; ALDOORI; JAMIL, 2019).

Ainda que disponíveis a relativo baixo custo, os sensores de imagem representam um desafio técnico para o seu manuseio no contexto das WSN, levando-se em consideração as taxas de transmissão das redes e o poder computacional dos dispositivos. A transmissão de grandes arquivos de imagens através destas redes podem causar um gargalo devido a limitações de energia e desempenho das WSN (ABDULLAH; ALDOORI; JAMIL, 2019).

O estudo de técnicas de transmissão e manuseio de imagens que possam otimizar e simplificar esses processos podem contribuir para a introdução de sensores de imagem nas WSN, permitindo o desenvolvimento de novas áreas aplicações destas.

1.1 Problemática

O problema de pesquisa deste trabalho trata da transmissão de imagens em WSN, sendo a pergunta de pesquisa: Como transmitir imagens em WSN dadas as limitações deste tipo de rede?

1.2 Contribuições

Como resposta à pergunta de pesquisa, este trabalho define uma API extensível e adaptável para a transmissão de imagens em redes de sensores, permitindo a experimentação dos diversos cenários identificados durante o trabalho, sendo esta sua contribuição principal. Um protótipo de tal API é implementada na plataforma Contiki-NG/Cooja para sua validação.

1.3 Objetivos geral e específicos

O objetivo geral da pesquisa é propor uma API que facilite a experimentação de métodos de transmissão de imagens em WSNs. Baseado neste objetivo geral são definidos os seguintes objetivos específicos:

- Identificar e apresentar as dificuldades e limitações de transmissão de imagens em WSNs;
- Identificar e apresentar os métodos que permitam a transmissão de imagens em WSNs;
- Identificar e apresentar a viabilidade de uso de simuladores de WSNs na experimentação de transmissão de imagens.

1.4 Organização do Trabalho

O trabalho se iniciou na seção 1 onde foram apresentados a introdução, justificativas, problemática, contribuições e objetivos.

Na seção 2, é descrita a metodologia utilizada na pesquisa. Nesta seção são descritas a evolução, instrumentos e detalhamento da pesquisa, tratamentos dos dados e os procedimentos utilizados para projeto da API.

Na seção 3 são apresentados os elementos necessários para o embasamento teórico em WSN, sensores de imagem, tecnologias de redes sem fio e protocolos aplicáveis a IoT. São estudados também aplicações de sensores de imagens e métodos para tratamento e transmissão de imagens nas WSN.

A seção 4 apresenta o projeto da API com a representação de classe UML de seus componentes e o seu detalhamento.

No capítulo final, seção 5, são descritas a ferramenta de simulação e os procedimentos para a configuração do ambiente de simulação, bem como são exemplificados casos de uso com diagramas de sequência detalhados.

O apêndice A contém um revisão da literatura de simuladores de redes IoT, e para encerrar o trabalho no apêndice B é dado o código fonte das interfaces da API.

2 Metodologia

A pesquisa se iniciou na seleção de ingresso do programa, tendo sido solicitado a apresentação de um projeto de pesquisa. Em tal projeto de pesquisa foi proposto um tema e as suas justificativas. O trabalho formal de pesquisa se iniciou após o embasamento teórico provido nas disciplinas de metologia científica do programa, por meio das quais foram apresentadas as bases teóricas para a definição da problemática, dos objetivos e para a escrita da dissertação. Também foram apresentados os mecanismos básicos e avançados de consultas nas principais bases de publicações. Após o período de disciplinas do programa iniciou-se o período de dedicação exclusiva à pesquisa com o objetivo e um orientador definidos.

De acordo com a literatura este trabalho poder ser classificado como uma pesquisa exploratória, pois visa proporcionar maior familiaridade com o problema com vistas a torná-lo explícito ou a construir hipóteses

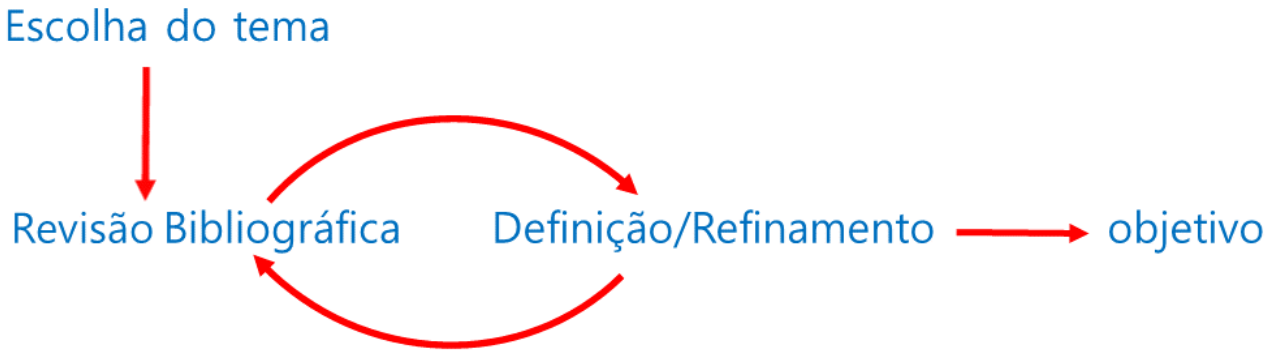
(SILVA; MENEZES, 2005). Os assuntos abordados neste capítulo foram baseados em (SILVA; MENEZES, 2005) e divididos nas seguintes tópicos:

- Evolução do trabalho, na seção 2.1;
- Instrumentos de pesquisa, na seção 2.2;
- Detalhamento da pesquisa, de modo a possibilitar a sua reprodução, na seção 2.3;
- Tratamento dos dados, na seção 2.4;
- Projeto da API e prototipação, na seção 2.5.

2.1 Evolução do trabalho

O trabalho foi fruto de várias etapas de revisão da literatura, o qual foi refinado e aprimorado ao final de cada etapa, levando-se em consideração o material obtido. De acordo com (WAZLAWICK, 2009) este mecanismo é comum ao pesquisador que não conhece exatamente a área de interesse, sendo necessário alguns ciclos de revisão da literatura até a definição do objetivo geral, como ilustrado na figura 1.

Figura 1: Processo de definição de objetivos da pesquisa.



Fonte: adaptado pelo autor, de acordo com (WAZLAWICK, 2009).

O tema de pesquisa inicialmente escolhido para a admissão no programa visava explorar algoritmos de tratamento de imagens utilizando sistemas computacionais de baixa capacidade e longa autonomia. Nesta fase, que será denominada etapa 1, o trabalho de pesquisa ocorreu de maneira não sistematizada, visto que foi feito antes da disciplinas de metodologia científica.

A próxima etapa do trabalho se iniciou com as disciplinas de metodologia do programa, o qual proveram bases teóricas para o processo formal de pesquisa. Além da estruturação da pesquisa na etapa 1, foram adicionados outros temas, com as tecnologias de redes e protocolos IoT. Tais pesquisas visavam embasar o desafio técnico do processamento de imagens em aplicações IoT.

A partir deste ponto foi iniciado a pesquisa em tempo integral com o término das disciplinas, assim como a definição da orientação do trabalho. Nesta etapa, o qual será identificado como etapa 2, foram pesquisados os seguintes temas:

- Redes e protocolos IoT;

- Técnicas de compressão de imagens para IoT;
- Sensores de imagem.

Na etapa 2, por meio das publicações obtidas, foram identificados trabalhos explorando as principais técnicas de tratamento de compressão de imagens. Foram também descobertas outras técnicas para tratamento de imagens em IoT: a detecção de mudanças, e a otimização nos dados de transmissão. Adicionalmente foram identificados trabalhos relacionados à especificação técnica de câmeras utilizadas em aplicações IoT com imagens. Tais trabalhos permitiram caracterizar tecnicamente os sensores de imagem viáveis para aplicações em IoT. Como sequência, na fase chamada aqui de etapa 3, adicionou-se também na pesquisa o tema de sensores de imagens em aplicações IoT. Esta período será denominado como etapa 3 e os temas de interesse da pesquisa foram:

- Redes IoT e protocolos;
- Técnicas de tratamento e transmissão de imagens para IoT;
- Sensores de imagem para aplicações IoT.

No evento de qualificação foi observado que na literatura redes e protocolos IoT abrangem diversas tecnologias, não sendo um consenso entre os pesquisadores a delimitação em sistemas de baixa capacidade computacional. Deste modo, como resultado deste evento o escopo do pesquisa foi delimitado as tecnologias compatíveis as WSN, propiciando a definição mais precisa dos objetivos do trabalho, que foram apresentados na seção 1.3.

Esta período será denominado como etapa 4 e os temas de interesse da pesquisa foram:

- Redes e protocolos IoT e WSN;
- Técnicas de tratamento de imagens para IoT e WSN;
- Sensores de imagem para aplicações IoT e WSN.

A evolução do trabalho de acordo com a fase do programa e o período são sintetizadas no quadro 1.

Quadro 1: Histórico de evolução do trabalho.

Etapa	**Fase do programa**	**Início**	**Final**
1	Projeto de pesquisa para ingresso no programa	Março 2019	Dezembro 2019
2	Fase de pesquisa do programa	Dezembro 2019	Janeiro 2021
3	Qualificação	Janeiro 2021	Julho 2021
4	Defesa	Julho 2021	Janeiro 2022

Fonte: elaborado pelo autor.

2.2 Instrumentos de pesquisa

Para a pesquisa bibliográfica foram utilizadas as bases de dados do ACM (ACM, 2021), IEEE Xplore (IEEE, 2021) e Scopus (ELSEVIER, 2021). As duas primeiras bases foram selecionadas devido a sua relevância em

publicações de engenharia e ciências da computação enquanto a terceira é um extenso banco de dados contendo resumos de publicações de outras bases.

Na organização dos artigos obtidos das bases de busca foi utilizado o Mendeley Desktop (MENDELEY, 2021) em sua versão para o Microsoft Windows. Segundo o portal da ferramenta existem versões para o Linux e o MacOs. Por meio de tal ferramenta também é possível realizar o compartilhamento dos artigos via nuvem.

Nos trabalhos de simulação foi utilizado o Contiki/Cooja. O Contiki-NG é um sistema operacional de tamanho reduzido que fornece suporte ao padrão IEEE 802.15.4 e vários protocolos IoT. Já o Cooja é o simulador do Contiki-NG, permitindo simular redes e aplicações desenvolvidas para o sistema.

O Contiki/Cooja são disponibilizadas por meio de licença de código aberto e os detalhes da instalação são dados na Wiki do projeto (CONTIKI-NG, 2021b). De acordo com a documentação é possível instalar nos sistema operacionais Linux e MacOs. Para o desenvolvimento do trabalho foi utilizado a distribuição Linux Ubuntu Desktop 64 bits versão 18.04. Tendo sido o Cooja desenvolvido em Java é necessário a instalação de um compilador compatível. Neste trabalho foi usada o OpenJDK versão 11.0.11.

2.3 Detalhamento da pesquisa

A pesquisa bibliografia foi realizada de acordo com as etapas apresentadas na seção 2.1, tendo sido iniciada no ano de 2019 e utilizado o critério de filtro cinco anos anteriores.

As expressões de busca procuravam por trabalhos envolvendo as tecnologias de redes e protocolos IoT e WSN, sensores de imagem, tratamento e transmissão de imagens aplicáveis a sistemas IoT e WSN. Tais expressões são sumarizadas no quadro 2 e tiveram evolução de acordo com a etapa da pesquisa:

Quadro 2: Evolução das palavras-chaves de busca do trabalho.

Etapa	**Tema**	**Expressão de busca**
1	Compressão de imagens em aplicações IoT	('Internet of Things' or Iot) and 'image compression'
2	Redes e protocolos IoT	('Internet of Things' or Iot) and protocols
	Técnicas de compressão de imagens IoT	Iot and 'image compression'
	Sensores de imagem	'image sensor' and technologies
3	Redes IoT e protocolos	('Internet of Things' or Iot) and protocols
	Tecnologias de sensores de imagem	'image sensor' and technologies
	Sensores de imagem IoT	('Internet of Things' or Iot) and ('image sensor' or camera)
	Tratamento de imagens IoT	('Internet of Things' or Iot) and image and compression
4	Tecnologias de redes IoT, WSN e protocolos	('Wireless Sensor Network' or WSN or 'Internet of Things' or Iot) and protocols
	Tecnologias de sensores de imagem	'image sensor' and technologies
	Sensores de imagem IoT e WSN	('Wireless Sensor Network' or WSN or 'Internet of Things' or Iot) and ('image sensor' or camera)
	Tratamento de imagens para IoT e WSN	('Wireless Sensor Network' or WSN or 'Internet of Things' or Iot)) and image and (transmission or technique or compression)

Fonte: elaborado pelo autor, de acordo com a etapa definida no quadro 1.

2.4 Tratamento dos dados

Após obtidos os trabalhos encontrados na pesquisa bibliográfica, foram lidos os resumos e classificados em três níveis de relevância: alta, média e baixa, como recomendado por (WAZLAWICK, 2009). O passo seguinte foi a leitura completa dos trabalhos classificados como alta relevância, e uma nova revisão dos trabalhos classificados como média relevância.

Os critérios de seleção dos trabalhos utilizados como referência da pesquisa foram:

- **Tecnologias e protocolos IoT e WSN:** Buscar característica técnicas e limitações dessas tecnologias e dos dispositivos envolvidos. Buscar também comparações e aplicações;
- **Sensores de imagem:** Buscar características técnicas e construtivas dos sensores de imagem, de forma a demonstrar a complexidade deste elementos quando comparados a sensores escalares;
- **Sistemas de captura imagem IoT e WSN:** Buscar características técnicas de dispositivos de captura utilizados em aplicações IoT e WSN, como resolução de pixels e profundidade de cores;
- **Técnicas de compressão de imagem em sistemas IoT e WSN:** Buscar técnicas de compressão de imagens usadas em aplicações IoT e WSN;
- **Técnicas de transmissão de imagem em sistemas IoT e WSN:** Buscar técnicas de transmissão otimizadas para aplicações IoT e WSN.

Como resultado final da pesquisa 522 trabalhos foram obtidos das bases de dados mencionadas na seção 2.2, sendo que 40 foram utilizados como suporte a dissertação nas suas várias etapas de escrita. A apresentação dos trabalhos de maior relevância é dada na seção 3.

2.5 Projeto da API e simulação

O projeto da API teve como requisito de entrada os elementos de rede, técnicas de compressão e transmissão de imagens identificados na seção 3 de revisão da literatura. Para o projeto da API definiu-se como características a facilidade de adaptação e extensão. Tais características são importantes pois se detectou na literatura que as técnicas são utilizadas com diversas variações de implementação e em vários arranjos.

Estando a API projetada, iniciou-se a implementação de um protótipo no ambiente Contiki/Cooja. Tal protótipo foi implementado em linguagem C e compilado como aplicação no sistema Contiki-NG. Adicionalmente, uma interface foi criada no Cooja de forma a facilitar a interação com a simulação.

Na etapa de simulação todos os componentes de transmissão, rede, fragmentação e manuseio de imagens estavam plenamente funcionais, entretanto os componentes que encapsulam as técnica de tratamento de imagens, implementam somente as interfaces de entrada e saída.

De forma a validar as interfaces da API, assim como a integração com o simulador e seus elementos, foi utilizado a técnica de prototipação. Foram também implementados e testados no protótipo os casos de uso mais importantes, como forma de refinar o projeto dos componentes e a integração com o simulador.

3 Revisão da literatura

Nesta seção é apresentada a revisão da literatura deste trabalho, com o objetivo de prover o embasamento teórico para a pesquisa, de acordo com os objetivos definidos na seção 1.3. Foram selecionados trabalhos compatíveis aos temas definidos e que pudessem apoiar o desenvolvimento da pesquisa.

O tema IoT tem uma larga abrangência e envolve múltiplas áreas de pesquisa, tais como tecnologias de redes, protocolos, dispositivos inteligentes, plataformas de HW, redes de sensores entre outros. No contexto deste trabalho, e de acordo com os objetivos da pesquisa são estudados os seguintes temas:

- Redes de sensores sem fio;
- Características técnicas de sensores de imagem;
- Tecnologias de redes sem fio e protocolos IoT;
- Técnicas de tratamento e transmissão de imagens para uso em WSN.

Tais temas são abordados nas sub-seções seguintes.

3.1 Redes de sensores sem fio

As WSN são estruturas de rede sem fio que embarcam elementos com funções de escopo específico, usualmente como alguma função de sensoreamento e atuação, agindo como interface entre a natureza e o mundo virtual. De acordo com (LIN et al., 2017) as WSN podem monitorar condições ambientais como temperatura, som, vibração, pressão entre outros. Tais redes são um subsistema dentro do IoT possibilitando o desenvolvimento de aplicações em diversas áreas:

- **Aplicações militares:** As WSN são parte integrante no comando das forças armadas, no controle, nas comunicações, na computação, na inteligência, na vigilância do campo de batalha e nos sistemas de reconhecimento e direcionamento (MATIN; ISLAM, 2012). No trabalho de (ZEITZ et al., 2017) são estudas aplicações na área militar, de acordo com os autores o campo de batalha zona se tornará uma rede de informações repleta de sistemas e dispositivos, todos em comunicação para auxiliar no compartilhamento de informações, tomada de decisões e controle de armas.
- **Monitoramento de áreas:** Os nós sensores são implantados em regiões onde algum fenômeno deve ser monitorado. Quando os sensores detectam o evento que esta sendo monitorado (calor, pressão etc), o evento é relatado a uma das estações base, que então toma a ação apropriada (MATIN; ISLAM, 2012).
- **Transporte:** As informações de tráfego em tempo real são coletadas via WSN para a alimentação de sistemas de transporte que alertam os motoristas sobre congestionamentos e problemas de tráfego (MATIN; ISLAM, 2012).
- **Aplicações de Saúde:** Algumas das aplicações de saúde das redes de sensores dão suporte a pessoas com deficiência, integrando o monitoramento destes pacientes, o diagnósticos e a administração de medicamentos em hospitais, o tele-monitoramento de dados fisiológicos humanos. Dentro de um hospital o monitoramento de pacientes e médicos (MATIN; ISLAM, 2012).
- **Detecção ambiental:** O termo Redes de Sensores Ambientais foi desenvolvido para abranger muitas aplicações de WSN para pesquisas de fatores geológicos e ambientais. Isso inclui detecção de vulcões, oceanos, geleiras, florestas, etc. Algumas outras áreas principais são: poluição do ar, detecção de fogo em floretas, monitoração de estufas e detecção de deslizamento de terras (MATIN; ISLAM, 2012).
- **Monitoramento estrutural:** Sensores sem fio podem ser utilizados para monitorar o movimento dentro de edifícios ou de uma infraestrutura, como pontes, viadutos, aterros, túneis, etc., permitindo práticas de engenharia para monitorar ativos remotamente, evitando assim os custos de visitas ao local (MATIN; ISLAM, 2012).

- **Aplicações em agricultura:** Utilizando-se uma rede de sensores sem fio, a manutenção de fiação não é necessária, sendo vantajoso para locais de difícil acesso. A automação da irrigação permite um uso mais eficiente da água e reduz o desperdício (MATIN; ISLAM, 2012). Em (CHEN; EAGER; MAKAROFF, 2019) é proposto um sistema para a coleta de imagens numa área de cultivo de canola utilizando a tecnologia Lora, que permite a transmissão de dados por até alguns quilômetros.

3.1.1 Estrutura de nós

Usualmente em engenharia um sensor significa um elemento transdutor, que tem a característica de converter uma grandeza física para outra que possa ser mensurável, (JAVAID et al., 2021) os define como elementos que capturam e traduzem seus atributos físicos em impulsos elétricos observáveis. Diversos tipos de sensores fazem parte do nosso cotidiano estando presentes em variadas aplicações, os tipos mais comuns de sensores utilizados em IoT e nas WSN segundo (PATEL; KUMAR, 2018) e (JAVAID et al., 2021), e suas aplicações são dados a seguir:

- **Sensores de temperatura:** São utilizados principalmente para controle de ar condicionado, freezers e outros dispositivos de controle ambiental. A medição da temperatura do solo é crucial para o crescimento das culturas na agricultura. Ajuda as plantas a se desenvolverem corretamente, permitindo uma produção ideal (JAVAID et al., 2021).

- **Sensores de pressão:** Medem a pressão de líquidos, gases ou outras formas de pressão, são usados em uma variedade de dispositivos. Esses sensores permitem a criação de sistemas IoT que monitoram sistemas e dispositivos acionados por pressão. Qualquer variação da faixa de pressão típica alerta o administrador do sistema sobre quaisquer problemas que precisem ser resolvidos (JAVAID et al., 2021).

- **Sensores de proximidade:** São comumente utilizados nos negócios de varejo (JAVAID et al., 2021).

- **Sensores de umidade:** Medem a quantidade de vapor de água em um ambiente de ar ou outros gases, que é conhecida como umidade, umidade relativa é o termo mais usado. Suas aplicações e usos podem ser encontrados nos domínios industrial e residencial para o controle de sistemas de aquecimento, ventilação e ar condicionado (JAVAID et al., 2021).

- **Sensores de fumaça:** Detectam fumaça (partículas e gases no ar), bem como sua quantidade. Eles já existem há algum tempo, e agora são ainda mais eficazes, graças ao advento do IoT, pois estão conectados a um sistema que avisa rapidamente o usuário sobre quaisquer problemas que surjam em um ambiente (JAVAID et al., 2021).

- **Sensores de infra-vermelho:** emitem ou detectam radiação infravermelha para perceber características particulares de seus arredores. Eles também podem detectar e medir o calor irradiado pelos itens. Atualmente, eles estão sendo empregados em uma série de projetos de IoT, principalmente na área da saúde, porque simplificam o monitoramento do fluxo sanguíneo e da pressão arterial (JAVAID et al., 2021).

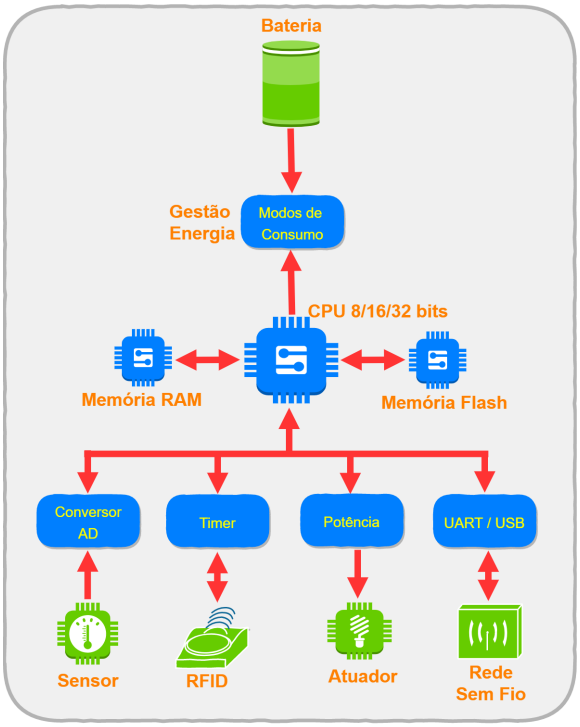
- **Aceleração ou inclinação:** Um acelerômetro é um tipo de transdutor que converte movimento mecânico em sinais elétricos medindo a aceleração real ou quantificável que um item experimenta devido a forças inerciais. É a taxa na qual a velocidade varia em relação ao tempo. Esses sensores agora são encontrados em milhões de produtos, incluindo smartphones, detecção de vibração, inclinação e aceleração são apenas alguns de seus usos (JAVAID et al., 2021).

No contexto do IoT e das WSN, o termo sensor ou nó tem um significado mais amplo que apenas um transdutor. O sensor se refere a um conjunto de elementos de HW. Isso é necessário pois além de medir uma grandeza física em si, os nós também precisam de capacidade de comunicação em rede. Segundo (PATEL; KUMAR, 2018) tais elementos são:

- CPU ou unidade central de processamento;
- Memória *Flash* e RAM, para armazenamento e execução dos programas;
- Conversor digital para analógico (AD);
- Temporizadores para contagem de tempo entre eventos;
- Dispositivos para gestão de energia que possibilitem ligar e desligar circuitos internos afim de poupar energia;
- Circuitos de potência de modo que um LED ou um buzzer possam emitir sinais visuais ou auditivos;
- Algum tipo de Interface de comunicação, serial ou USB por exemplo, de modo a implementar a comunicação com a interface de rede sem fio;
- Interface de rede sem fio, como o padrão IEEE 802.15.4, o WiFi e o Lora entre outros.

De forma a padronizar a linguagem do trabalho daqui adiante o conjunto de elementos de HW de um sensor WSN será denominado CSN (Componente Sensor da WSN), a figura 2 mostra um diagrama de um CSN com seus elementos típicos de HW.

Figura 2: Elementos básicos de HW de um CSN.

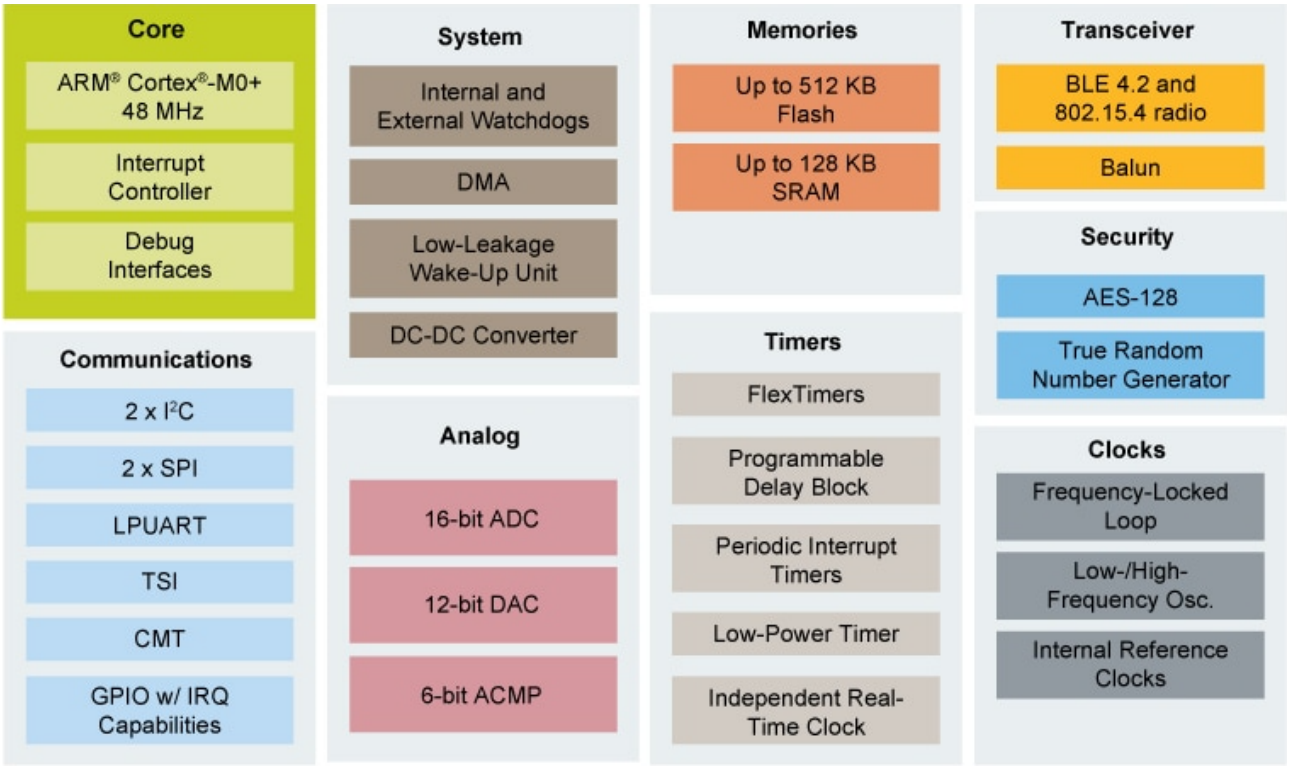


Fonte: adaptado pelo autor, de acordo com (PATEL; KUMAR, 2018).

A indústria de semicondutores oferece chips com grande parte dos elementos de HW embarcados num único encapsulamento, denominado microcontrolador, o qual facilita o projeto de um CSN. Como exemplo, a NXP® Semicondutores oferece a linha Kinetis MKW41Z (NXP, 2020a) que traz uma série de periféricos integrados a CPU e a memória, como mostrado na Figura 3. Nesta família de chips estão integrados os seguintes elementos:

- CPU Arm Cortex M0, rodando à 48 MHz;
- Memória flash de até 512 Kb e Ram de até 128 Kb;
- Conversores AD de 16 e 12 bits;
- Temporizadores;
- Barramento UART e I2C;
- Módulo de rádio Bluetooth Low Energy (BLE) 4.2;
- Módulo de rádio 802.15.4;
- Interface para tela de toque.

Figura 3: Visão geral de periféricos da linha Kinetis MKW41Z.



Fonte: (NXP, 2020a).

Do ponto de vista do projeto de HW de um CSN, dependendo da especificação do projeto e utilizando-se um chip similar ao MKW41Z serão necessários poucos elementos adicionais, permitindo assim projetos de baixo

custo, visto que uma unidade da linha Knetis MKW41Z com 256 Kb de Flash tem o custo de 2.52 dólares a unidade (NXP, 2020b).

Pelas especificações técnicas deste componente, com memória de apenas algumas centenas de kilobytes, fica claro a limitação do seu poder computacional, limitando também as aplicações nele embarcadas. Outras sistemas computacionais mais poderosos são possíveis, todavia haverá aumento do custo, do consumo de energia e da complexidade do projeto.

Além das questões de limitação de custo e do poder de processamento, o consumo de energia é outro fator limitante no projeto das WSN (MATIN; ISLAM, 2012), visto que os CSN podem ser alimentados por bateria.

3.1.2 Organização lógica da rede de sensores sem fio

Logicamente, os sensores de uma rede podem assumir funções específicas na organização da rede, e se conectam formando diversos tipos de topologias. Como exemplo, numa rede ZigBee, encontramos os seguintes tipos de nós (KUMAR; MANE, 2017):

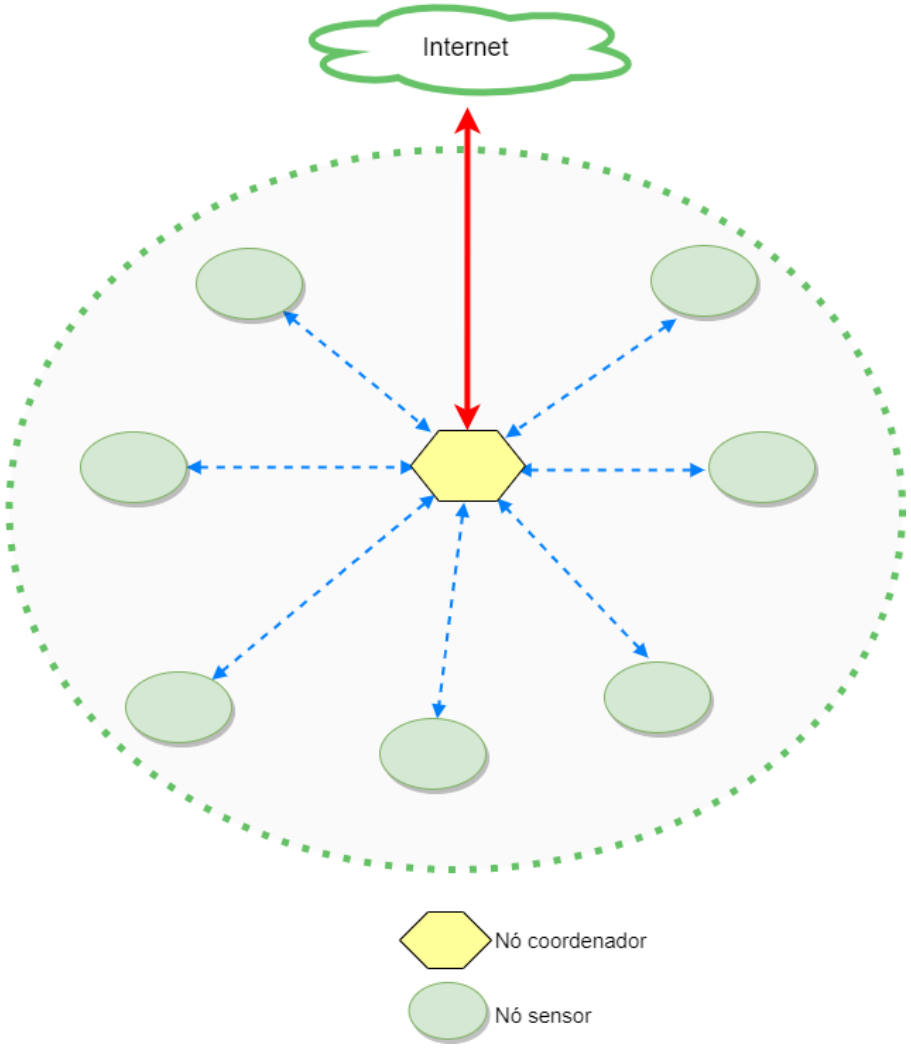
- **Coordenador:** É o responsável pela rede, incluindo a inicialização e a manutenção;
- **Roteador:** São usados para estender o alcance da rede, adicionalmente são responsáveis pelo controle de fluxo de controle da dados utilizando estratégias de roteamento hierárquico na rede;
- **Sensor:** Tem a função somente de sensor e de conexão a um coordenador ou roteador.

A função dentro da topologia da rede não implica necessariamente que um nó roteador ou coordenador não possam acumular a função de sensor simultaneamente. Em todos os tipos de topologias é possível acessar o mundo externo conectando-se o coordenador a uma outra rede, com acesso a internet por exemplo.

Do mesmo modo, tomando o ZigBee como exemplo, podemos ter as seguintes topologias de rede (KUMAR; MANE, 2017):

- **Topologia estrela:** Nesta topologia, o coordenador é responsável por toda a rede, os sensores só podem se comunicar diretamente com o coordenador, esta topologia é adequada para aplicações com requisito de tempo crítico. Na figura 4 é apresentada uma ilustração da topologia estrela.

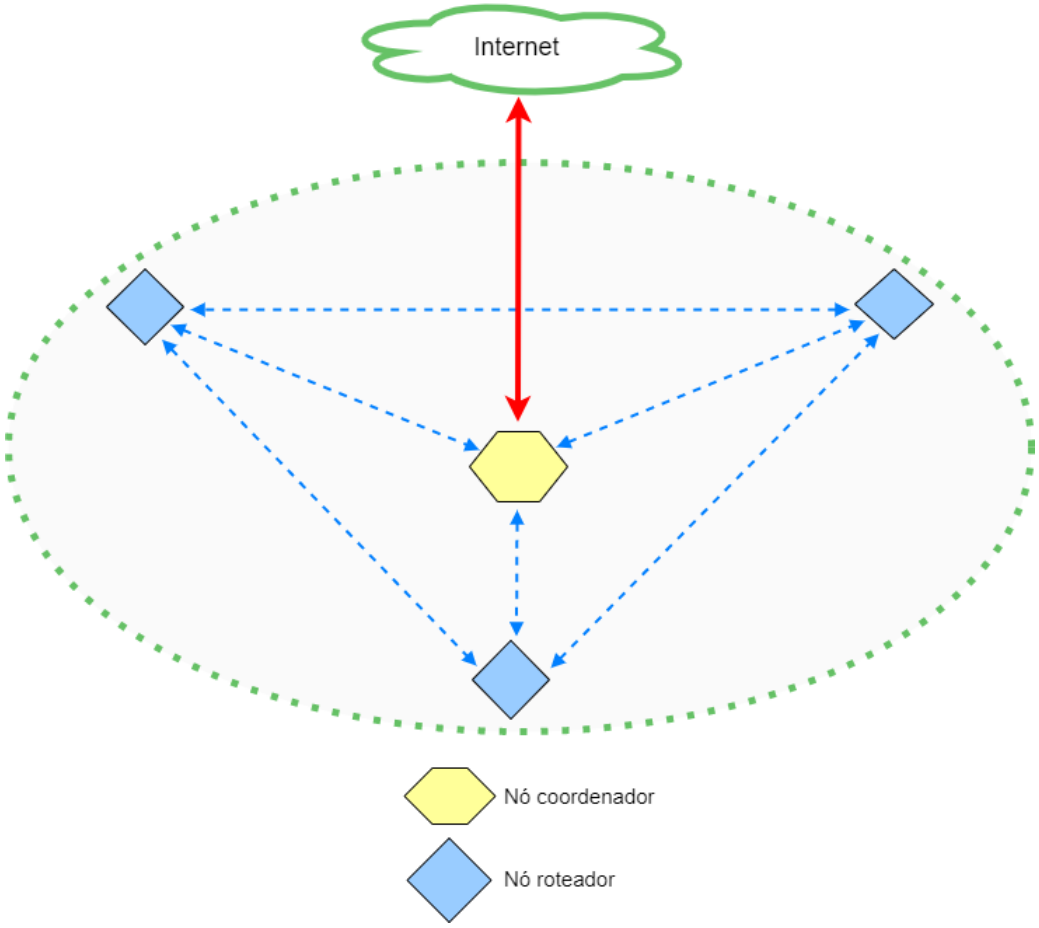
Figura 4: Topologia em estrela numa rede ZigBee.



Fonte: adaptado pelo autor, de acordo com (KUMAR; MANE, 2017).

- **Topologia em malha:** Numa rede em malha todos os nós roteadores são permitidos se comunicarem entre si, o roteamento é descentralizado, possibilitando-se o roteamento alternativo quando um roteador falha. Roteadores são utilizados para estender o alcance da rede e integrar os sensores. Também é necessário a função do coordenador, a representação da topologia em malhada é dada na figura 5.

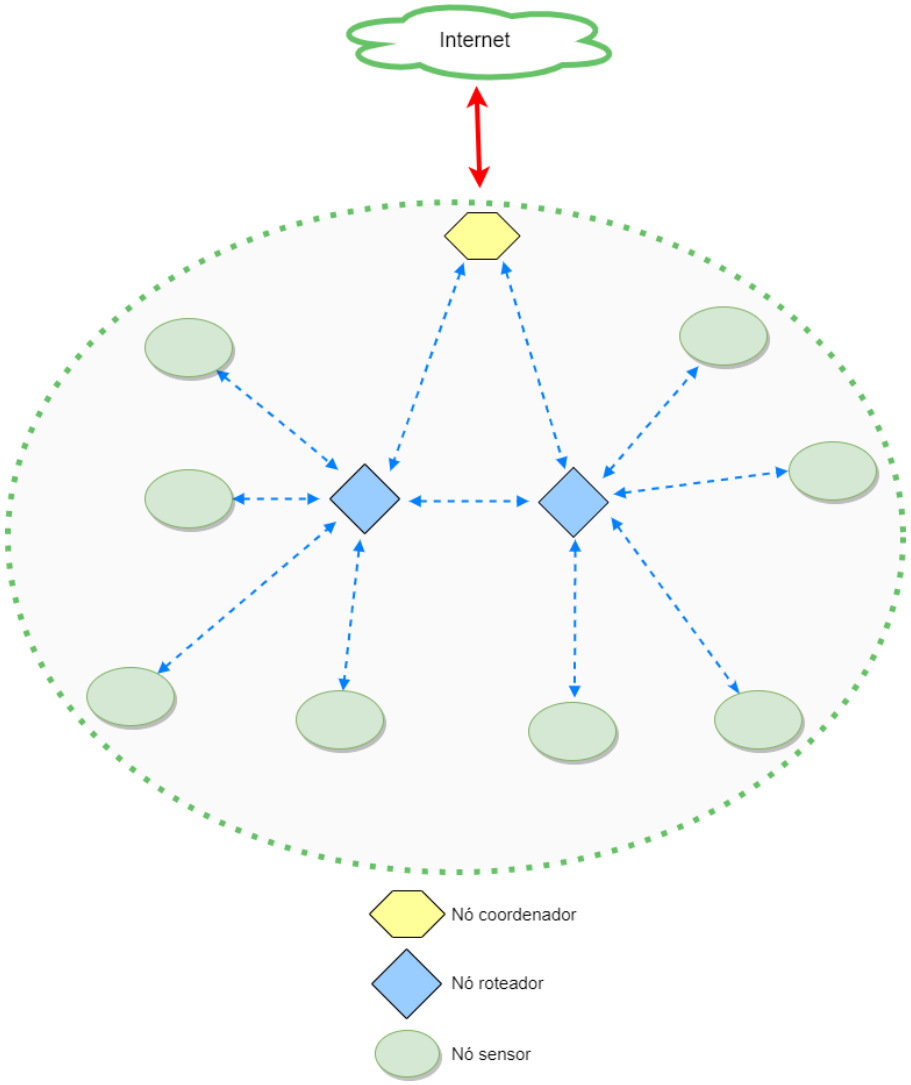
Figura 5: Topologia em malha numa rede ZigBee.



Fonte: adaptado pelo autor, de acordo com (KUMAR; MANE, 2017).

- **Topologia em grupo de árvore ou mista:** São redes híbridas que permitem sub-redes de roteadores interligados a um coordenador. Uma rede mista e ilustrada na figura 6.

Figura 6: Topologia em grupo de árvores ou mista numa rede ZigBee.



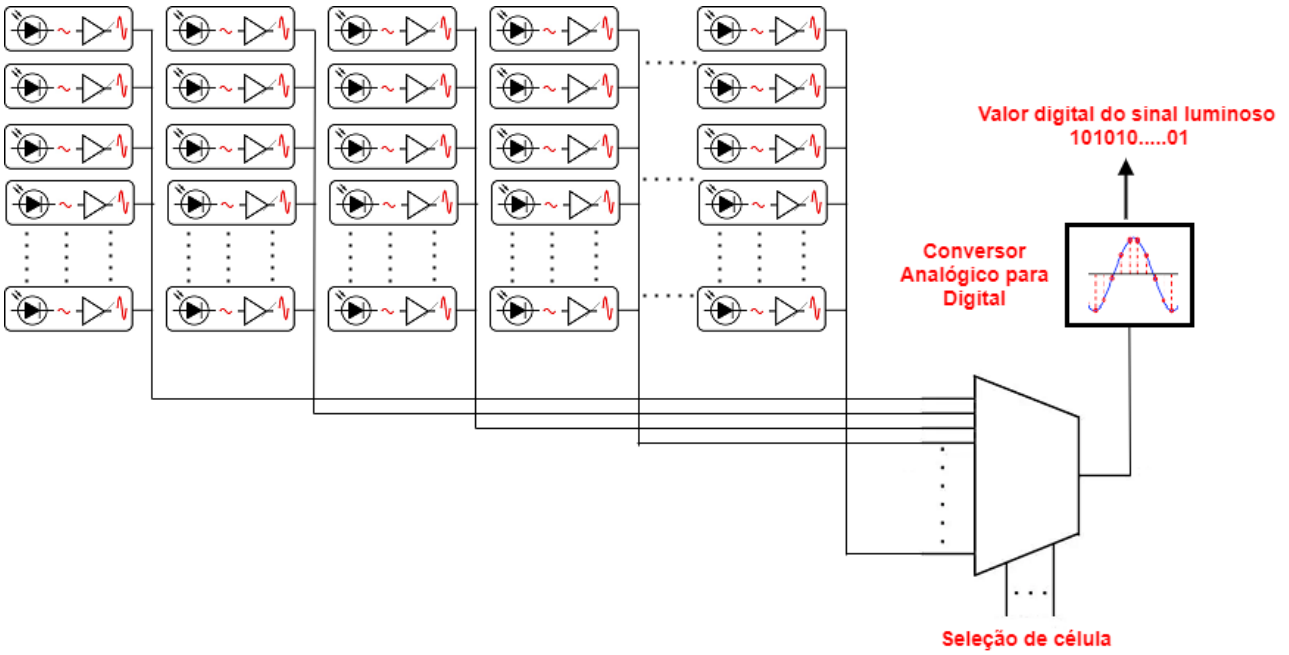
Fonte: adaptado pelo autor, de acordo com (KUMAR; MANE, 2017).

3.2 Sensores de imagem

Sensores de imagem são os elementos principais de um sistema de captura de imagens e possuem a característica de transformar a luz em sinais elétricos. Um sensor de imagem é composto de um agrupamento de várias células sensitivas de luz. Tais células são distribuídas espacialmente em forma de matriz, e são chamadas também de pixel.

O princípio de funcionamento de um sensor de imagem é a conversão do fóton (luz) em estímulo elétrico, por meio de um elemento chamado fotodiodo. A tensão produzida pelo fotodiodo é amplificada e convertida para uma grandeza digital. O número de células presentes no sensor determina a resolução do sensor. Geralmente num espaço de milímetros quadrados são dispostas N x M células, definindo assim a resolução do sensor. Baseado em (MEHTA; PATEL; MEHTA, 2015) é apresentado na figura 7 o diagrama genérico de um sensor de imagem.

Figura 7: Diagrama genérico de um sensor de imagem.



Fonte: adaptado pelo autor, de acordo com (MEHTA; PATEL; MEHTA, 2015).

Atualmente o *Complementary Metal Oxide Semiconductor* (CMOS) e o *Charge Coupled Device* (CDD) são duas das principais tecnologias de construção de sensores de imagens. No trabalho de (MEHTA; PATEL; MEHTA, 2015) são discutidas as duas tecnologias. Enquanto o CCD permite obter imagens com qualidade superior, o CMOS por sua vez tem menor custo. Adicionalmente, (MEHTA; PATEL; MEHTA, 2015) mencionam também as seguintes vantagens do CCD sobre o CMOS:

- No sensor de imagem CMOS, o tempo para se obter uma imagem é de 1/30 segundo, enquanto no sensor de imagem CCD é de 1/60 segundo para uma imagem;
- O CCD possui um conversor de carga para tensão por sensor, enquanto o CMOS possui apenas um por pixel;
- A qualidade da imagem do sensor de imagem CMOS é menor que sensor de imagem CCD;
- O desfoque é o principal problema na imagem do sensor CMOS, problema não observado no sensor de imagem CCD;
- A velocidade de captura de uma imagem é maior no CCD que no CMOS;
- O sensor de imagem CCD é mais sensível à luz que o sensor de imagem CMOS.

Visto que um sensor de imagem é uma matriz de M x N sensores individuais, uma imagem será então representada por uma matriz M x N de valores escalares.

A introdução de sensores de imagem em aplicações IoT traz um grande impacto no desenvolvimento de software, no HW necessário e também nas redes. Quando comparado as aplicações com sensores que fornecem

um único valor, conhecidos como sensores escalares, os sensores de imagem representam uma quantidade muito maior de dados para serem processados e trafegados pela rede. Como exemplo, um sensor de imagem com resolução de 640 x 480 pixeis pode ser comparado a um sistema com 307.200 de sensores escalares.

Como já exposto anteriormente, dispositivos IoT são projetados com componentes dotados de pouca memória e baixo poder computacional que se conectam em redes de baixa velocidade, neste contexto a utilização de sensores de imagem se torna desafiador.

Levando-se em consideração o sensor de imagem, quanto menor a sua resolução maior será a viabilidade de implementação em um sistema IoT. Por outro lado resoluções muito pequenas poderão resultar em imagens que não contenham informações viáveis para análise. Desta forma a revisão da literatura poderá nos ajudar na identificação dos dispositivo de imagem disponíveis para aplicações IoT.

Identificar tais padrões de resolução de imagens serão importantes na estimativa de tamanhos de arquivos que serão manuseados pelos dispositivos assim com o trafego de dados na rede, visto que a resolução impacta diretamente no bloco de dados gerados pelo sensor.

No trabalho de (COSTA, 2020) foram pesquisadas plataformas de HW utilizadas em aplicações de redes de sensores visuais (VSN), auxiliando assim na definição dos padrões a serem considerados neste trabalho. Nesta pesquisa foram identificados dispositivos com sensores de imagem com os seguintes características mínimas de resolução:

- Resolução VGA: 640x480 pixels;
- Resolução QVGA: 320x240 pixels;
- Resolução CIF: 352x240 pixels.

Além da resolução em si, outro fator envolvido na quantidade de dados gerados por um sensor de imagem é a profundidade de cores gerada pelos sensores, como mostrado nos padrões descritos abaixo:

- **1 bit por pixel:** imagens em preto e branco. Uma imagem em resolução VGA e nesta profundidade terá o tamanho de 38.4 kbytes;
- **4 bits por pixel:** imagens com 16 cores, tipicamente representadas com 16 tons de cinza. Uma imagem em resolução VGA e nesta profundidade terá o tamanho de 153.6 Kbytes;
- **8 bits por pixel:** imagens com 256 cores, tipicamente representadas com 256 tons de cinza. Uma imagem em resolução VGA e nesta profundidade terá o tamanho de 307.2 Kbytes;
- **24 bits por pixel:** imagens com 16.777.216 cores, padrão conhecido como *true color*, capaz de representar a quantidade máxima de cores perceptível pelo sistema visual humano. Uma imagem em resolução VGA e nesta profundidade terá o tamanho de 921.6 Kbytes.

Finalmente, é possível concluir pelo trabalho de (COSTA, 2020) que neste tipo de aplicação, todos os dispositivos de imagem são sensores de imagem CMOS, que como visto anteriormente tem menor custo que os sensores baseados em CCD.

Analisando-se os dispositivos de captura de imagem comercialmente disponíveis apresentados por (COSTA, 2020), conclui-se que eles embarcam diversos componentes internos além da matriz de pixels, em alguns casos a imagem é até mesmo compactada internamente. A uCAMIII (4DSYSTEMS, 2020) e a Omnivision 7675 (OMNIVISION, 2021) são dois dispositivos representativo para as aplicações VSN. Estes dispositivos apresentam as seguintes características:

uCAMIII (4DSYSTEMS, 2020):

- Embarca um sensor VGA;
- Formatos: JPEG ou imagem pura;
- Profundidade de cores: 8 bits em tons de cinza ou colorida em 16 bits (Azul, Verde e Vermelho);
- Resoluções suportadas em imagem pura: QQVGA, 128x128, 128x96 e 80x60;
- Resoluções suportadas em Formato JPEG: VGA, QVGA ou 160x128.

OV7675 (OMNIVISION, 2021):

- Embarca um sensor VGA;
- Formato: Imagem pura;
- Profundidade de cores: 5 bits Vermelho, 6 bits Verde e 5 bits Azul (2 bytes / pixel);
- Resoluções suportadas: VGA, QVGA e QQVGA.

3.3 Tecnologias de redes sem fio e protocolos IoT

Com já exposto na seção 3.1.1 a interface de rede sem fio é um componente mandatório em um CSN segundo(PATEL; KUMAR, 2018), logo a revisão da literatura das tecnologias de rede sem fio aplicáveis a IoT é de grande relevância na identificação das tecnologias mais adequadas às aplicações visuais.

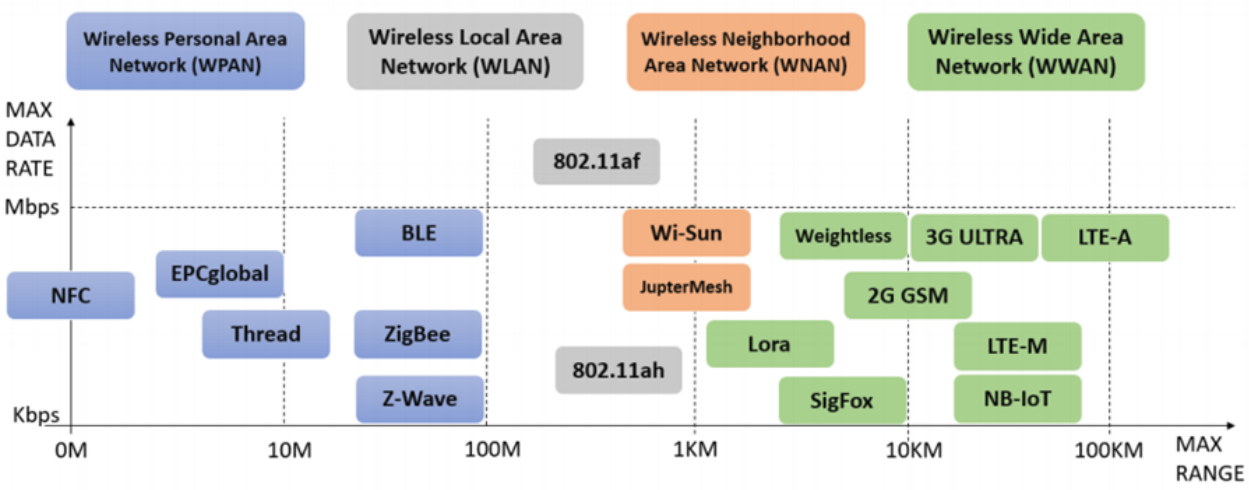
Outra característica do IoT é a diversidade de tecnologias de rede sem fios, em (MOCNEJ, 2018) são apresentadas comparativamente 22 tecnologias de redes IoT e suas características. Tais tecnologias de redes sem fio podem atender distâncias desde alguns centímetros até algumas dezenas de quilômetros (MOCNEJ, 2018). Além disso, as taxas de transmissão podem variar de alguns kilobytes por segundo até alguns megabytes por segundo (MOCNEJ, 2018).

Em (NOVELLI et al., 2018) há uma análise das principais tecnologias móveis aplicáveis a IoT, assim como o alcance e taxas de transmissão. Tais informações são apresentadas na Figura 8.

Considerando-se o alcance, os autores dividem as redes móveis em quatro grupos:

- **WPAN:** Wireless Personal Area Network (NOVELLI et al., 2018);
- **WLAN:** Wireless Neighboardhood Area Network (NOVELLI et al., 2018);
- **WNLAN:** Wireless Neighboardhood Area Network (NOVELLI et al., 2018);
- **WWAN:** Wireless Wide Area Network (NOVELLI et al., 2018).

Figura 8: Principais tecnologias de redes móveis utilizadas em IoT, alcance e taxas de transmissão.



Fonte: (NOVELLI et al., 2018).

As redes WPANs ou redes móveis pessoais, são redes de curto alcance, tipicamente conectam dispositivos portáteis ou móveis. Em (NOVELLI et al., 2018), são consideradas as seguintes tecnologias de rede móvel nesta categoria:

- **NFC:** Near Field Communication é um protocolo de comunicação de curto alcance que proporciona comunicação fácil e segura. É baseado nos padrões ISO 14443, ISO 18092 e FeliCa. Opera em 13.56 MHz, podendo atingir até 10 cm de distância e com taxa de máxima de transmissão de dados de 424 Kb/s (NOVELLI et al., 2018).
- **EPCglobal:** A tecnologia chamada de Código Eletrônico de Produto (EPC) pode transportar uma larga quantidade de informações. Dispositivos com esta tecnologia podem ser lidos em distâncias de até 10 m, mesmo quando não visível. O EPCglobal pode ser aplicado para identificação de dispositivos e comunicação passiva. O protocolo EPC define as camadas físicas e lógicas para o retro-espelhamento passivo (passive-backscatter) de energia e para o mecanismo de comunicação. É um sistema de identificação por rádio frequência (RFID) que opera na faixa de frequência de 860 a 920 MHz (NOVELLI et al., 2018).
- **ZigBee:** É baseado no padrão IEEE 802.15.4 e composto de quatro camadas principais: física, de enlace, de rede e camada de aplicação. Também é fornecida uma camada de segurança. Opera nas frequências de 868 MHz (Europa) , 915 MHz (EUA) ou 2,4 GHz (Austrália). Pode atingir até 20 metros de distância e com taxa máxima de transmissão de dados de 900 Kb/s (NOVELLI et al., 2018).
- **Thread:** É uma customização do protocolo ZigBee para fornece uma comunicação confiável, econômica, de baixo consumo de energia e de dispositivo a dispositivo sem fio. O Openthread é uma implementação Thread v1.1.1 lançado pelo Nest Labs como uma implementação de código aberto, o padrão define o IEEE 802.15.4 como tecnologia de rede sem fio, camada MAC com segurança, estabelecimento de link e roteamento em malha (GOPALUNI et al., 2019).
- **Z-Wave:** É um padrão que tem definido a camada de rede e de endereçamento pelo recomendação ITU-T G.9959. o Z-Wave tem como principal proposta permitir a transmissão confiável de uma mensagem curta

de uma unidade de controle para um ou mais nós da rede. Opera em 900MHz ou 2,4 GHz e com taxa máxima de transmissão de dados de 200 Kb/s (NOVELLI et al., 2018).

- **BLE:** Bluetooth Low Energy fornece uma solução em cenários onde uma solução de salto único (single-hop) é necessária, com semelhanças de implantação do Bluetooth. Opera na frequência de 2,4 GHz, pode atingir até 50 metros e com taxa máxima de transmissão de dados de 1Mbps (NOVELLI et al., 2018).

As redes WLAN, ou redes sem fio locais, usualmente conectam dispositivos a internet sem a necessidade do cabeamento. Em (NOVELLI et al., 2018), são consideradas as redes derivadas do padrão IEEE 802.11.

O IEEE 802.11 (STANLEY, 2019) ou Wi-Fi (ALLIANCE, 2019) é um padrão de rede sem fio largamente utilizado em residências para a conexão de computadores pessoais e dispositivos móveis à internet. No contexto IoT as futuras redes domésticas incluirão uma diversidade de dispositivos, tais como laptops, smartphones, sensores e dispositivos inteligentes. Esses dispositivos irão operar usando protocolos IoT padrão, tais como o CoAP, MQTT e o HTTP RESTful. As redes WiFi poderão fornecer um elo de conexão entre sub-redes e o núcleo de rede primário que suportarão a conectividade onipresente necessária para dispositivos IoT (POKHREL; WILLIAMSON, 2018).

De modo a melhorar o consumo e o alcance das redes IEEE 802.11 foram definidas as seguintes variantes:

- **IEEE 802.11.ah:** É um padrão que visa estender o alcance do padrão Wi-Fi assim como diminuir o consumo de energia. Opera na faixa sub Ghz de 900 MHz não licenciada, pode atingir até 1Km de cobertura em áreas externas, e taxa máxima de transmissão de dados de 100 Kb/s (NOVELLI et al., 2018).

- **IEEE 802.11.af:** Similar ao padrão IEEE 802.11.ah, se utiliza da faixa de espectro licenciado de 54 MHz e 790 Mhz em localidades aonde este espectro não está alocado para transmissões de TV aberta. Pode atingir até 1Km e taxa máxima de transmissão de 1 Mb/s (NOVELLI et al., 2018).

As redes WNLAN, ou redes sem fio de vizinhança, são redes sem fio que podem interligar uma pequena vizinhança, englobando algumas pequenas redes residenciais HAN (Home Area Network) vizinhas, no contexto das WSN as WNANs são redes sem fio que interligam sub redes de objetos inteligentes. Em (NOVELLI et al., 2018), são consideradas as seguintes tecnologias de rede móvel nesta categoria:

- **Wi-Sun:** Baseado no padrão IEEE 802.15.4, para aplicações de redes inteligentes de energia, esta em operação na Coréia do Sul e no Japão (NOVELLI et al., 2018).

- **JupterMesh:** Também baseado no padrão IEEE 802.15.4, objetiva atingir taxas de transmissão máxima de até 800 Kb/s. Implementa o roteamento IPV6 incluindo o 6LoWPAN e o RPL (NOVELLI et al., 2018).

Finalmente, as redes WWAN, ou redes sem fio de longo alcance, são as redes que podem interligar áreas extensas, como as redes de telefonia móvel. Em (NOVELLI et al., 2018), são consideradas as seguintes tecnologias de rede móvel nesta categoria:

- **SigFox:** É uma tecnologia destinada para ambientes aonde os requisitos de taxa de dados, carga útil e troca de mensagens são muito baixos, em área de cobertura de até alguns quilômetros. Consiste em uma solução proprietária UNB, opera nas bandas de 869 MHz na Europa e 915 MHz na América do Norte (NOVELLI et al., 2018).

- **Lora:** Long Range é uma tecnologia desenvolvida pela Semtech, objetiva longo alcance e baixo consumo de energia. Opera nas bandas de 433 MHz, 915 MHz e 923 MHz, pode atingir até 10 Km de distância e com taxa máxima de transmissão de dados de 50 Kb/s (MOCNEJ, 2018).

- **2G GSM:** Tecnologias de segunda geração de telefonia celular, baseadas no padrão GSM e definidas pela 3GPP, tais como o GPRS e o EGPRS, podem cobrir dezenas de quilômetros, operam em várias faixas do espectro licenciado (NOVELLI et al., 2018). As taxas de transmissão máximas de dados são de 21.4 kb/s para o GPRS, enquanto para o EGPRS é de 59.2 kb/s por canal (MOCNEJ, 2018).

- **3G Ultra:** Tecnologias de transferência de dados de terceira geração de telefonia celular, tais como UMTS e o HSPA. Assim como o 2G operam em diversas faixas do espectro licenciado de frequências, pode atingir mais de 10 Km de distância e com taxas de transmissão máxima de 14 Mb/s (HSPA) na direção do móvel para a estação de rádio. Comparativamente ao 2G, o 3G tem maior consumo de energia assim como o custo do modem, sendo que usualmente aplicações IoT são atendidas pelo taxas de transmissão de dados do 2G que o coloca como uma opção vantajosa frente ao 3G (NOVELLI et al., 2018).

- **Weightless:** Nesta tecnologia são definidos dois padrões o Weightless-N e o Weightless-P, o Weightless-N é um padrão UNB para comunicação unilateral de um nó para a estação rádio base, opera nas bandas sub-GHz, pode atingir até 5km e com taxa máxima de transmissão de dado de 100Kb/s. O Weightless-P oferece dois PHYs não proprietários com conectividade bidirecional, assim permite aumentar a confiabilidade utilizando protocolos de reconhecimento, atingindo uma taxa máxima de transmissão de dados de 100 Kb/s (NOVELLI et al., 2018).

- **LTE / LTE-A:** São tecnologias de comunicação móvel de quarta geração, operam em várias faixas da espectro licenciado de frequências, assim como o 2G e 3G podem atingir dezenas de Km e com taxas máximas de transmissão de dados de dezenas de Mb/s. o LTE excede aos objetivos das WSN tanto em taxas de transmissão quanto em custo e consumo de energia, sendo assim a 3GPP trabalhou em versões mais simples destes padrões de forma a atender assim aos requisitos de aplicações IoT (NOVELLI et al., 2018).

- **Nb-IoT / LTE-M:** São simplificações dos padrões 4G LTE de forma a atender aplicações IoT, diminuindo assim os custos, o consumo de energia e também as taxas de transmissão de dados. Elas Podem coexistir com os demais padrões de telefonia e se utilizam de faixas licenciadas do espectro, podem atingir até dezenas de quilômetro e com taxas máximas de transmissão de dados de 180 Kb/s no Nb-Iot e de 4 Mb/s no LTE-M (MOCNEJ, 2018).

Além do alcance e taxa de transmissão, outra característica importante das redes sem fio aplicáveis ao IoT é o consumo de operação destas. Levando-se em consideração o consumo de energia as redes podem ser divididas nos seguintes grupos (RAZA; KULKARNI; SOORIYABANDARA, 2017):

- **LPWAN ou LPWA:** Low Power Wide Area Networks;

- **LR-WPAN:** Low-Rate Wireless Personal Area Networks;

- **WLAN:** Wirelles Local Network (já visto anteriormente).

As tecnologias LPWAN (Low Power Wide Area) objetivam atender grande área de cobertura, baixo consumo e simplicidade a baixo custo. De acordo com (RAZA; KULKARNI; SOORIYABANDARA, 2017) as redes LPWA representam um novo paradigma da comunicação, que complementará a tradicional rede celular e as tecnologias sem fio de curto alcance no tratamento de diversos requisitos em aplicações IoT. Tecnologias LPWA oferecem um conjunto exclusivo de recursos, incluindo conectividade de área ampla para dispositivos de baixa energia e baixa taxa de dados, não fornecidos por tecnologias sem fio tradicionais.

Dentre as tecnologias de rede classificadas como LPWAN e já mencionadas anteriormente estão: SigFox, LoraWan, Weightless W/N/P, Nb-Iot, LTE-M, IEEE 802.11.ah e IEEE 802.11.af. Adicionalmente em (RAZA; KULKARNI; SOORIYABANDARA, 2017) o DASH7 é classificado uma tecnologia de rede LPWAN.

O DASH7 Alliance é um consórcio da indústria que define uma pilha de rede completa para conectividade LPWA conhecida como DASH7 Alliance Protocol (D7AP). Tendo origem no padrão ISO / IEC 18000-7 para a interface aérea para dispositivos de identificação por radiofrequência (RFID), o D7AP evoluiu para uma pilha que fornece conectividade de médio alcance para sensores e atuadores de baixa potência. Pode operar em 433 MHz, 868 MHz, 915 MHz e atingir até 5 Km com taxa máxima de transmissão de dados de 166 Kb/s (MOCNEJ, 2018).

Redes LR-WPAN são aderentes ao padrão IEEE 802.15.4, especialmente desenvolvido para dispositivos embarcados de baixa potência, curto alcance e baixa taxa de bits. Os grupos de trabalho IEEE 802.15 visam manter o HW com custos baixos e a compatibilidade do protocolo entre os sensores. O padrão descreve a camada física e também a subcamada de controle de acesso à mídia (RAZA; KULKARNI; SOORIYABANDARA, 2017).

Os padrões IEEE 802.15.4g e IEEE 802.15.4k são variantes do padrão IEEE 802.15.4 que visam estender o alcance do rádio, assim como diminuir o consumo de energia, ambos operam nas bandas ISM (*Industrial, Cientifico e Medico*). Tais bandas são internacionalmente reservadas para o desenvolvimento industrial, científico e médico, sendo reservada no Brasil a faixa de 2.4 GHz. O padrão IEEE 802.15.4g pode atingir até 1 Km com taxa máxima de transmissão de dados de 800 Kb/s, enquanto o padrão IEEE 802.15.4k pode atingir até 5 Km com taxas máxima de transmissão de dados de 128 Kb/s (CHANG; MASON, 2012).

O 6LowPAN é uma adaptação da camada de rede sobre IEEE 802.15.4 que é padronizada pelo IETF (IETF, 2021), objetiva levar conectividade IPv6 as redes de baixo consumo (MOCNEJ, 2018).

De acordo com os trabalhos apresentados, fisicamente quanto maior a frequência de operação, maior será a energia necessária para a transmissão, pois a atenuação de propagação aumenta com a frequência, logo frequências Sub-Ghz são usualmente padronizadas para as tecnologias das WSN, o que permite a diminuição do consumo.

A taxa de transmissão de dados é uma limitação em tais tecnologias Sub-Ghz, visto que são limitadas pela menor largura de banda que ocupam, estando usualmente abaixo de 1 Mb/s, o que na maioria dos casos não se caracteriza como um problema, já que não é objetivo das WSN atender a um alto trafego de dados.

Além do consumo da rede sem fio, o consumo de energia de um CSN dependerá também dos outros elementos de HW do sistema, sendo o processador um componente de grande consumo. No microcontrolador mencionado anteriormente (NXP, 2020a), rodando em modo de baixo consumo à 4 MHz o consumo típico é de 0,25 miliamperes, considerando-se a alimentação por duas baterias comerciais de 2000 mAh terá vida útil de 666 dias.

A estimativa apresentada acima apenas revela a magnitude de consumo destes dispositivos quando em modo de baixo consumo. O consumo real de um CSN será dado pela ponderação do tempo em que permanece em plena operação, aonde grande parte dos componentes estão ativos e consumindo, e pelo tempo que passa em modo de baixo consumo. Estratégias intermediárias de consumo também podem ser adotadas de forma a otimizar a vida útil da bateria.

No componente apresentado (NXP, 2020a) várias configurações de consumo são possíveis, geralmente a velocidade do clock é um fator preponderante para o consumo, além da opção de operação à 4 MHz que consome 0,25 mA, algumas outras são oferecidas. Rodando no clock máximo à 48 MHz o consumo é de aproximadamente 8 mA, enquanto no modo de parada profunda (stop) é de apenas 2 uA.

Visto as possibilidades de gestão de energia internas ao microcontrolador, que é o principal elemento de HW, pode-se estender tais estratégias para a gestão de energia de todos outros elementos do circuito. Manter o sistema nos modos de menor consumo possível e no máximo tempo permitido pela aplicação pode otimizar o

consumo de energia e estender a vida útil da bateria.

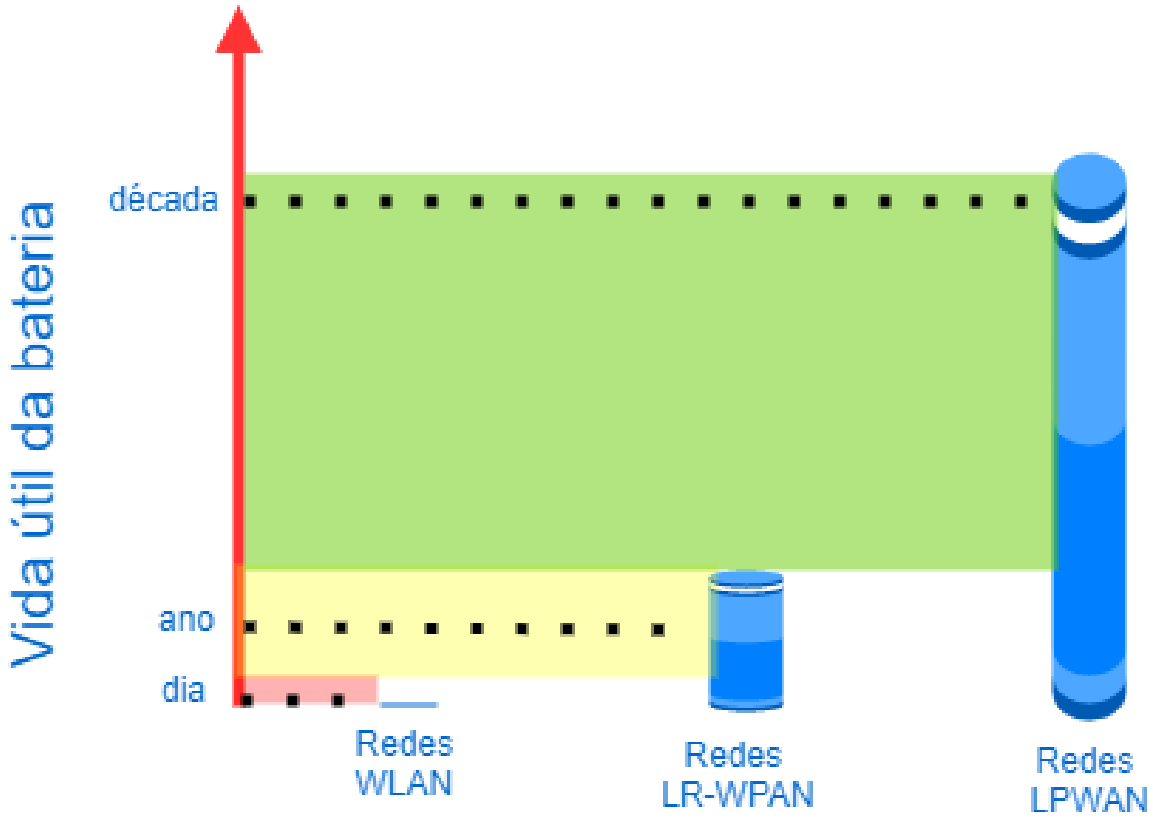
Em (MAJDOUL; SRIFI, 2017) é apresentada uma comparação de vida útil para uma bateria alimentando um CSN da tecnologia IEEE 802.15.4 (LR-WPAN) e também na tecnologia IEEE 802.11n (WLAN), de acordo com o trabalho, enquanto um CSN IEEE 802.15.4 pode ser alimentado por um período de 730 a 1095 dias, um CSN da IEEE 802.11n (Wi-Fi convencional) pode ser alimentado somente por 1 ou 2 dias. Já em (ADELANTADO et al., 2017) é estimado o tempo de vida de um CSN utilizando a tecnologia de rede Lora (LPWAN), quando alimentado por baterias é esperado que a vida útil delas possa se estender por até uma década.

Tais comparações levam em consideração apenas as características técnicas das redes, tais como potência consumida para transmissão e recepção, padrões mínimos de dados enviados e recebidos e intervalos de sincronização exigidos pela tecnologia da rede. Desta forma é necessário levar em consideração todos os outros elementos do CSN e as estratégias de gestão de energia para minimização do consumo de energia numa aplicação real de um CSN.

Por meio da literatura é possível então concluir que as tecnologias de redes que objetivam ao atendimento das WSN almejam estender a vida útil das baterias por até anos de autonomia, enquanto um CSN conectado a uma rede WLAN Wi-Fi doméstica, por exemplo, tem autonomia de apenas alguns dias quando alimentado por baterias. As tecnologias de rede LR-WPAN podem estender a vida útil por até alguns anos, enquanto as LPWAN são projetadas para atingir até uma década de autonomia.

Levando-se em consideração o consumo de bateria estimado por (MAJDOUL; SRIFI, 2017) e (ADELANTADO et al., 2017), é apresentado na figura 9 um gráfico comparativo da vida útil de bateria em um sistema do ponto de vista da tecnologia de rede utilizada.

Figura 9: Estimativa da vida de bateria de acordo com a tecnologia de rede utilizada.



Fonte: adaptado pelo autor, de acordo com (MAJDOUL; SRIFI, 2017) e (ADELANTADO et al., 2017).

3.4 Técnicas de tratamento de imagens para uso em WSN

De forma a atender os requisitos dos CSN e WSN, algumas técnicas são identificadas na literatura para habilitar a transmissão de imagens. Entre tais técnicas tem-se:

- a compressão da imagem, discutida na seção 3.4.1;
- a detecção de mudanças, discutida na seção 3.4.2;
- a confirmação de pacotes, discutida na seção 3.4.3;
- a de priorização de componentes da imagem, discutida na seção 3.4.4.

3.4.1 Compressão de imagens para uso em WSN

Na pesquisa de (PATEL; CHAUDHARY, 2017) são discutidos métodos energeticamente eficientes para a compressão de imagens, o qual impactam positivamente no tempo de vida de um CSN. Neste trabalho são discutidas as técnicas DCT, DWT e suas variantes.

O DCT ou Transformada Discreta do Cosseno é uma técnica muito importante para a comunidade de pesquisa que trabalha com compressão de imagens. DCT é uma técnica de transformação usada para transformar a representação espacial em componentes de frequências elementares. Nesta técnica a imagem é dividida em blocos, os quais são expressos como uma soma de funções cosseno. O método é derivado da Transformada de Fourier de forma que os coeficientes de baixa frequência têm maiores magnitude, enquanto coeficientes de frequência mais altas têm magnitude menor (PATEL; CHAUDHARY, 2017). O JPEG é um método de compressão de imagem bem conhecida e baseado em DCT.

Como variações do DCT são encontradas as seguintes técnicas:

- **LC-DCT- Low complexity DCT:** Os autores (KOUADRIA et al., 2013) propõe uma estratégia de compressão DCT eficiente em termos de energia, eles combinam o conceito Transformada Discreta Binária de Cosseno (BDCT) proposto por (BOUGUEZEL; AHMAD; SWAMY, 2013) adicionando-se uma estratégia de corte, de acordo com (BOUGUEZEL; AHMAD; SWAMY, 2013) a resolução de uma BDCT requer apenas 24 adições em um bloco de 8x8.

- **Fast Zonal DCT:** Esta técnica é apresentada por (LECUIRE; MAKKAOUI; MOUREAUX, 2012), e objetiva a conservação de energia na compressão de imagens em WSN. A técnica combina os conceitos de DCT baseado em Cordic (HEYNE et al., 2006) e Zonal DCT. O DCT baseado em Cordic é um algoritmo sem multiplicação usado no domínio 1D-DCT. No DCT baseado em Cordic, os blocos de coeficientes do DCT são obtidos aplicando-se o algoritmo em blocos de 8 linhas e 8 colunas. De forma a reduzir o número de operações e o consumo de energia (LECUIRE; MAKKAOUI; MOUREAUX, 2012) introduziram o Fast Zonal, aplicando-se o algoritmo sobre as 8 primeiras linhas e k colunas, obtendo assim menos coeficientes DCT.

Na DWT ou Transformada Discreta de Wavelet um sinal é decomposto em um conjunto de funções de base wavelet ortogonal. Essas funções são espacialmente localizadas. As funções wavelet são versões traduzidas e escalonadas das wavelets mães, essas funções estão espacialmente localizadas. Similarmente a análise de Fourier, o DWT é inversível, de modo que o sinal original pode ser reconstituído completamente a partir da sua representação DWT. Para a compressão de imagens o DWT 2-D é usado. O DWT 2-D pode ser implementado aplicando primeiro o DWT 1-D, um filtro passa-baixo a todas as linhas da imagem e, em seguida, aplicando o filtro passa-alto para todas as colunas (PATEL; CHAUDHARY, 2017).

Baseadas na DWT são encontradas as seguintes variantes:

- **Entropy based Wavelet Compression:** Nessa técnica, a Transformada Discreta de Wavelet é usada para compressão. Primeiramente a imagem é dividida em fragmentos de 16 pontos. Em seguida, a entropia de cada bloco é calculada atribuindo-se um valor de prioridade de entropia a cada bloco. Desta forma os blocos são ordenados e somente blocos de baixa entropia são comprimidos. Nesta abordagem alguns blocos de imagens são transmitidos com compressão, enquanto outros sem (MITTAL et al., 2016).

- **IC-DWT:** Para esta variação do DWT os autores (MOZAMMEL; CHOWDHURY; KHATUN, 2012) usam a Transformada Discreta de Wavelet com uma estratégia de corte. Nessa abordagem, o primeiro DWT é aplicado nas imagens, pelas quais quatro sub-bandas são geradas. Em seguida, esses coeficientes são comparados com o valor limite pré-inicializado. Coeficientes com valores abaixo do valor limite são definidos como zero e acima do valor limite são codificados usando técnicas de codificação por entropia de menor perda.

Técnicas hibridas utilizando-se o DCT e o DWT simultaneamente são analisadas por (PATEL; CHAUDHARY, 2017):

- **HIC using DWT and DCT:** A técnica apresentada por (BHARATH; PADMAJADEVI, 2013) utiliza imagens com resolução de 256 × 256, sendo aplicada a DWT em cada uma das sub-imagens de tamanho 32 × 32 em 2 níveis. Em cada nível, são descartadas as sub-bandas LH, HL e HH. desta forma 75% da imagem é compactada, então é aplicado o DCT nos blocos 8 × 8 resultantes.
- **HIC using DWT and Fast Zonal DCT:** O conceito de DWT com Fast Zonal DCT é usado para melhorar a compressão durante a transmissão em WSN (KISHK et al., 2014). O consumo de energia é reduzido dividindo o trabalho de compressão em diferentes clusters, particionando-se a imagens em vários blocos dependendo do número de clusters disponíveis. A técnica objetiva melhorar a compressão da imagem quando comparado ao JPEG2000.

De forma a se avaliar as performance das técnicas de compressão algumas métricas são definidas:

- **MSE - Mean Square Error:** Expressa o erro quadrático médio entre a informação contida em uma imagem após o método de descompressão Id e a imagem original Ir, onde m × n é o número de pixels (PATEL; CHAUDHARY, 2017):

$$MSE = 1/mn \sum_{0}^{m-1} \sum_{0}^{n-1} (I_d(m,n) - I_r(m,n))^2$$

- **PSNR Peak Signal to Noise Ratio:** É a relação entre o valor máximo possível da imagem e a potência de distorção que afeta a qualidade da representação desta, quanto mais alto o PSNR melhor será a qualidade da imagem decomposta (PATEL; CHAUDHARY, 2017), ele é definido por:

$$PSNR = 10log_{10}(255^2/MSE)$$

- **CR Compression Ratio:** É utilizado como métrica de eficiência de compressão, quanto maior a taxa de compressão significa uma melhor compressão (PATEL; CHAUDHARY, 2017). Na expressão Ac é o número de elementos da imagem comprimida e Ai o número de elementos da imagem original:

$$CR = Ac/Ai$$

Utilizando-se as métricas mencionadas anteriormente e as técnicas analisadas por (PATEL; CHAUDHARY, 2017), um comparativo é apresentado no quadro 3.

Quadro 3: Comparativo de técnicas de compressão.

	Técnica de compressão de imagem						
Parâmetro	**LC-DCT**	**Fast Zonal DCT**	**Entropy based Wavelet Compress**	**IC-DWT**	**Energy Efficient Distributed image compression**	**HIC using DWT and DCT**	**HIC using DWT and Fast Zonal DCT**
Técnica	DCT	DCT	DWT	DWT	DWT	Híbrida	Híbrida
Memória	Baixa	Baixa	Alta	Alta	Baixa	Alta	Baixa
CR	Baixo	Baixo	Alto	Alto	Baixo	Alto	Alto
Poder Process.	Baixo	Baixo	Alto	Alto	Baixo	Alto	Baixo
PSNR	Baixo	Baixo	Medio	Alto	Alto	Alto	Médio
Perform. Tempo	Baixo	Baixo	Alto	Alto	Baixo	Alto	Baixo
Energia Compact.	Alto	Alto	Baixo	Baixo	Biaxo	Médio	Médio
Complex. Comput.	Alta	Alta	Baixa	Baixa	Baixa	Média	Média

Fonte: elaborado pelo autor, de acordo com (PATEL; CHAUDHARY, 2017).

Um estudo comparativo entre técnicas de compactação de imagens analisados sobre protocolos de roteamento utilizando o simulador NS-2 são apresentados em (DEEPTHI; RAO; PRASAD, 2018). Nesta simulação foram comparados os métodos de compactação EZW e SPIHT, sob os algoritmos de roteamento AODV, DSDV, esses métodos são detalhados a seguir:

- **EZW:** Embedded zero tree wavelet, é uma transformada wavelet discreta e entropia codificada por meio de quantização aproximada sucessivas (DEEPTHI; RAO; PRASAD, 2018);
- **SPIHT:** Set partitioning of hierarchical tree, é um algoritmo de compressão eficiente, embarcado, simples, rápido e auto adaptativo (DEEPTHI; RAO; PRASAD, 2018);
- **AODV:** Adhoc on-demand distance vector, é um protocolo de roteamento que armazena os dados em formas de tabelas e com mecanismo de descoberta de rotas (DEEPTHI; RAO; PRASAD, 2018);
- **DSR:** Dynamic source routing, é um roteamento baseado na fonte, que significa uma vez o caminho é estabelecido, a fonte envia os dados, que é chamada de informação de roteamento da fonte e no pacote são incluídos os dados e o caminho de roteamento. Este protocolo possui descoberta e manutenção de rotas (DEEPTHI; RAO; PRASAD, 2018).

Neste experimento conduzido por (DEEPTHI; RAO; PRASAD, 2018), o número de nós simulados foi de 10, 20 e 30 tendo como padrão de rede o WiFi (IEEE 802.11). As imagens foram compridas aplicando-se os métodos EZW e SPIHT, tendo sido obtidos os seguintes resultados:

- **Técnicas de compressão:** Método SPIHT obtêm taxas de compressão maiores o que tem impacto positivo no tráfego da rede, enquanto o EZW obteve melhor qualidade de imagem utilizando-se métricas como o PSNR ou MSE (DEEPTHI; RAO; PRASAD, 2018);
- **Protocolos de roteamento:** Relativo aos protocolos de roteamento o AODV tem um desempenho melhor em uma rede com um número maior de nós, enquanto o DSR tem um desempenho melhor quando

o número de nós é menor (DEEPTHI; RAO; PRASAD, 2018). Outra conclusão é que quanto maior a taxa de compressão menor será o atraso de recebimento dos pacotes no nó receptor.

3.4.2 Detecção de mudanças

As técnicas de detecção de mudança objetivam avaliar se houveram mudança significativa do conteúdo em uma sequência de imagens. A sua utilização pode ser vantajosa em aplicações IoT sendo um critério de decisão para a execução de algoritmos de compressão ou transmissão.

No sistema de vigilância apresentado por (PHAM, 2016) é utilizado a tecnologia de longo alcance Lora e imagens com resolução 128x128 em tons de cinza. Neste sistema são utilizadas técnicas de detecção de mudanças baseadas na simples diferenciação de imagens, técnicas de compressão e transmissão de dados. O método de compressão é um codificador semelhante ao JPEG e opera em blocos de 8x8 pixels com otimizações avançadas na computação de dados para manter a sobrecarga computacional baixa. De acordo com o autor o sistema é tolerante a perda de pacotes durante a transmissão desta forma não é necessária a confirmação de recebimento de pacotes.

Dentre as técnicas de detecção de mudanças de imagens a de simples diferenciação é a mais básica encontrada de acordo com (MINU; SHETTY, 2015), considerando-se um valor de limiar T, ela é definida da seguinte forma:

$$D(x) = I_2(x) - I_1(x)$$

$$B(x) = \begin{cases} 1 & \text{quando } |\mathrm{D(x)}|{>}\mathrm{T} \\ 0 & \text{caso contrário} \end{cases}$$

Onde D(x) representa a diferença entre regiões de imagens subsequentes, e B(x) a função indicando se a região (x) foi modificada, sendo 1 se houve alteração ou 0 caso contrário.

3.4.3 Confirmação de pacotes

Além dos métodos de compressão clássicos e de detecção de mudanças outra técnica foi identificada na literatura, relacionada a mudanças no protocolo de aplicação para otimização da transmissão da imagem na WSN.

No trabalho de (CHEN; EAGER; MAKAROFF, 2019) é apresentado um sistema de monitoramento remoto de um campo de cultivo de canola por meio de um sistema de aquisição de imagens e transmissão via redes Lora. As imagens são coletadas no local de cultivo e enviadas a uma estação de análise distante alguns quilômetros. De acordo com o autor a transmissão de imagens sobre o Lora é um desafio técnico devido a uma taxa de transmissão insuficiente e comprimento do quadro da mensagem de apenas 255 bytes. No sistema apresentado, chamado Multi-Packet Lora (MPLR), as imagens capturadas são comprimidas no padrão JPEG. A estratégia para a transmissão é a fragmentação da imagem em blocos menores. Todavia a confirmação do recebimento dos pacotes pelo nó receptor é realizado para um grupo de mensagens e não para um pacote individualmente. De acordo com o autor a técnica pode acelerar a transmissão de 2 a 7 vezes comparativamente a um sistema com confirmação individual por pacotes.

Embora o trabalho acima também use compactação, nota-se que a mudança na política de confirmação de pacotes pode também trazer benefícios em redes com taxas de transmissão muito baixas.

3.4.4 Priorização de componentes da imagem

Em (FELEMBAN; NASEER; AMJAD, 2020) uma técnica baseada na priorização de componentes da imagem é apresentada. Nesta técnica a imagem é dividida em 2 camadas, na primeira camada os bits mais significativos de cada pixel são agrupados e na segunda os bits menos significativos. Neste trabalho um sistema real é utilizado para a validação utilizando-se uma rede ZigBee com roteamento de único e múltiplos saltos. Na etapa seguinte as camadas são fragmentadas em blocos de 72 bytes, devido a limitação de um pacote ZigBee, sendo então enviadas do nó sensor (que contém a câmera) ao nó receptor. Na transmissão a primeira camada é enviada ao nó receptor seguida da segunda.

Utilizando-se como métrica um valor de referência para o PSNR da imagem obtida no receptor, os autores (FELEMBAN; NASEER; AMJAD, 2020) demonstram ser atingido o valor de referência do PSNR em um tempo menor quando comparado a transmissão da imagem sem a priorização. Nesta técnica não é utilizado nenhum método de compressão.

4 Projeto da API

Dado o problema geral de modelar a transmissão de uma imagem em uma rede qualquer, pode-se abstrair os seguintes componentes:

Imagem (Image): O item de informação a ser transmitido pela rede;

Fragmentador (Slicer): Qualquer rede tem um tamanho máximo de dados que podem ser transmitidos de uma única vez. Se o tamanho do bloco de dados transmitido é maior que a capacidade da rede, a fragmentação em partes menores é necessária, sendo a funções deste componente realizar este trabalho;

Protocolo de Aplicação (AppProtocol): Para coordenar a transmissão dos dados entre quem vai enviar e quem recebe os dados;

Comunicação (Transfer): Uma entidade representando o meio em que os dados serão efetivamente transmitidos.

Dado que aqui se trata especificamente da transmissão de imagens em WSN, com uso de técnicas específicas de otimização, são necessários ainda os seguintes componentes:

Compactador (Compression): Elemento para representar o algoritmo de compactação da imagem, caso exista;

Dado Compactado (CompressedData): Se existe um compactador, é necessário considerar um componente que represente os dados compactados;

Detector de mudanças (ChangeDetector): Componente para implementar o algoritmo de detecção de mudanças, caso exista.

Adicionalmente, um componente de suporte é mapeado, o banco de imagens (ImageDb). Tal componente representa uma grupo de imagens prontas para uso, facilitando a experimentação dos algoritmos e do arranjo de simulação.

No quadro 4 são mapeados nas referências os componentes de fragmentação, compactação e detecção de mudanças, que habilitam a API a transmitir imagens em WSN. É mapeado também o componente de protocolo de aplicação, para contemplar o caso onde a otimização é feita na confirmação dos pacotes, como mostrado na

seção 3.4.3. Note que, excetuando-se os casos onde se usam protocolos de transporte com confirmação, como o Transfer Control Protocol (TCP), a confirmação (ou não) dos pacotes enviados na rede precisam ser tratados diretamente na aplicação.

Quadro 4: Métodos aplicáveis aos trabalhos de referência: [1] (KOUADRIA et al., 2013); [2] (LECUIRE; MAKKAOUI; MOUREAUX, 2012); [3] (MITTAL et al., 2016); [4] (MOZAMMEL; CHOWDHURY; KHATUN, 2012); [5] (BHARATH; PADMAJADEVI, 2013); [6] (KISHK et al., 2014); [7] (DEEPTHI; RAO; PRASAD, 2018); [8] (PHAM, 2016); [9] (CHEN; EAGER; MAKAROFF, 2019); [10] (FELEMBAN; NASEER; AMJAD, 2020).

	Componente da API			
Trabalho	**Componente de fragmentação de imagens (Slicer)**	**Componente de detecção de mudança (ChangeDetect)**	**Componente de compressão (Compression)**	**Componente de protocolode (aplicação) (AppProtocol)**
[1]	✓		✓	
[2]	✓		✓	
[3]	✓		✓	
[4]			✓	
[5]	✓		✓	
[6]	✓		✓	
[7]	✓		✓	
[8]	✓	✓	✓	✓
[9]	✓		✓	✓
[10]	✓			✓

Fonte: elaborado pelo autor.

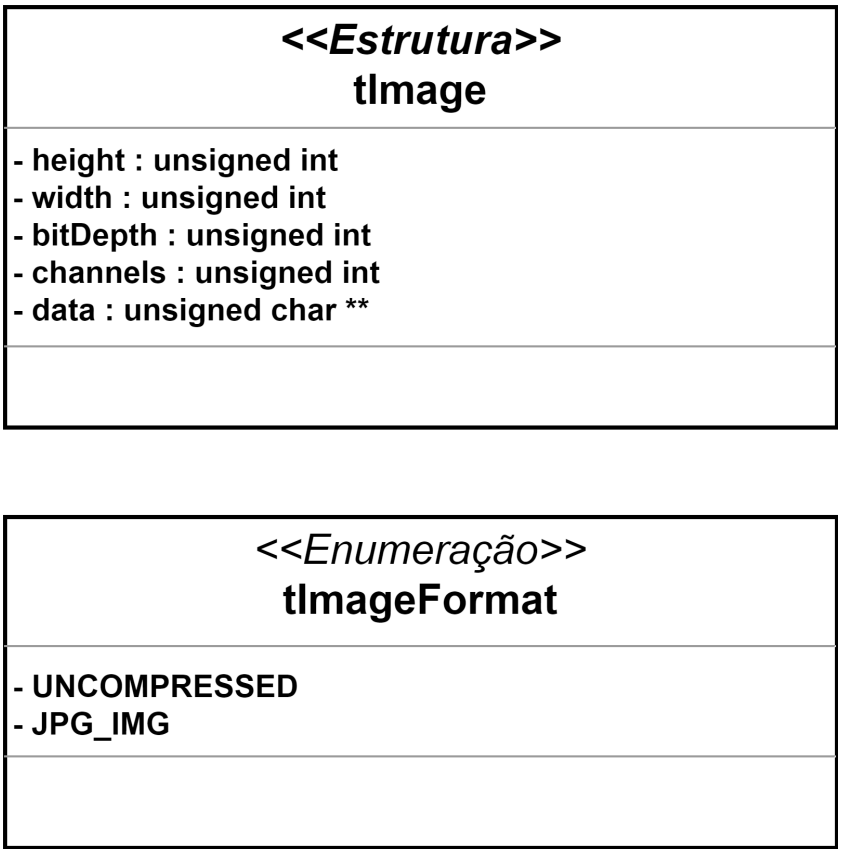
Na seção 4.1 são detalhados os componentes da API mencionados acima, enquanto na seção 4.2 são dados detalhes da implementação.

4.1 Componentes da API

4.1.1 Componente de Definição de Imagens

Um sensor de imagem é um elemento caracterizado por uma matriz de altura x largura de elementos, o qual é denominado resolução. Uma imagem é representada matematicamente por um vetor ou uma matriz de bytes. Cada elemento, também chamado de pixel, é caracterizado pelo quantidade de cores que pode assumir, expressado por meio do número de bits que o representa. O componente de definição de imagens encapsula as características de uma imagem, e sua representação de classe UML é dado na figura 10.

Figura 10: Representação de classe UML do componente de definição de imagens.



Fonte: elaborado pelo autor.

Atributos do componente tImage:

- height: Altura em pixels;
- width: Largura em pixels;
- bitDepth: Profundidade em bits das cores;
- channels: Número de matrizes de cores;
- data: Matriz de dados da Imagem não comprimidos;
- id: ID da imagem.

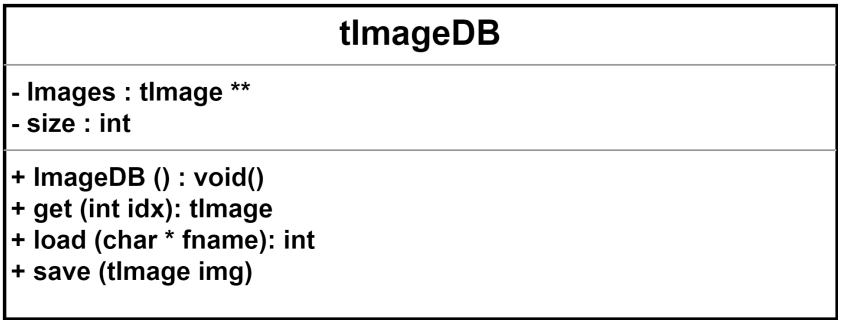
Enumeração tImageFormat:

- UNCOMPRESSED: Imagem não comprimida;
- JPG_IMG: Imagem comprimida.

4.1.2 Componente repositório de imagem

O repositório de imagens é um componente de suporte, representando um banco de dados de imagens para ser usado na experimentação. O projeto do componente do repositório de imagens é representado por meio da sua classe UML na figura 11.

Figura 11: Representação de classe UML do componente de captura de imagens e suas interfaces.



Fonte: elaborado pelo autor.

Atributos do componente ImageDb:

- images: Ponteiro para as imagens;
- size: Número de imagens.

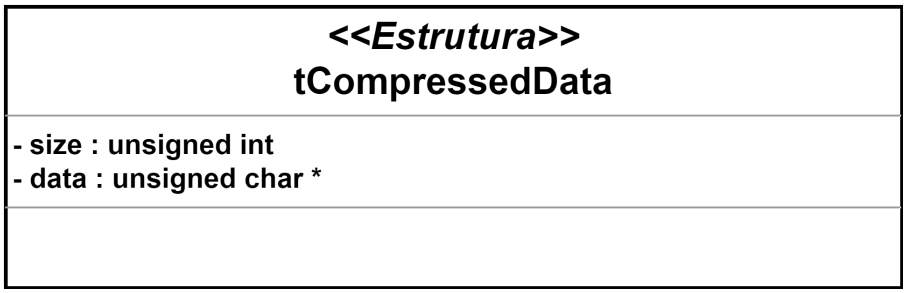
Métodos do componente ImageDb:

- ImageDb: Construtor do componente;
- load: Carrega imagem no banco de dados;
- get: Obtêm imagem do banco;
- save: Salva uma imagem no banco.

4.1.3 Componente de Dados Comprimidos

O componente de dados comprimidos define a estrutura de dados para o manuseio de dados comprimidos, e sua representação de classe UML é definida na figura 12.

Figura 12: Representação de classe UML do componente de compressão de dados.



Fonte: elaborado pelo autor.

Atributos do tCompressedData:

- size: Tamanho dos dados em bytes;
- data: Arquivos de dados.

4.1.4 Componente de detecção de mudança

As técnicas de detecção de mudança objetivam a redução da quantidade dedados a serem trafegados. Em uma sequência de imagens, as regiões desta que permanecem inalteradas não necessitam ser transmitidas, diminuindo o tráfego de dados e poupando energia da rede.

A interface do componente de detecção de mudança de imagem é dado na figura 13.

Figura 13: Representação de classe UML do componente de detecção de mudança de imagens.

@default@default

Fonte: elaborado pelo autor.

Métodos do componente:

- Init: Inicia componente;
- changeDetect: Detecta mudanças na imagem;
- setImage: Reconstitui uma imagem completa recebida;
- setChange: Reconstitui uma imagem com as porções recebidas.

Enumeração tChangedStatus:

- NO_CHANGE: Nenhuma alteração em relação a imagem anterior;
- FULL_CHANGE: Imagem com múltiplas alterações necessita ser enviada completamente;
- PARTIAL_CHANGE: Imagem com poucas alterações, somente mudanças serão enviadas.

4.1.5 Componente de Compressão

Por meio das técnicas de compressão de imagens é possível reduzir o tamanho do conjunto de dados que representam uma imagem, o qual impactam positivamente na diminuição do tráfego da rede e no consumo de energia para a transmissão desta. Assim sendo a previsão de um componente de compressão é mandatório no fluxo de elementos da API. A interface do componente de detecção de mudança de imagem é dado na figura 14.

Figura 14: Representação de classe UML do componente de compressão de imagens.

Fonte: elaborado pelo autor.

Métodos do componente Compression:

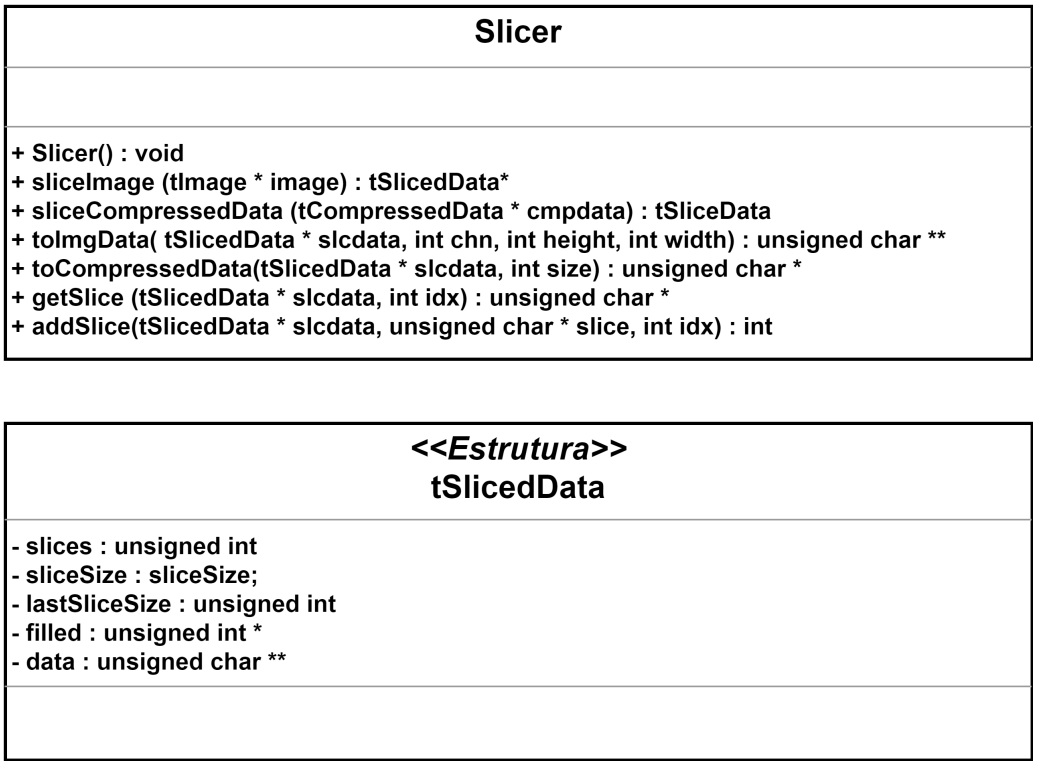
- Compression: Construtor do componente;
- compressImage: Comprime imagem;
- uncompressImage: Descomprime imagem.

4.1.6 Componente de fragmentação

Dado as características técnicas das redes, com baixas taxas de transmissão e pacotes de bytes limitados a centenas de bytes, e a limitação computacional dos CSN, um componente de fragmentação tem função chave na cadeia de elementos no contexto de manuseio de imagens. Por meio da fragmentação é possível dividir um problema de grande complexidade (bloco extenso de dados) em vários problemas menores, o qual pode ser solucionado pelo dispositivo.

A interface do componente de fragmentação é representada através de sua classe UML na figura 15.

Figura 15: Representação de classe UML do componente de fragmentação de imagens.



Fonte: elaborado pelo autor.

Métodos do componente Slicer:

- Slicer: Construtor do componente;
- sliceImage: Converte imagem não compactada em slices;
- sliceCompressedData: Converte imagem compactada em slices;
- isFilled: Indica se todos os slices que foram recebidos no server;
- toImgData: Converte um slice na matriz de dados da imagem não compactada;
- toCompressedData: Converte um slice em um fragmento de imagem compactada;
- getSlice: Obtêm um slice da imagem;
- addSlice: Adiciona um slice ao conjunto de blocos recebidos.

Atributos da estrutura tSlicedData:

- slices: Número de slices;
- sliceSize: Tamanho do slice;
- lastSliceSize: Tamanho do último slice;

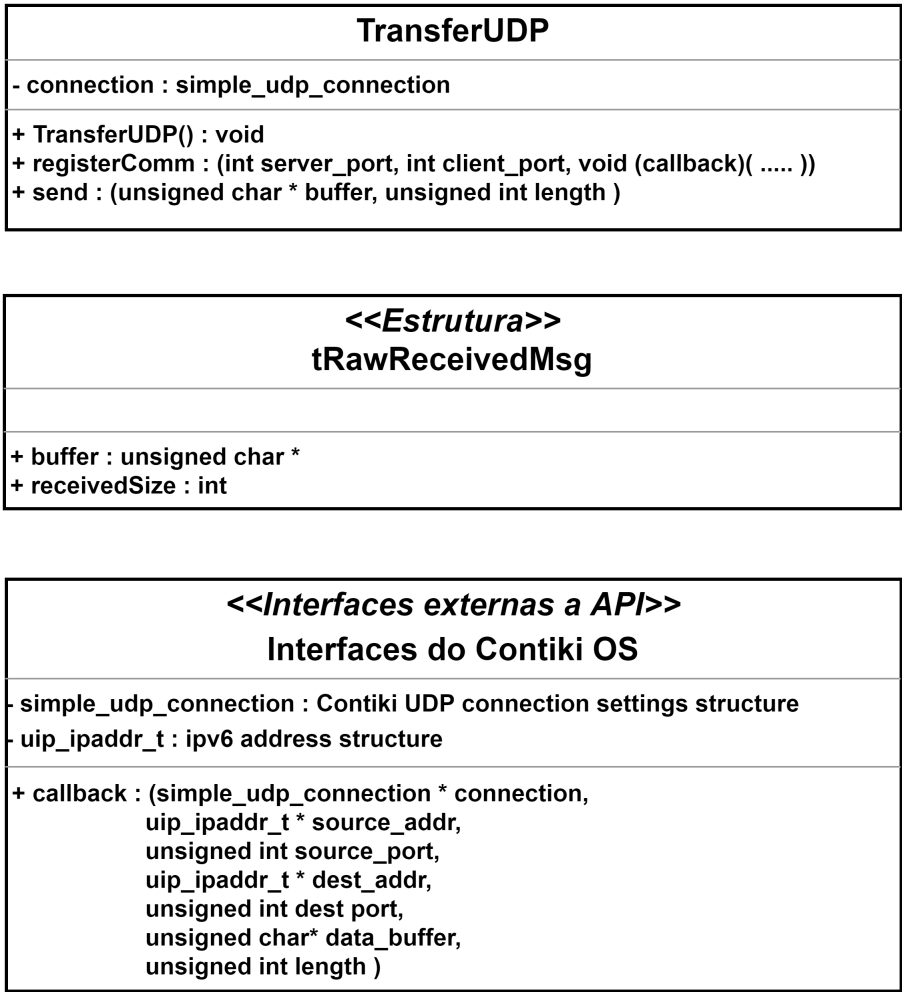
- filled: Indica quais slices já chegaram;
- data: Matriz de Slices.

4.1.7 Componente de Transferência UDP

Como já apresentado, o UDP é o protocolo mais indicado para o transporte em WSN, devido a sua baixa sobrecarga nos pacotes de dados, propiciando a otimização da camada de transporte de dados. O Contiki-NG suporta nativamente o UDP, por meio da camada de adaptação 6lowpan, o qual permitem que pacotes IPV6 trafeguem internamente.

A representação de classe UML do componente de Transferência UDP é dado na figura 16.

Figura 16: Representação de classe UML do componente de Transporte UDP.



Fonte: elaborado pelo autor.

Atributos do componente TransferUDP:

- simple_udp_connection: Estrutura de dados da conexão UDP.

Métodos do componente TransferUDP:

- TransferUDP: Construtor do componente;
- registerComm: Função de registro da comunicação;
- send: Função de envio de um bloco de dados.

Atributos da estrutura tRawReceivedMsg:

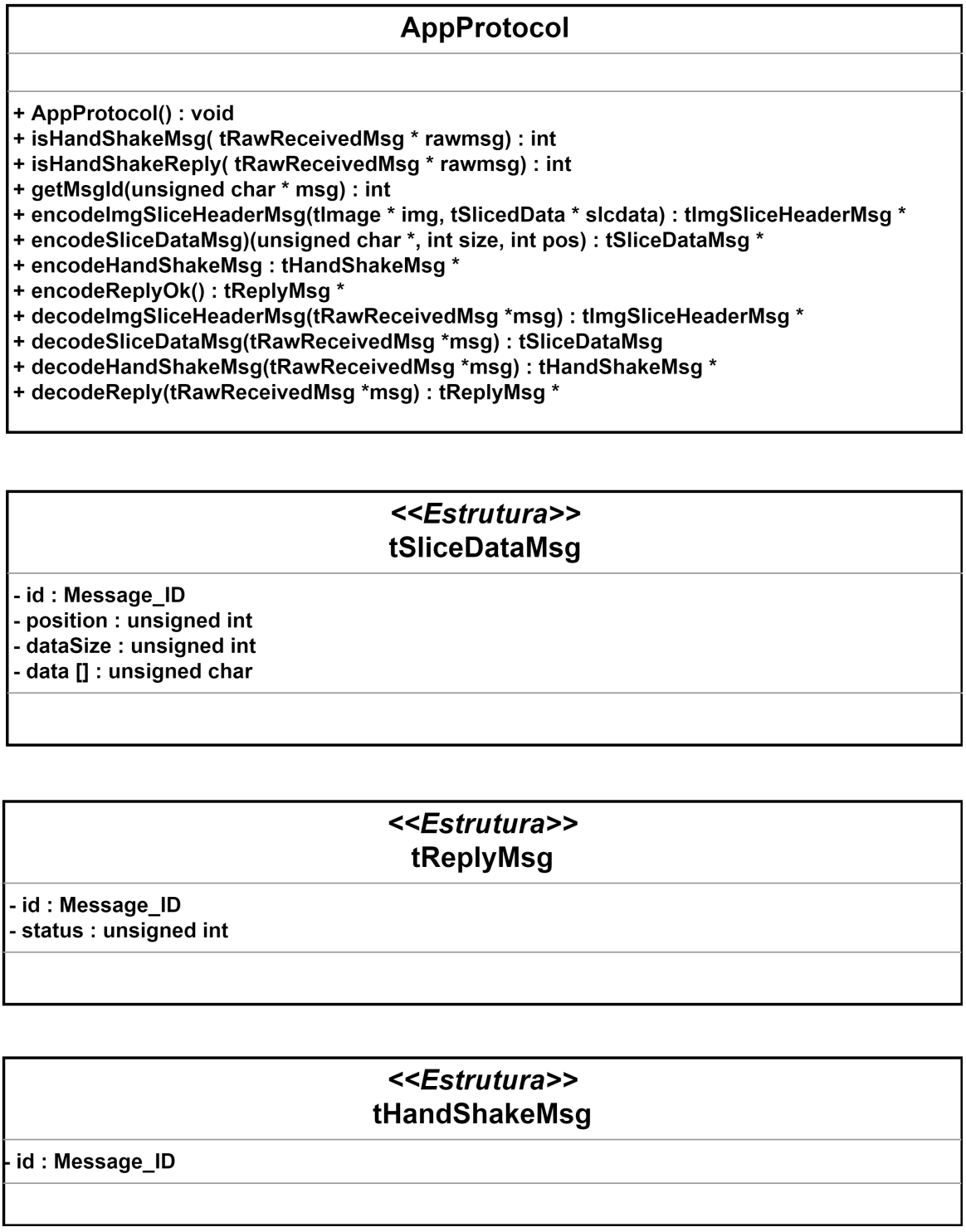
- buffer: Bufer de dados recebido;
- receivedSize: Número de bytes recebidos.

4.1.8 Componente de Protocolo de Aplicação

O componente de protocolo de aplicação tem o objetivo de prover suporte a comunicação entre os nós de rede cliente e servidor, permitindo assim que as operações blocos de dados gerados nas etapas anteriores possam ser transferidos do componente com função de cliente ao componente com função de servidor.

O Protocolo de Aplicação e descrito através da sua representação de classe UML dado nas figuras 17 e 18.

Figura 17: Representação de classe UML do componente de Protocolo de aplicação parte 1.



Fonte: elaborado pelo autor.

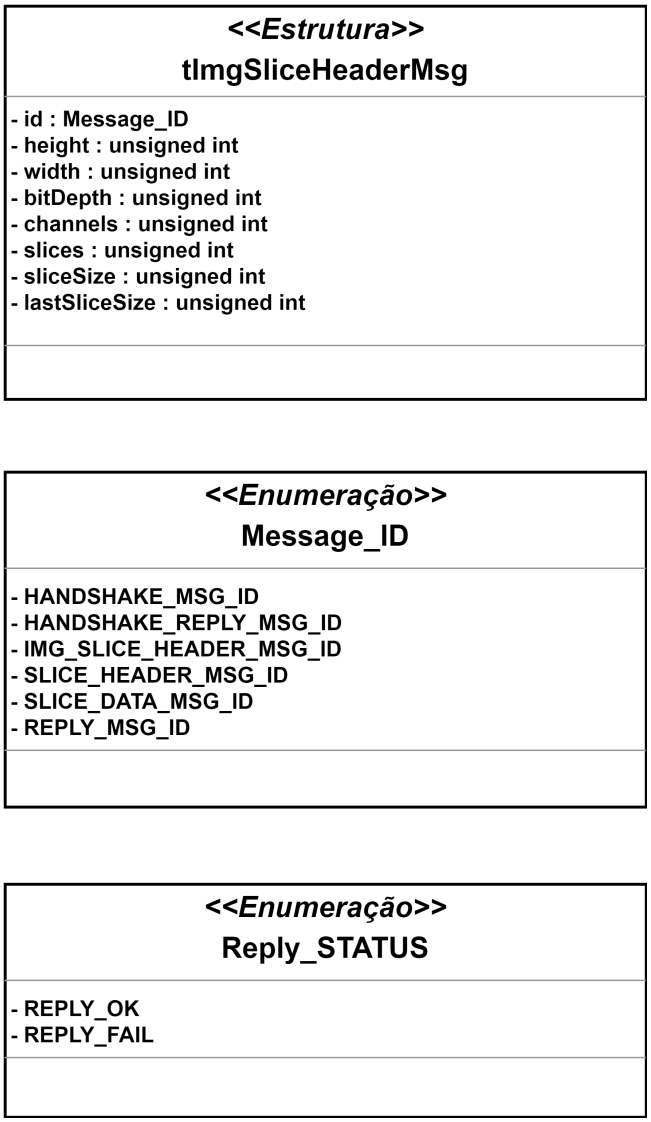
Métodos do componente AppProtocol:

- AppProtocol: Construtor do componente;
- isHandShakeMsg: Tratamento da mensagem HandShakeMsg;
- isHandShakeReply: Tratamento da mensagem HandShakeReply;
- isReplyOK: Tratamento da mensagem isReplyOK;
- getMsgId: Obtêm ID da mensagem;
- encodeImgSliceHeaderMsg: Codifica a mensagem slice header;
- encodeSliceDataMsg: Codifica a mensagem slice data;
- encodeHandShakeMsg: Codifica a mensagem de HandShake;
- encodeReplyOk: Codifica a mensagem Reply OK;
- decodeImgSliceHeaderMsg: Decodifica a mensagem slice header;
- decodeSliceDataMsg: Decodifica a mensagem slice data;
- decodeHandShakeMsg: Decodifica a mensagem de HandShake;
- decodeReply: Decodifica a mensagem de resposta.

Atributos da estrutura tImgSliceHeaderMsg:

- Message_ID id: ID da mensagem;
- height: Altura da imagem;
- width: Largura da imagem;
- bitDepth: Profundidade de cores;
- channels: Número de cores;
- slices: Número de slices;
- sliceSize: Número de bytes do fragmentos;
- lastSliceSize: Último fragmento recebido.

Figura 18: Representação de classe UML do componente de Protocolo de aplicação parte 2.



Fonte: elaborado pelo autor.

Atributos da estrutura tReplyMsg:

- Message_ID id: ID da mensagem;
- status: Estado da resposta.

Enumeração Message_ID:

- HANDSHAKE_MSG_ID: ID da mensagem de Handshake;
- HANDSHAKE_REPLY_MSG_ID: ID da mensagem de resposta de Handshake;
- IMG_SLICE_HEADER_MSG_ID: ID da mensagem de cabeçalho de um fragmento de imagem;

- SLICE_HEADER_MSG_ID: ID da mensagem de cabeçalho de um fragmento de dados;
- SLICE_DATA_MSG_ID: ID da mensagem de um fragmento de dados;
- REPLY_MSG_ID: ID da mensagem de resposta.

Enumeração Reply_STATUS:

- REPLY_OK: Sucesso na recepção;
- REPLY_FAIL: Falha na recepção.

4.2 Implementação da API

A implementação do protótipo da API foi feita em linguagem C, visto que os programas de validação foram compilados para o sistema operacional Contiki-NG. Idealmente, tal implementação poderia ser feita em alguma linguagem orientada a objetos como C++ ou Java, para utilização de características como herança e polimorfismo. Tais características facilitariam a adaptação e extensão da API para novas implementações.

Para permitir uma API com capacidades de adaptação e extensão mesmo utilizando linguagem C, cada componente definido no projeto da API na seção 4 é implementado como uma estrutura que contém os ponteiros para as funções implementando as funcionalidades da API.

Como exemplo, o componente que implementa o protocolo de aplicação é dado pela seguinte estrutura em C:

```
typedef struct tAppProtocol {
        int (*isHandShakeMsg)(tRawReceivedMsg *);
        int (*isHandShakeReply)(tRawReceivedMsg *);
        int (*isReplyOK)(tRawReceivedMsg *);
        int (*getMsgId)(unsigned char *);
        tImgSliceHeaderMsg * (*encodeImgSliceHeaderMsg)(tImage *, tSlicedData *);
        tSliceDataMsg * (*encodeSliceDataMsg)(unsigned char *, int, int);
        tHandShakeMsg * (*encodeHandShakeMsg)();
        tReplyMsg * (*encodeReplyOk)();
        tImgSliceHeaderMsg * (*decodeImgSliceHeaderMsg)(tRawReceivedMsg *msg);
        tSliceDataMsg * (*decodeSliceDataMsg)(tRawReceivedMsg *msg);
        tHandShakeMsg * (*decodeHandShakeMsg)(tRawReceivedMsg *msg);
        tReplyMsg * (*decodeReply)(tRawReceivedMsg *msg);
} tAppProtocol;
```

Tomando a função *encodeSliceDataMsg()*, que empacota um pacote de dados para ser enviado pela rede, poder-se ia tomar uma implementação de referência como:

```
tSliceDataMsg * tAppProtocol_encodeSliceDataMsg
  (unsigned char *data, int size, int pos){
        tSliceDataMsg * msg = (tSliceDataMsg*) malloc (sizeof(tSliceDataMsg));

        msg->id = SLICE_DATA_MSG_ID;
        msg->position = pos;
        msg->dataSize = size;
        memcpy(msg->data, data,size);
        return msg;
}
```

Tal implementação então pode ser inicializada no componente de protocolo de aplicação como:

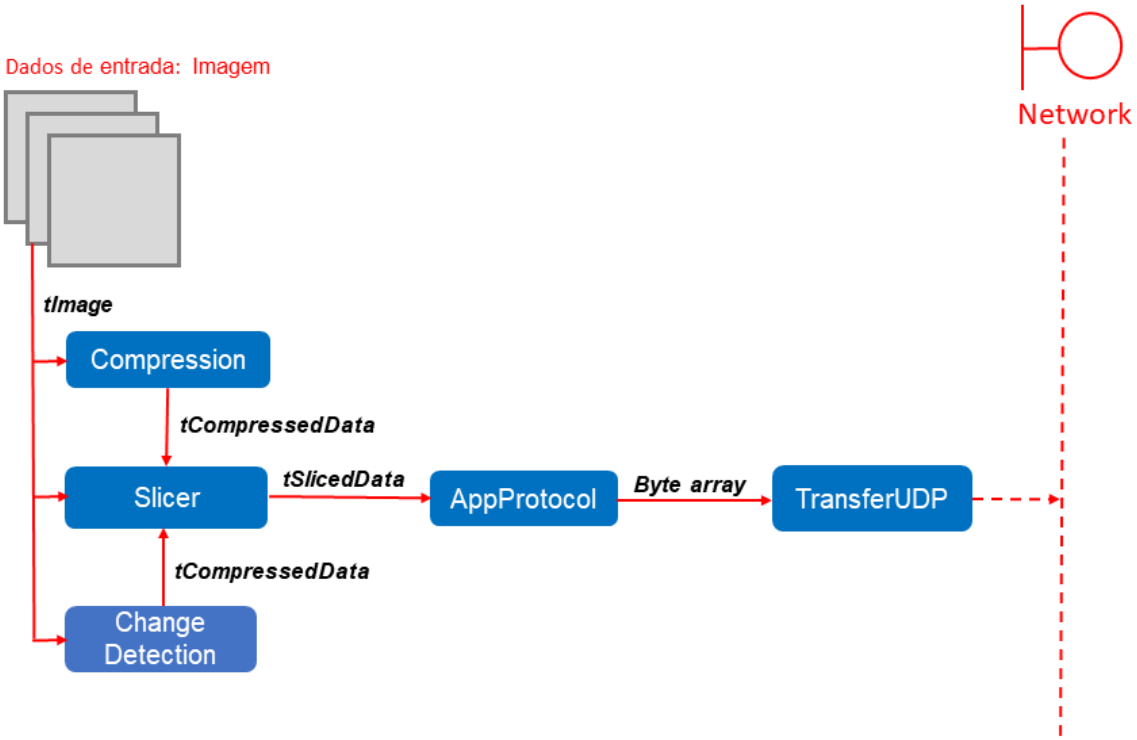
```
tAppProtocol * tAppProtocol_create () {
        tAppProtocol * protocol = (tAppProtocol*) malloc (sizeof(tAppProtocol));

...
        protocol->encodeSliceDataMsg = &tAppProtocol_encodeSliceDataMsg;
...

        return protocol;
}
```

Dada tal organização da API, pode-se modificar seu funcionamento de maneira simples, somente implementando uma função com comportamento diferente e inicializando o componente com o ponteiro para a nova função. Deste modo se considera que tal projeto da API permite que esta seja facilmente adaptável.

Se uma nova funcionalidade é necessário no componente, como por exemplo, codificar uma mensagem diferente, que pode ser necessária para uma aplicação diferente, basta adicionar o ponteiro para tal função no componente, implementar tal função e fazer a inicialização do ponteiro. Deste modo, considera-se também que a API é facilmente extensível.

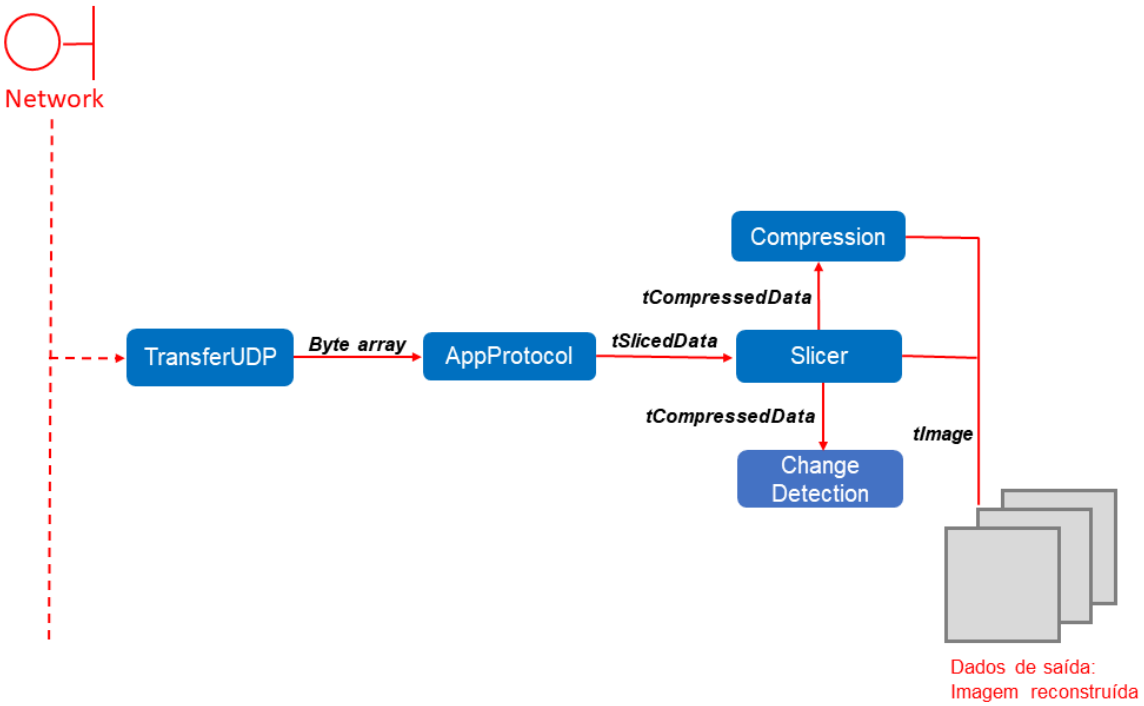
As possíveis formas de utilização dos componentes da API e as interfaces utilizadas nos lados cliente e servidor são dadas nas figuras 19 e 20.

Figura 19: Diagrama de utilização dos componentes da API e suas interfaces do lado cliente.



Fonte: elaborado pelo autor.

Figura 20: Diagrama de utilização dos componentes da API e suas interfaces do lado servidor.



Fonte: elaborado pelo autor.

5 Prototipação e Simulação

De forma a se verificar a viabilidade de utilização da API e a integração num ambiente de simulação, foi implementado um protótipo da API o qual possibilita a transmissão e a recepção de imagens. Neste protótipo os componentes ImageDb, Slicer e AppProtocol e TransferUDP estão plenamente funcionais, enquanto o ChangeDetection e o Compression implementam apenas a conversão de tipos para a simulação.

Para a comprovação da flexibilidade no uso dos componentes da API foram simulados alguns possíveis casos de uso em ambiente Cooja. O Cooja é a ferramenta de simulação nativa do Contiki-NG, que é um sistema operacional de código aberto. Tal sistema operacional implementa nativamente vários padrões de tecnologias IoT, como o suporte à IEEE 802.15.4, RPL, 6LowPAN e UDP (CONTIKI-NG, 2021a).

O Sistema Operacional Contiki-NG é escrito em linguagem C padrão e as aplicações são compiladas juntamente com o sistema operacional, enquanto o Cooja é desenvolvido em Java é compilado através do OpenJDK (ORACLE, 2021), que é uma plataforma Java de código aberto.

Nos trabalhos de simulação foi utilizado a distribuição Linux Ubuntu Desktop 64 bits versão 18.04, e a versão do OpenJDK 11.0.11. A instalação do Conitiki é dada no Wiki do projeto (CONTIKI-NG, 2021b), e de acordo com a documentação é possível instalar em Sistema Operacionais Linux e MacOs.

O ambiente de simulação no Cooja é baseado em elementos denominados motes, que são os nós da rede, representando um CSN. Para criação de cada nó é necessário informar a plataforma de HW e uma aplicação, que deve ser escrita em linguagem C.

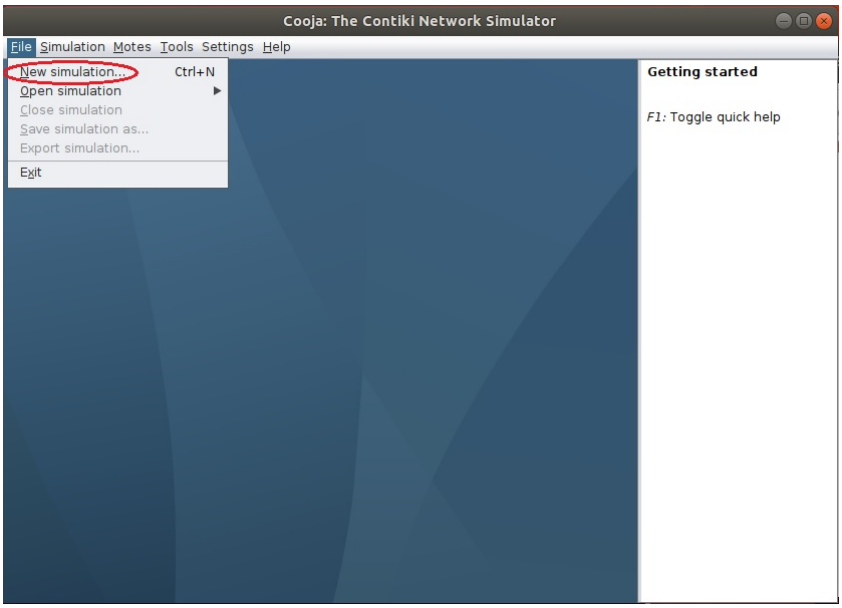
Dentro do projeto Contiki-NG existem algumas plataformas de HW nativamente suportados, sendo estas projetos de placas disponíveis comercialmente. Tais placas embarcam interface de rede sem fio IEEE 802.15.4 e alguns outros dispositivos como sensores de temperatura, botões e LEDs. Esses placas são denominadas *Mote Type* na interface da simulação do Cooja. A Z1 (ZOLERTIA, 2021) é exemplo de placa suportada pelo projeto Contiki-NG, e a lista completa de plataformas de HW pode ser encontrada na Wiki do projeto (CONTIKI-NG, 2021b).

Numa mesma simulação podem coexistir nós de diferentes plataformas de HW e aplicações, todavia é necessário que o compilador para a plataforma selecionada esteja propriamente configurado na cadeia de ferramentas de compilação do sistema operacional.

Resumidamente, as etapas para a configuração do ambiente de simulação é dada nos passos a seguir:

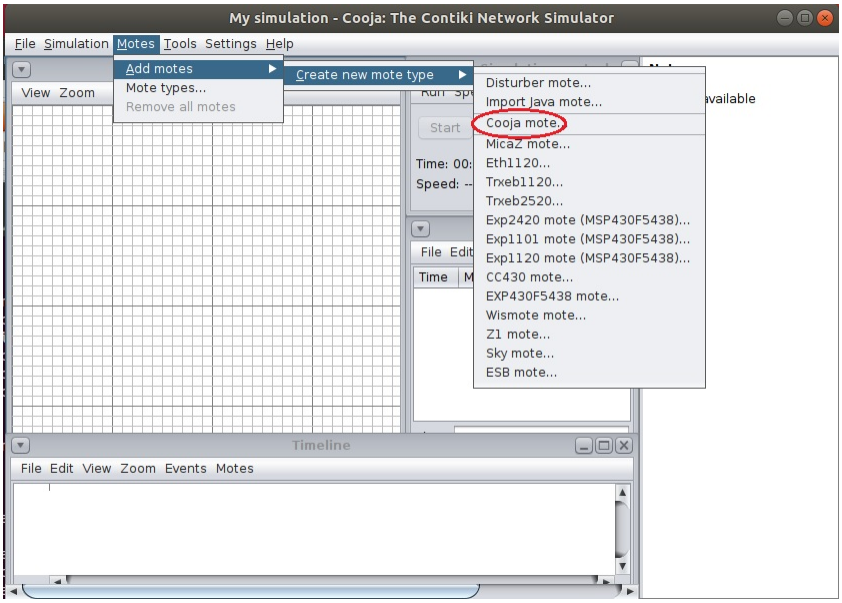
- Criação de uma simulação, mostrado na figura 21;
- Criação dos nós, mostrado na figura 22;
- Compilação da aplicação, mostrado na figura 23.

Figura 21: Criação de uma simulação no Cooja.



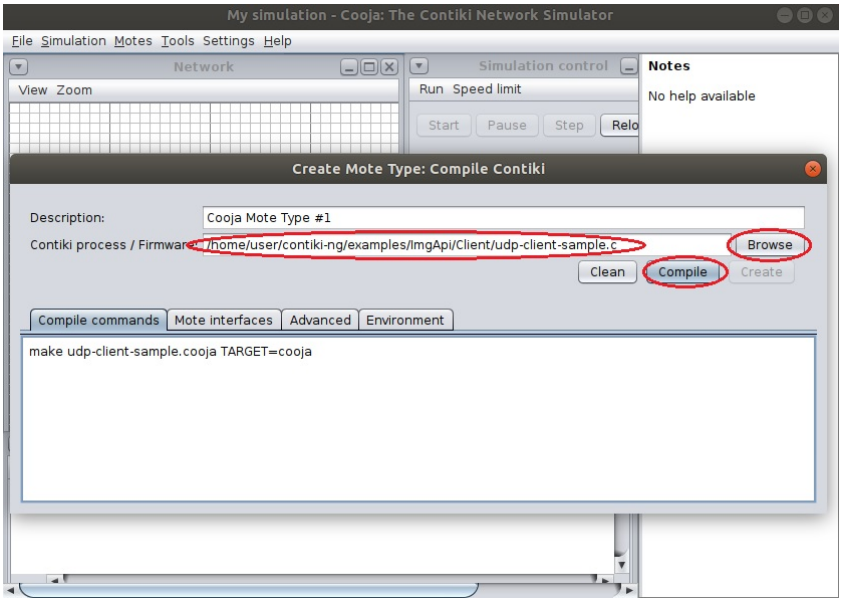
Fonte: elaborado pelo autor.

Figura 22: Criação de um mote Cooja.



Fonte: elaborado pelo autor.

Figura 23: Compilação de uma aplicação utilizando o Contiki-NG para simulação no Cooja.



Fonte: elaborado pelo autor.

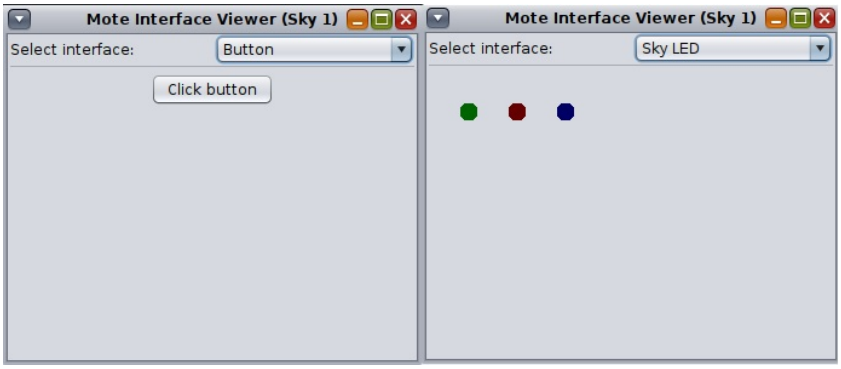
Após a etapa de compilação, o cenário de teste pode ser criado pela adição dos nós de teste.

No Cooja, para cada plataforma de HW são disponibilizados elementos de interface gráfica para a interação externa com o nó. Para acessar o menu de interfaces disponíveis deve-se posicionar o mouse sobre a representação

gráfica do nó, então seguir o caminho através dos menus: botão direito do mouse -> Mote Tools -> Mote Interface Viewer -> Select Interface.

Como exemplo, a figura 24 mostra a interface gráfica do nó da placa Sky, contendo o botão da placa e os leds. Tais elementos representam os respectivos componentes na placa física.

Figura 24: Interfaces do nó na simulação, exemplo da Sky.



Fonte: elaborado pelo autor.

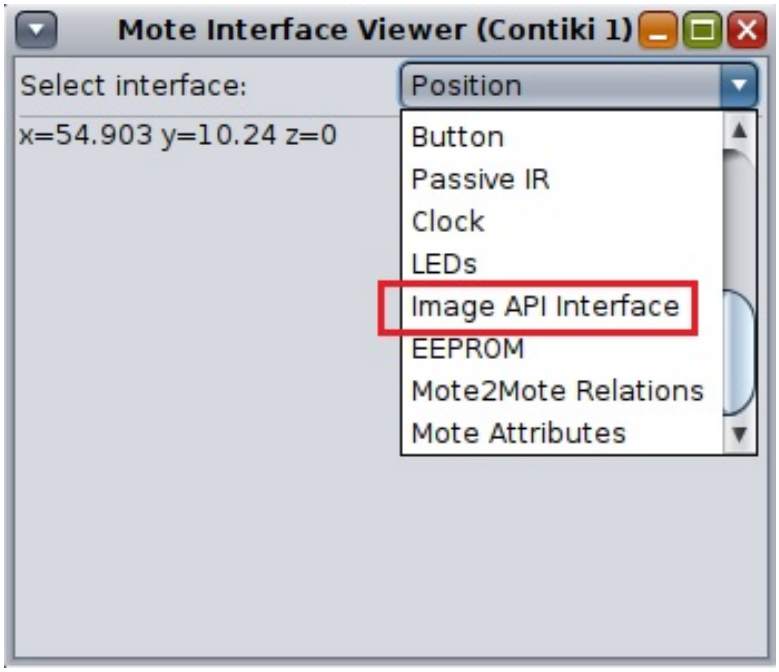
5.1 Simulações de Casos de Uso

Na simulação dos casos de uso foram utilizados um nó com a função de cliente e um outro com função de servidor, ambos com *Mote Type* Cooja. Este é um mote genérico e não esta associado a nenhuma projeto de HW. Utilizando-se tal *Mote Type* é possível construir uma aplicação tendo acesso a todos os serviços disponíveis no Contiki-NG e adicionalmente utilizar uma aplicação no ambiente de simulação do Cooja.

O compilador C GNU GCC deve estar propriamente configurado no sistema operacional para a utilização do *Mote Type* Cooja. Neste trabalho foi utilizado a versão do GCC 7.5.0 para o Linux 64 bits.

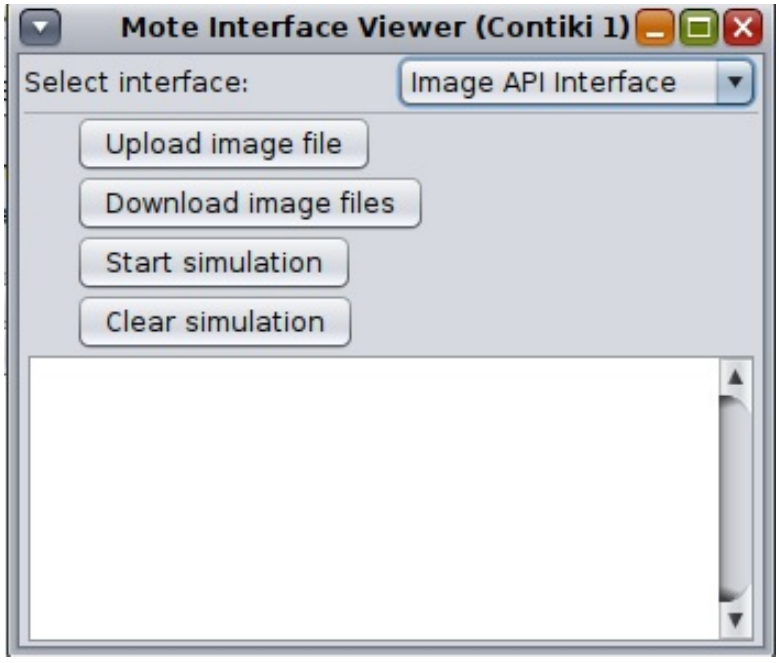
De forma a interagir com a simulação foi criada uma interface para a entrada e saída de dados da API, esta interface esta disponível para nós do tipo Cooja. O acesso a interface de entrada e saída de dados da API pode ser visualizada na figura 25, e o layout da interface pela figura 26

Figura 25: Acesso a interface de entrada e saída de dados da API no menu de interfaces.



Fonte: elaborado pelo autor.

Figura 26: Layout da interface de entrada e saída de dados da API.



Fonte: elaborado pelo autor.

Na interface de entrada e saída de dados da API estão contempladas as seguintes funcionalidades:

- **Botão Upload image file:** Adiciona um arquivo ao ambiente de simulação em um nó client;
- **Botão Download image file:** Salva os arquivos de simulação recebido pela simulação em um nó servidor;
- **Botão Start simulation:** Inicia uma simulação;
- **Botão clear simulation:** Limpa os arquivos de simulação;
- **Painel inferior:** Mensagens do sistema.

Visto que não é possível acessar o sistema de arquivo do Linux via aplicação Contiki-NG, a interface facilita a manipulação de arquivos de entrada e saída durante as simulações.

O Cooja implementa um mecanismo de comunicação entre a interface do Simulador (Java) e a aplicação Contiki-NG (linguagem C), por meio deste é possível ler e escrever variáveis definidas na aplicação. Desta maneira na aplicação foram previstos 10 vetores de entrada e 10 vetores de saída com tamanho até 500 Kb, o qual ficam acessíveis na simulação da aplicação do Contiki-NG, inclusive ao componente ImageDB.

Para a realização dos testes foram padronizados a ordem de carga das variáveis da simulação utilizando-se o botão Upload via interface, sendo a mesma ordem de leitura no ImageDB, desta forma a ordem de introdução das imagens na interface gráfica será o ordem de carga no componente da API.

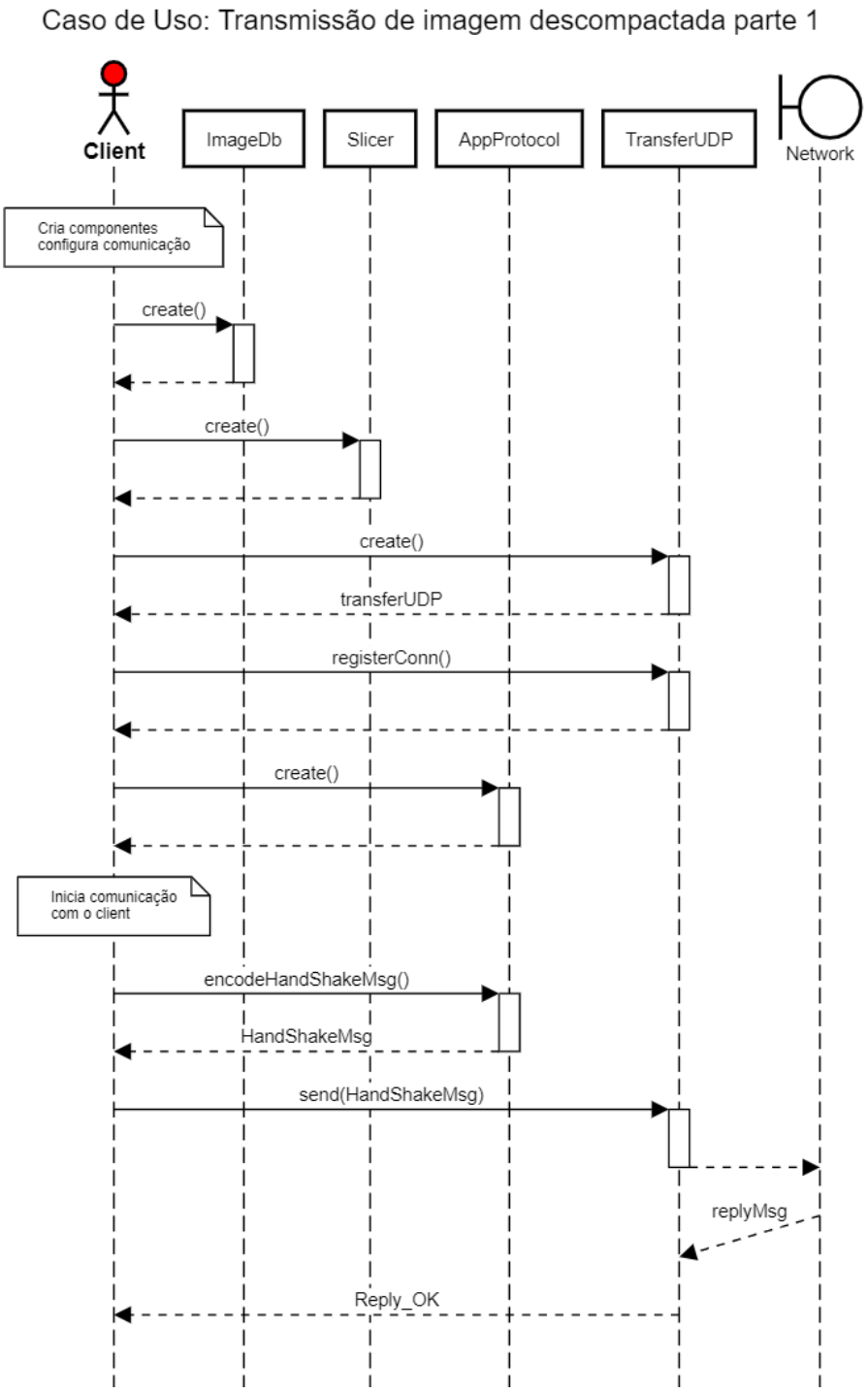
Quando da ação de Download todos os vetores de saída carregados serão gravados no sistema de arquivos Linux na forma binária nomeados da seguinte forma simul_output_0 até simul_output_n.

Outras soluções podem ser facilmente implementadas podendo ser necessário modificações na lógica da interface e na aplicação.

5.1.1 Caso de Uso: Imagem descompactada transmitida

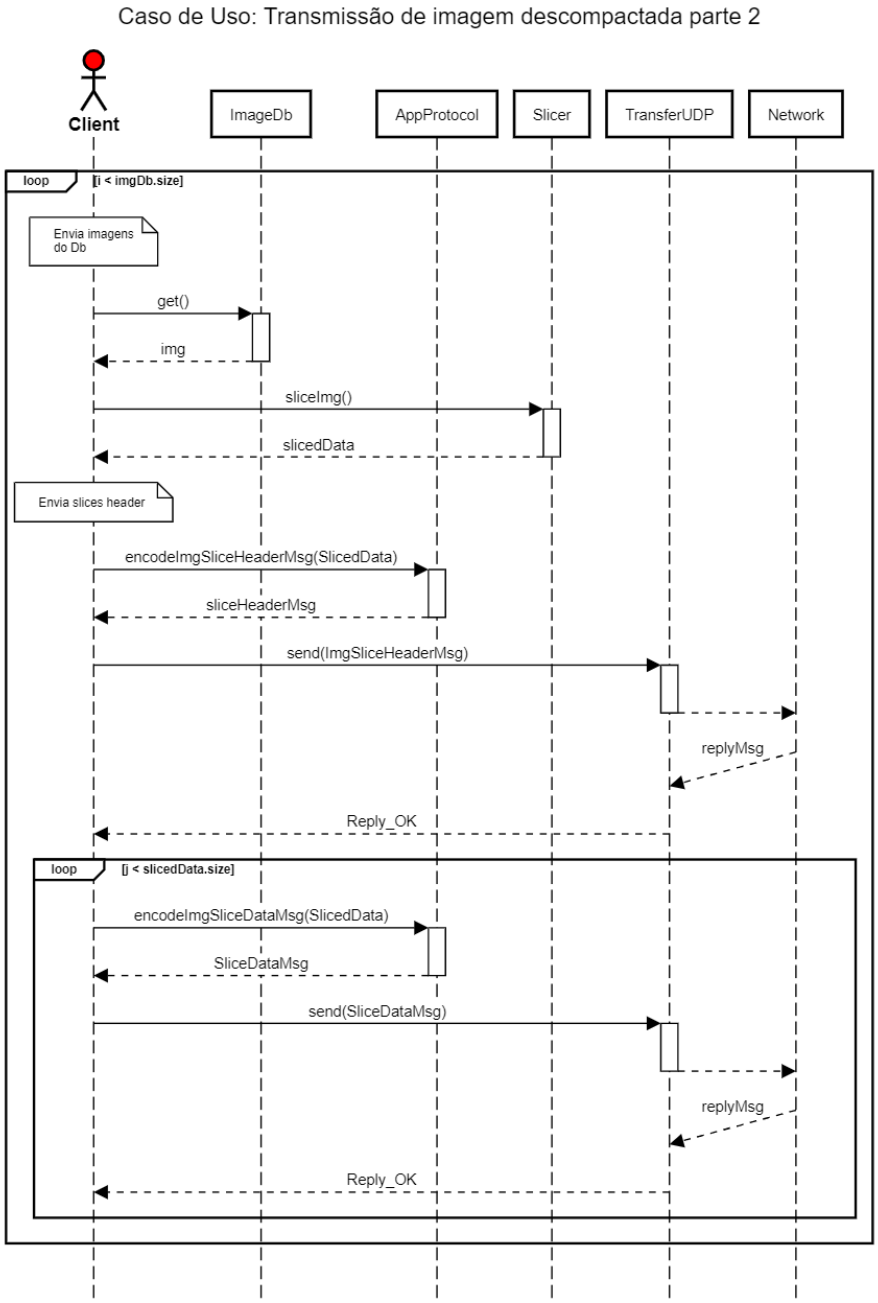
Neste caso de uso um conjunto de imagens descompactadas são enviada do cliente ao servidor, o diagrama de sequência da API nos lados cliente e servidor são dados respectivamente nas figuras 27, 28, 29 e 30.

Figura 27: Diagrama de sequência da API: Cliente transmitindo imagem descompactada (parte 1).



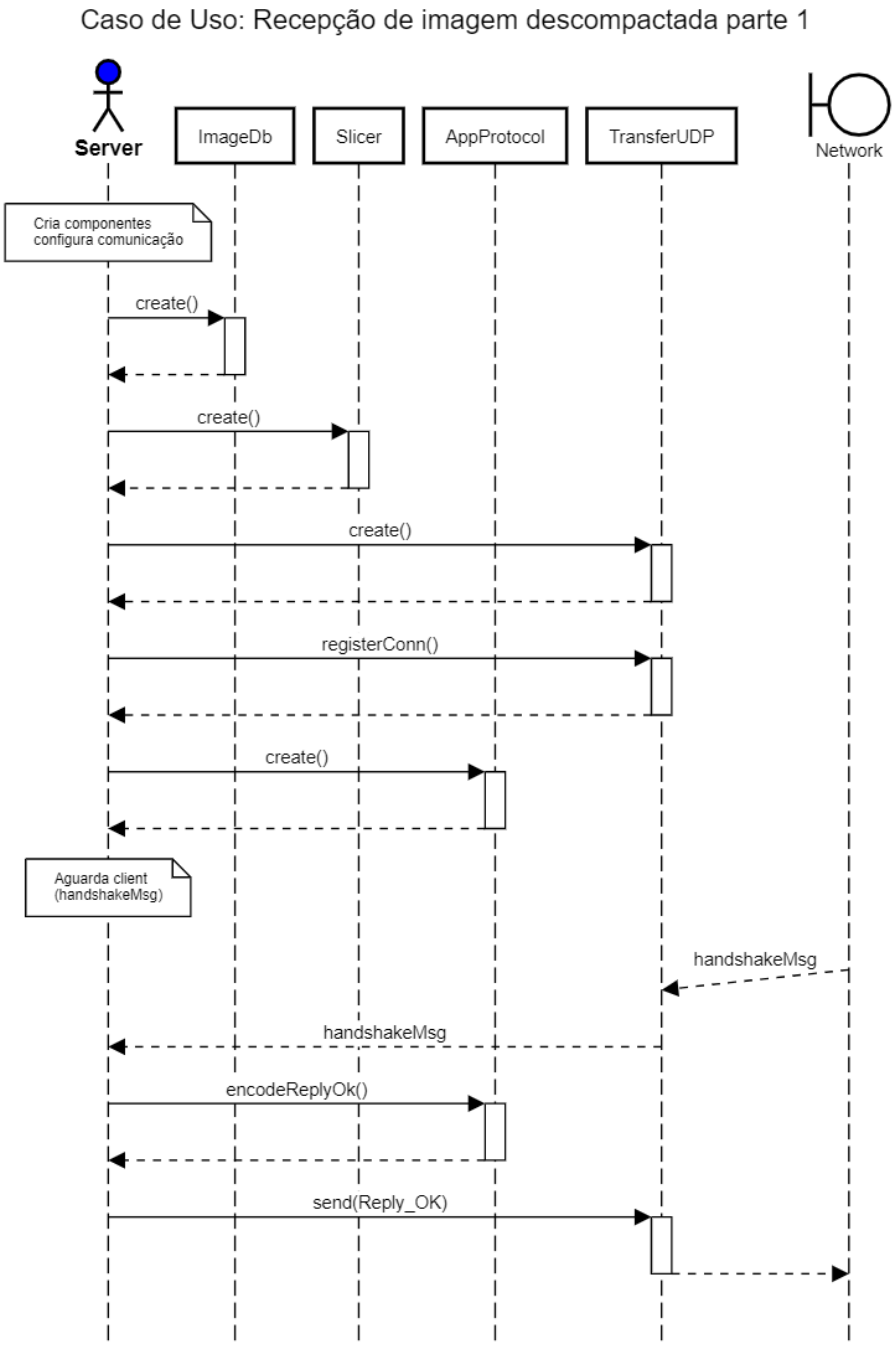
Fonte: elaborado pelo autor.

Figura 28: Diagrama de sequência da API: Cliente transmitindo imagem descompactada (parte 2).



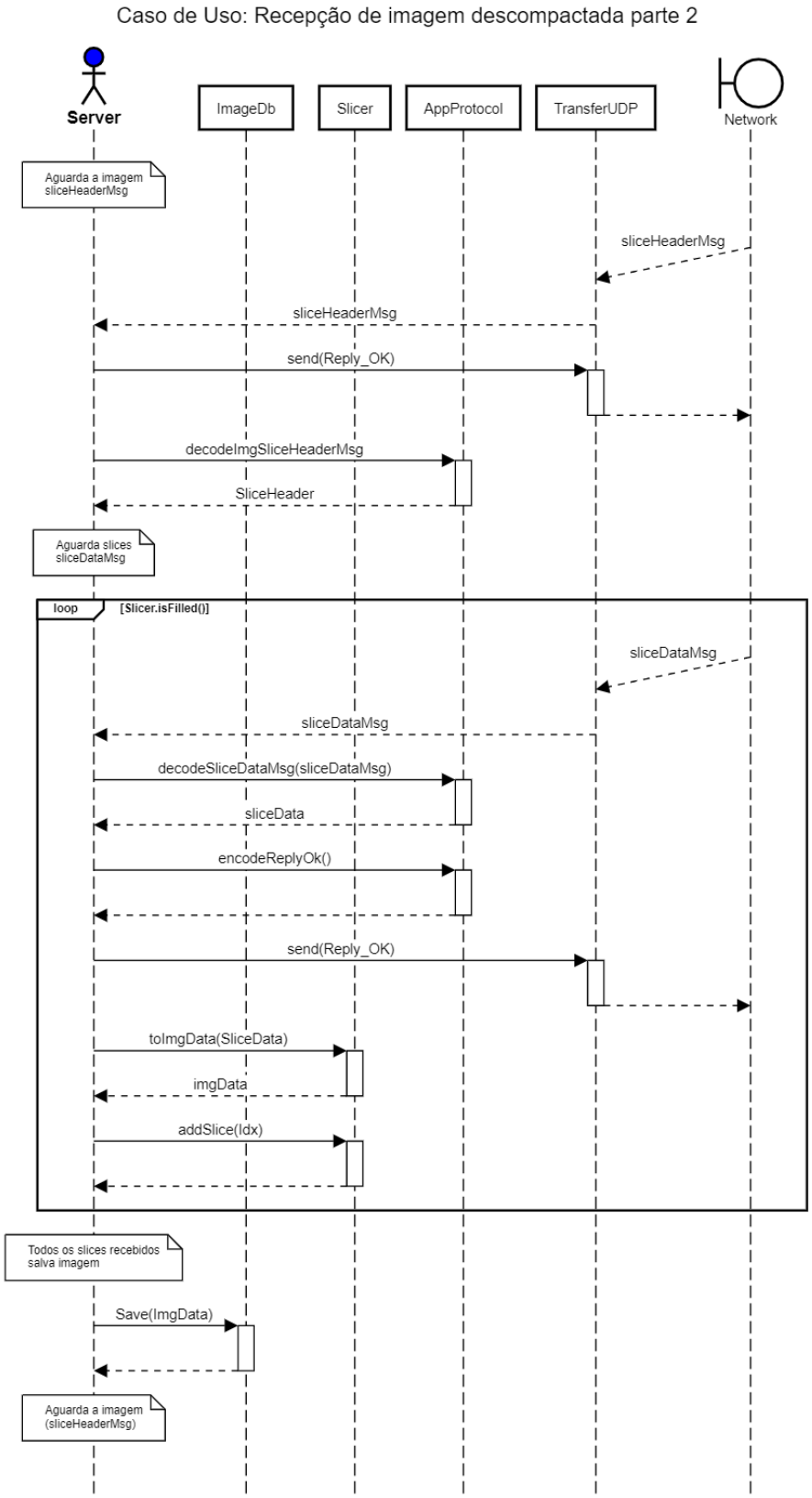
Fonte: elaborado pelo autor.

Figura 29: Diagrama de sequência da API: Servidor recebendo descompactada (parte 1).



Fonte: elaborado pelo autor.

Figura 30: Diagrama de sequência da API: Servidor recebendo imagens descompactada (parte 2).

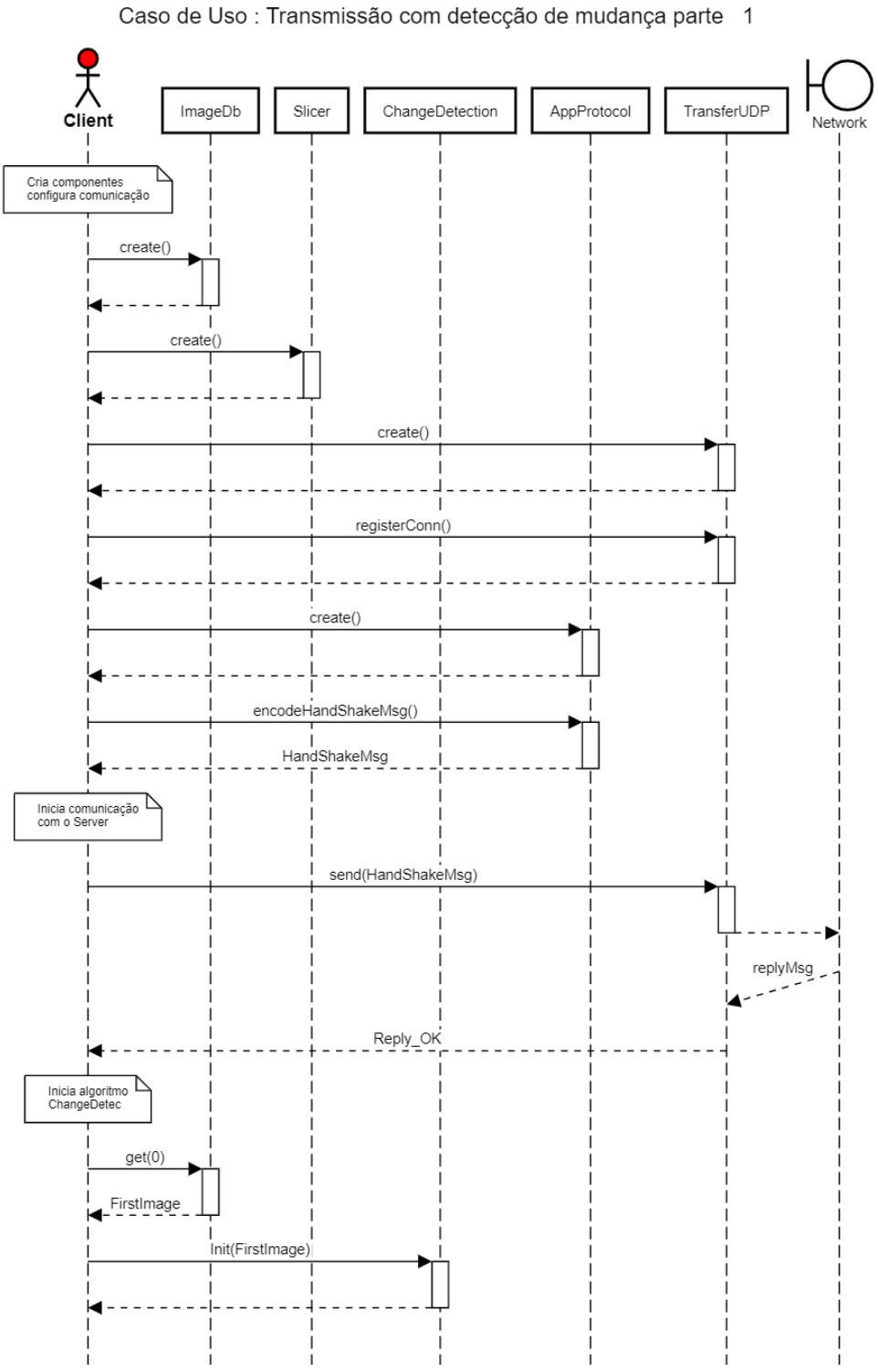


Fonte: elaborado pelo autor.

5.1.2 Caso de Uso: Detecção de mudança imagem

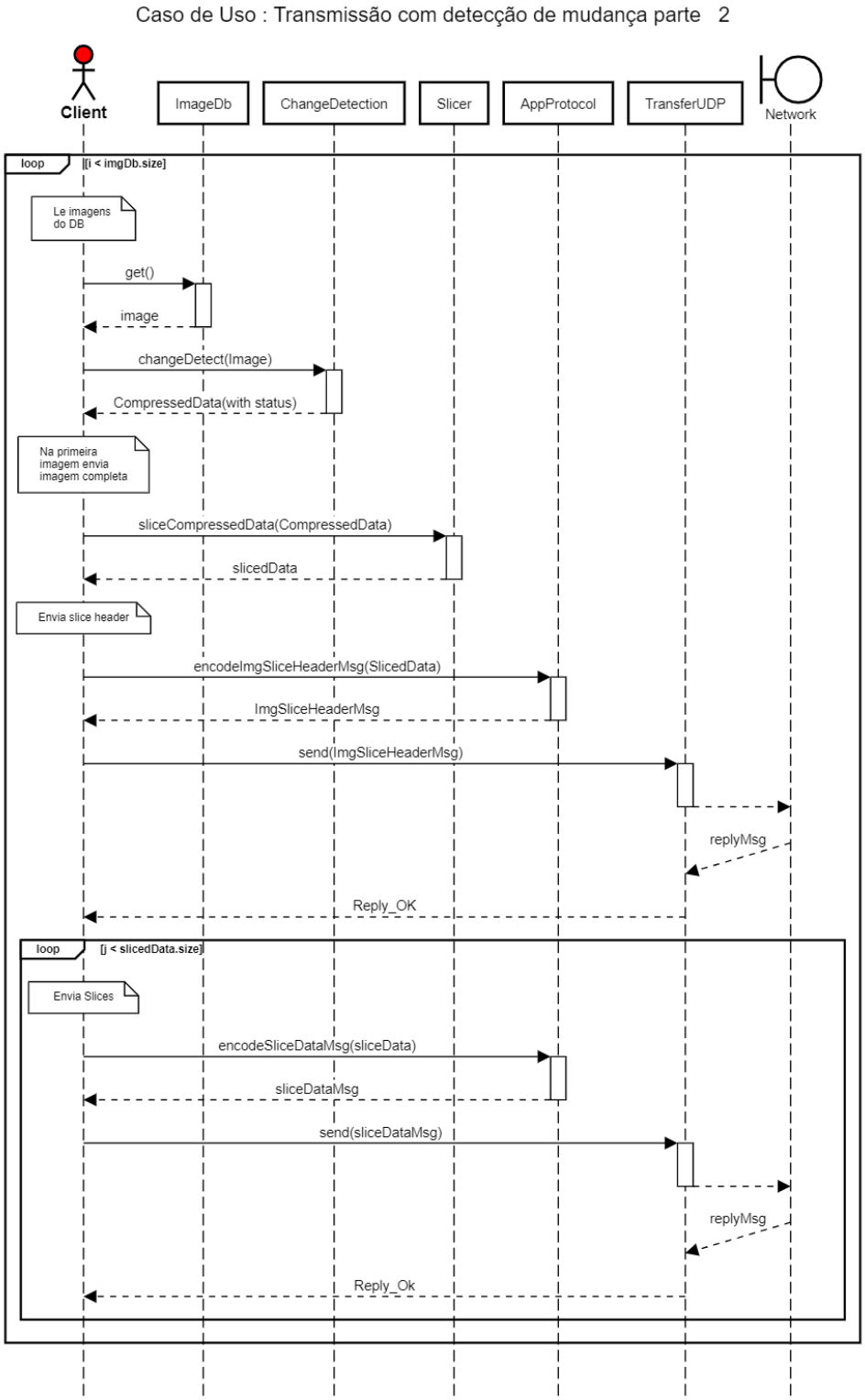
Neste cenário, o cliente analisa a sequência de imagens e apenas a porção alterada em relação a imagem anterior é transmitida ao servidor. O diagrama de sequência da API nos lados cliente e servidor são dados respectivamente nas figuras 31, 32, 33 e 34.

Figura 31: Diagrama de sequência da API: Cliente transmitindo detecção de mudança (parte 1).



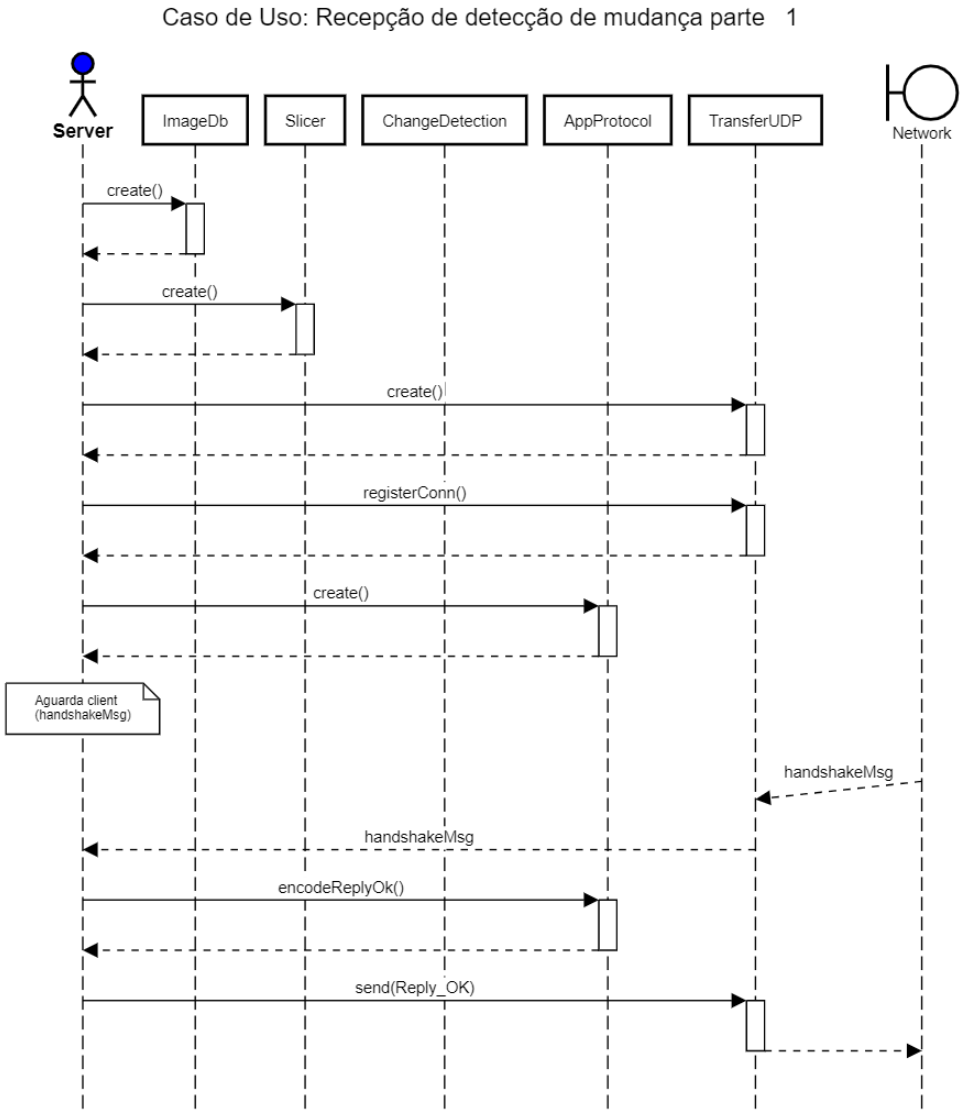
Fonte: elaborado pelo autor.

Figura 32: Diagrama de sequência da API: Cliente transmitindo detecção de mudança (parte 2).



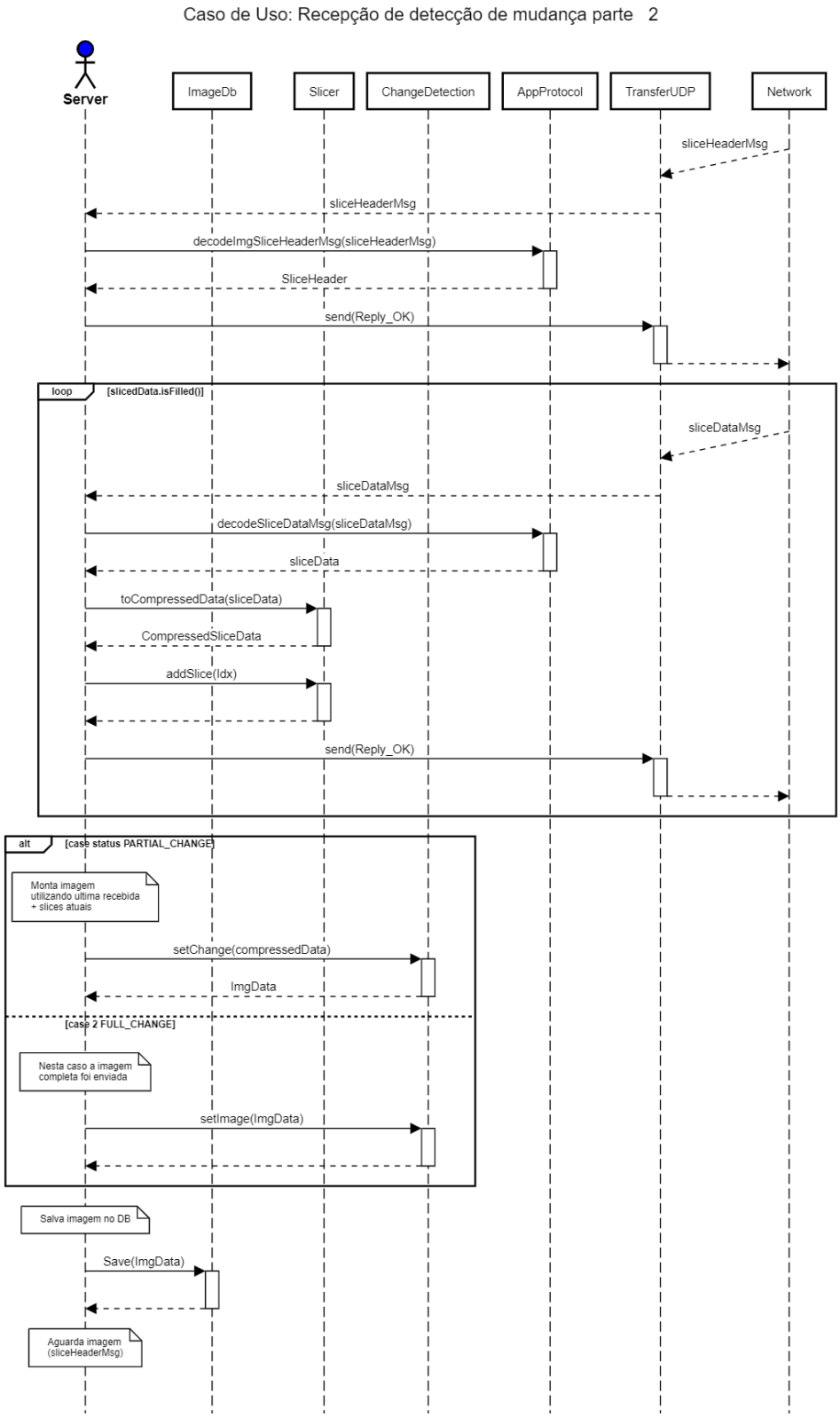
Fonte: elaborado pelo autor.

Figura 33: Diagrama de sequência da API: Servidor recebendo detecção de mudança (parte 1).



Fonte: elaborado pelo autor.

Figura 34: Diagrama de sequência da API: Servidor recebendo detecção de mudança (parte 1).

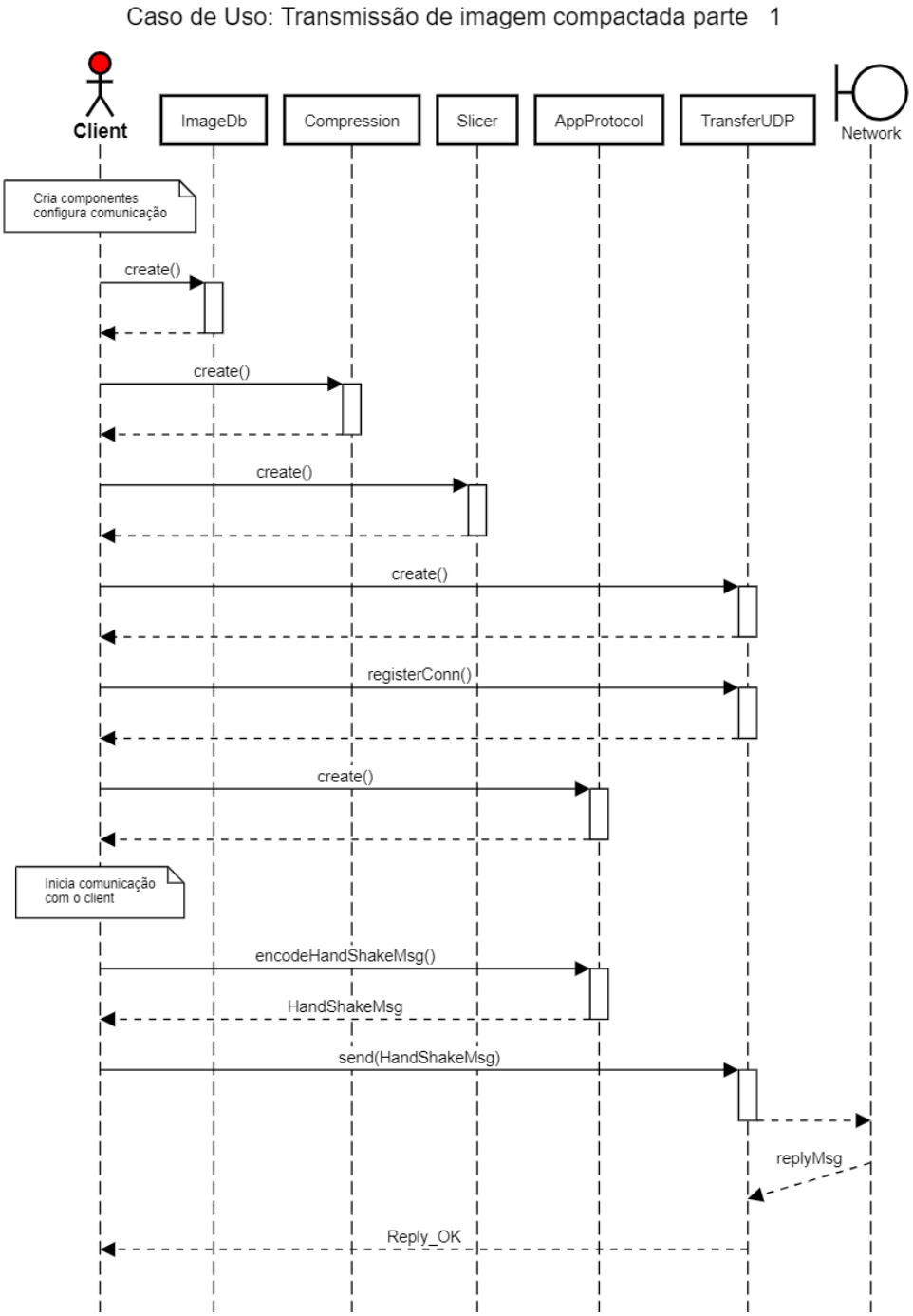


Fonte: elaborado pelo autor.

5.1.3 Caso de Uso: Transmissão de uma imagem compactada

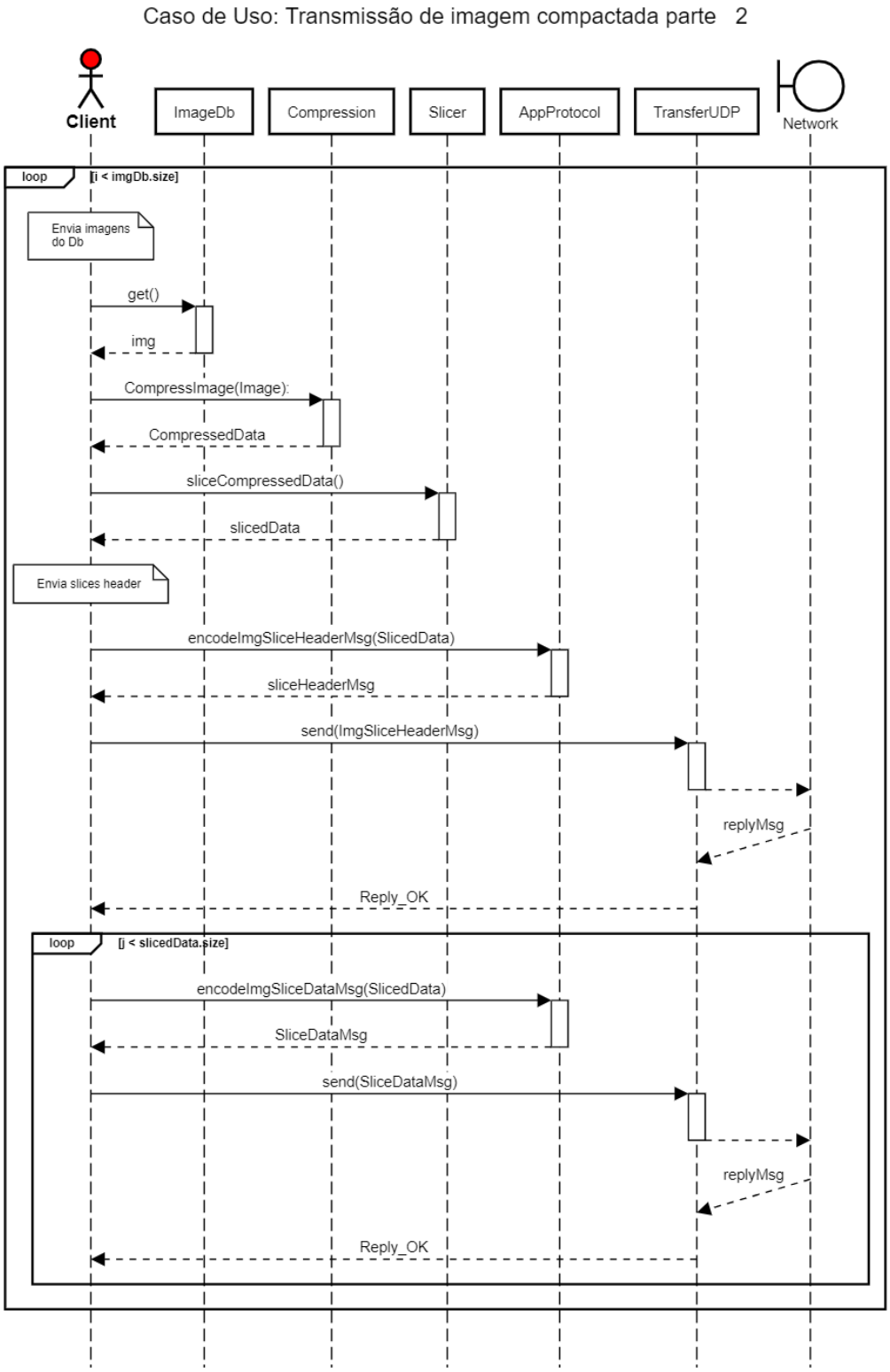
Para este caso de uso é exemplificado a transmissão de uma imagem compactada. O diagrama de sequência da API nos lados cliente e servidor são dados respectivamente nas figuras 35, 36, 37 e 38.

Figura 35: Diagrama de sequência da API: Cliente transmitindo imagem compactada (parte 1).



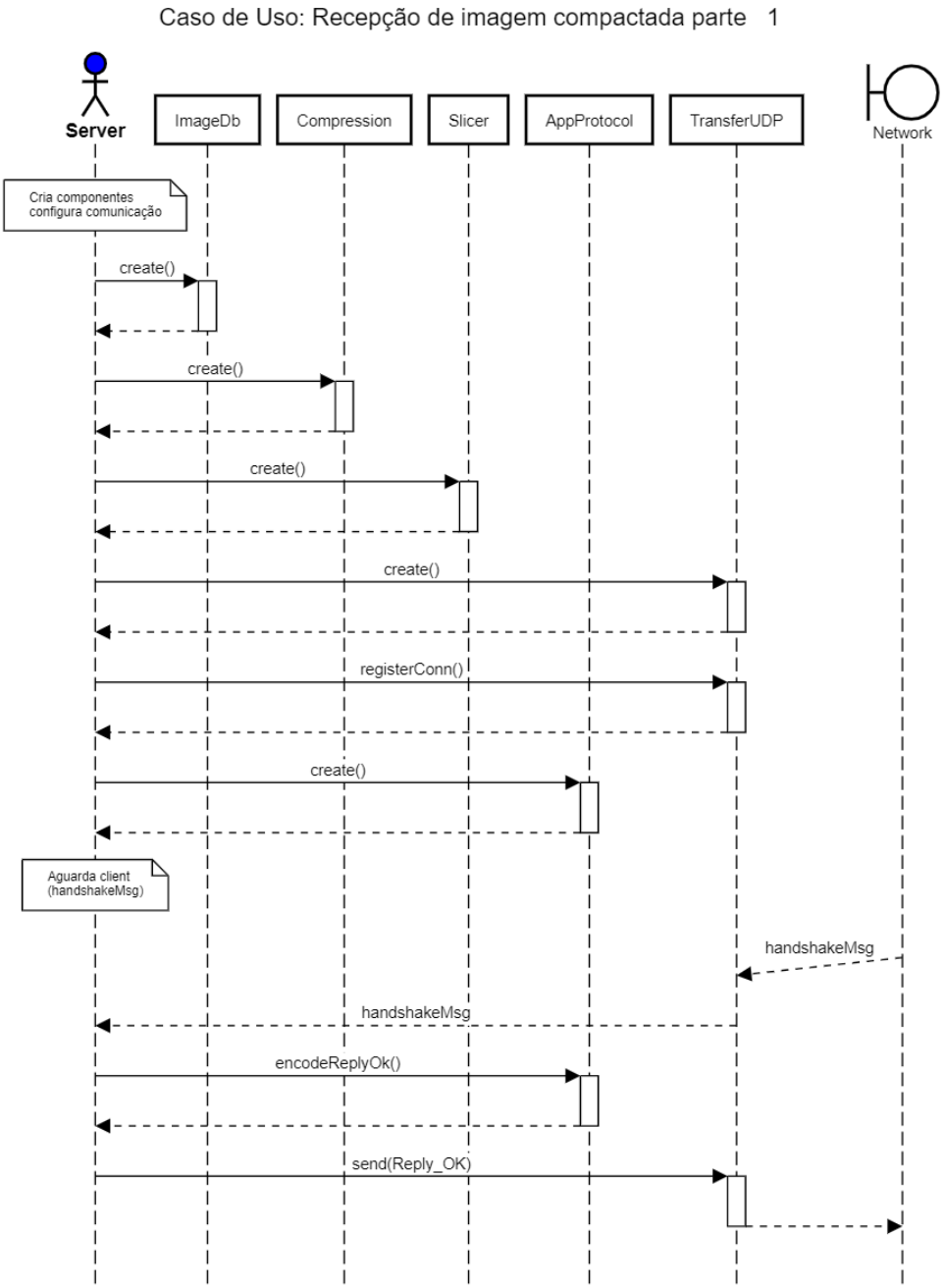
Fonte: elaborado pelo autor.

Figura 36: Diagrama de sequência da API: Cliente transmitindo imagem compactada (parte 2).



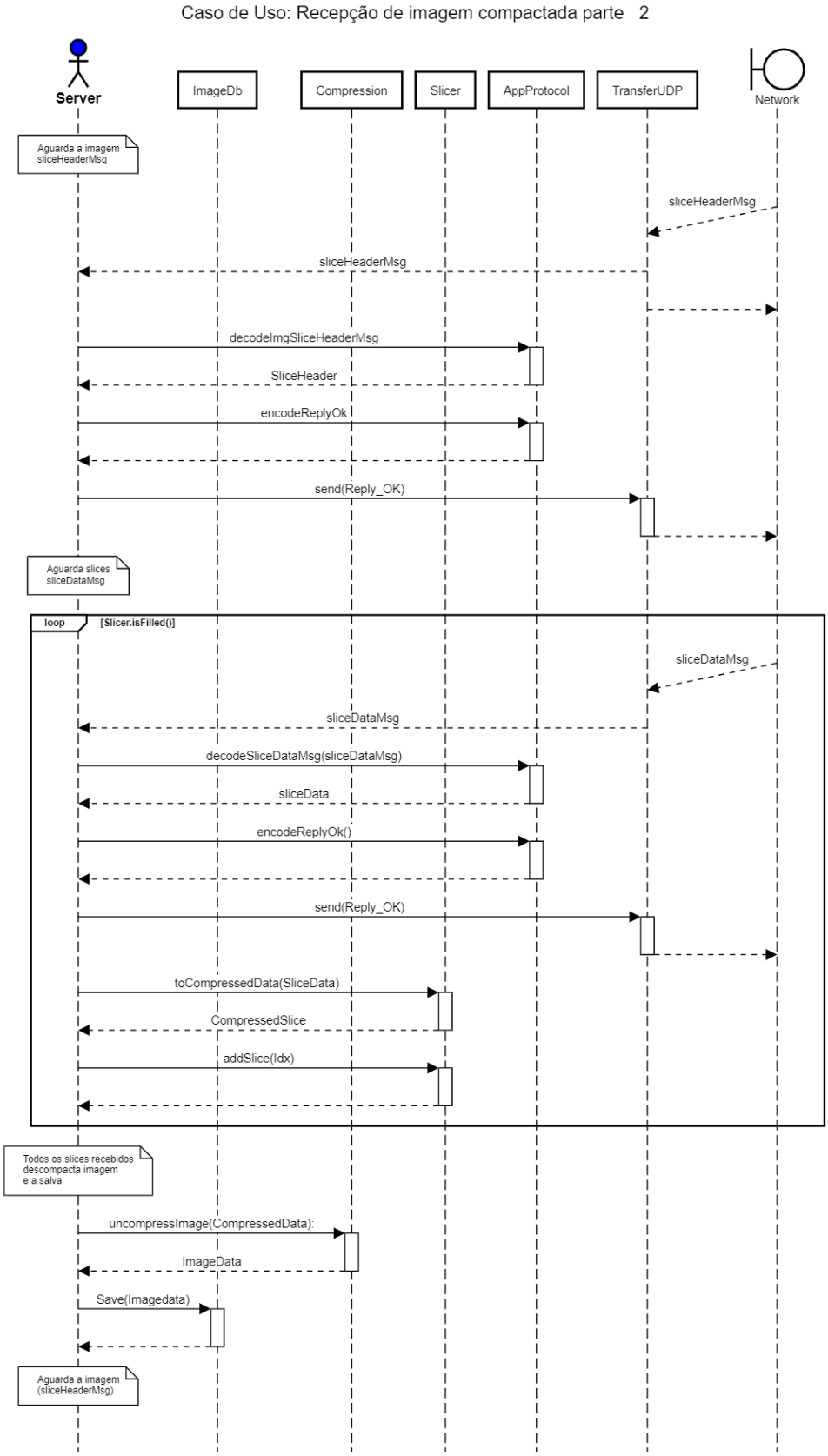
Fonte: elaborado pelo autor.

Figura 37: Diagrama de sequência da API: Servidor recebendo imagem compactada (parte 1).



Fonte: elaborado pelo autor.

Figura 38: Diagrama de sequência da API: Servidor recebendo imagem compactada (parte 2).



Fonte: elaborado pela autor.

6 Conclusão e trabalhos futuros

Com o desenvolvimento da indústria de semicondutores e a oferta de sensores de imagem a baixo custo, tem se tornado atrativo a integração destes componentes em aplicações IoT, todavia sensores de imagens ainda não tem plena integração nas WSN, que embarcam sistemas computacionais com baixo poder computacional, com limitação de energia e redes de baixa capacidade de transmissão.

De forma a contornar as limitações das WSN várias técnicas têm sido apresentadas na literatura para o tratamento e transmissão de imagens neste sistemas, já que o manuseio de imagens representa uma tarefa complexa e um grande desafio técnico neste contexto. As técnicas apresentadas na literatura objetivam a redução da complexidade dos algoritmos de compressão convencionais e da otimização de uso da rede, o qual permitem a redução do poder computacional requerido, do consumo de energia e do tráfego de dados na rede.

Neste trabalho foi apresentada uma API que encapsula tais técnicas em componentes. Adicionalmente, também foram definidos componentes para a manipulação de imagens visando sua utilização em simuladores de redes, tendo sido implementado um protótipo funcional para o sistema operacional Contiki-NG e simulado no Cooja. De forma a validar as interfaces e tipos definidos, assim como a integração no simulador, alguns casos de uso foram simulados. A API está plenamente funcional para o envio e recepção de imagens no ambiente de simulação utilizado.

Tal abordagem contribui para a padronização de um processo complexo utilizando-se técnicas de engenharia de software e também facilita o entendimento das etapas envolvidas por meio da decomposição em componentes. O uso da API propicia uma base de apoio a um sistema de experimentação e simulação para o tratamento e transmissão de imagens em WSN e aplicações IoT

Como foram identificados na literatura uma grande variedade de técnicas e tipos de redes, consequentemente para cada requisito de aplicação pode ser vantajoso a utilização de uma ou mais técnicas em conjunto para se contemplar os requisitos estabelecidos.

Embora nos simuladores seja possível a utilização de qualquer cenário relativo à resolução das imagens e às técnicas utilizadas, a API não prove nenhum mecanismo de avaliação de tempo de execução e de recursos consumidos como memória ou energia. Isso representa uma limitação quando da avaliação de um determinado cenário. Para obter tais informações é necessário a utilização dos recursos do simulador utilizado.

Relativo à evolução da pesquisa, foram necessários alguns ciclos de refinamento de revisão da literatura para que o escopo da pesquisa fosse corretamente delimitado. Contrariando a hipótese inicial da pesquisa de que somente técnicas de compressão de imagens seriam aplicáveis ao contexto de aplicações visuais IoT, técnicas de transmissão e de detecção de mudança também foram encontradas. A relevância das WSN foi identificada durante o trabalho, tornando seu estudo parte integrante do objetivo final do trabalho

Levando-se em consideração as limitações e temas relacionados na literatura são sugeridos os seguintes trabalhos futuros:

- **Simuladores de redes:** Para a extensão da API a outros sistemas de simulação é necessário o aprofundamento dos estudos em simuladores de redes, o qual é apresentado no apêndice A, onde são pesquisadas características básicas dos simuladores. Tal pesquisa poderia aprimorar os componentes associados a comunicação e também na adição de componentes para a medida de performance e consumo de energia das técnicas utilizadas.

- **Plataformas de HW IoT e WSN:** A pesquisa em plataformas de HW poderá melhorar o entendimento das limitações destes sistemas computacionais, já que nos trabalhos foram identificadas sistemas compatíveis com o Contiki-NG que requerem menos de 50 Kb de memória disponíveis até sistemas que rodam

versões reduzidas do Linux, como Raspberry. Este tema poderia resultar em otimizações dependendo do tipo de plataforma e auxiliar a utilização em sistemas reais e não somente em simuladores.

- **Segurança de redes:** É um tema de bastante interesse aos pesquisadores IoT e WSN, e também um requisito fundamental de qualquer aplicação. Durante a revisão da literatura foram identificados trabalhos com o tema associados a transmissão de imagens, o qual não foram considerados neste trabalho. De forma similar às técnicas de compressão de imagem, algoritmos de criptografia tem alta complexidade computacional e elevado consumo de energia. Como resultado desta pesquisa um componente de segurança poderia ser adicionado à API, tornando-a mais completa.

Anexos

A Simuladores de redes IoT

A simulação é um processo de recriação do mundo real através de um ambiente controlado, que pode ser um programa de computador ou um protótipo de dimensões reduzidas. Existem grandes benefícios na utilização de simuladores, que é uma prática bem comum em todos os ramos da engenharia e tem auxiliado o desenvolvimento de diversas disciplinas.

Através da criação de modelos para representar o sistema real, a simulação permite a análise preliminar de desempenho de um sistema sem a sua construção, reduzindo os custos e o tempo de desenvolvimento. Além do mais, é possível descobrir problemas críticos e impeditivos que usualmente só seriam encontrados após a construção do sistema real.

Como já apresentado, o IoT é composto por diversas tecnologias, padrões e protocolos, criando inúmeras possibilidades de casos de uso, o que torna a simulação uma ferramenta de grande importância no desenvolvimento destas soluções, já que a simulação usualmente envolve menos custos que a construção de protótipos físicos.

A escalabilidade é uma dúvida frequente levantada pelo pesquisadores a respeito das tecnologias voltadas para o IoT (D´ANGELO; FERRETTI; GHINI, 2017), então a simulação de rede tem um papel chave na análise do comportamento dessas tecnologias de rede e protocolos quando há um grande número de dispositivos conectados, já que são previstos bilhões de dispositivos inteligentes conectados as redes IoT.

Como em qualquer área de engenharia, a simulação de redes já é uma prática bastante popular entre os pesquisadores em todo o mundo, como mencionado por (SARKAR; HALIM, 2008). Segundo tais autores o objetivo de usar qualquer simulador é modelar e prever com precisão o comportamento de um sistema do mundo real.

A utilização de ferramentas de simulação é essencial (NOVELLI et al., 2018), um modelo de simulação pode avaliar se a combinação de tecnologias escolhidas é uma boa solução para determinado cenário, com avaliação quantitativa e qualitativa (D´ANGELO; FERRETTI; GHINI, 2017).

No trabalho de (OJIE; PEREIRA, 2017) é apresentada uma revisão literária de ferramentas de simulação em IoT. Os autores argumentam sobre a importância da simulação e da dificuldade da seleção de uma ferramenta devido ao fato do IoT ser uma área recente e abrangendo diversas tipos de aplicações como cuidados de saúde, transporte, agricultura e cidades inteligentes, além de outras. Assim, seguem os autores, para encurtar o tempo de desenvolvimento destas aplicações de maneira segura, as ferramentas de simulação são essenciais. O trabalho identificada algumas ferramentas apropriadas a simulação em IoT:

- **NS-3:** É um simulador de eventos discretos que poder ser usado para a simulação de redes. Esta ferramenta pode ser usada para fins de pesquisa e desenvolvimento de um ambiente de simulação, compreende um fluxo de trabalho de simulação que pode ser usado para configuração de dispositivos de rede e análise (OJIE; PEREIRA, 2017);
- **NS-2:** Plataforma de simulação utilizada para a pesquisa acadêmica. Ele é um simulador de eventos discretos que fornece suporte para o simulação de protocolos de rede e roteamento, incluindo protocolos de rede multicast com e sem fio (OJIE; PEREIRA, 2017);
- **OMNeT++:** É um simulador de eventos discreto desenvolvido em C ++. A ferramenta suporta vários componentes de rede, com licença aberta, uma IDE e um kernel de simulação integrados (OJIE; PEREIRA, 2017);

- **CupCarbon:** É um simulador que pode ser usado por pesquisadores e desenvolvedores na avaliação dos conceitos básicos de redes de sensores sem fio em um ambiente de cidade inteligente (OJIE; PEREIRA, 2017);
- **Castalia:** É uma plataforma de simulação projetada principalmente para a estrutura de redes sem fio com dispositivos embarcados de baixo consumo (OJIE; PEREIRA, 2017).

Adicionalmente aos simuladores já apresentados, foi identificados na literatura o Cooja como ferramentas de grande relevância para a simulação de redes e protocolos de IoT. O Cooja é o simulador integrado ao sistema operacional Contiki-NG, que é especializado em protocolos de IoT (CONTIKI-NG, 2021b), ele é um simulador de evento discreto utilizado para simulação de domínio específico em larga escala que oferece boa escalabilidade (FERRETTI et al., 2017).

Um conjunto de características e protocolos nativamente atendidos pelas ferramentas mencionadas anteriormente é apresentado comparativamente nos quadros 5 e 6:

Quadro 5: Comparativo de características de simuladores de rede.

Simulador	**Licença**	**Plataforma Simulação**	**Linguagem**	**Escalabilidade >100 nós**	**Aplicações**
NS-2	GPL	Universal	C, C++	Sim	Pesquisa Acadêmica
NS-3	Open	Universal	C++, Python	Sim	Pesquisa Acadêmica
OMNET++	Academic	TinyOS	C, C++	Sim	Pesquisa Acadêmica
CupCarbon	Open	Universal	Java	Sim	Pesquisa acadêmica Desenv. sistemas
Castalia	GNU	Universal	Java	Sim	Pesquisa acadêmica Desenv. sistemas
Cooja	BSD-3	Contiki-NG	C	Sim	Desenv. Sistemas

Fonte: adaptada pelo autor, de acordo com(OJIE; PEREIRA, 2017), (FERRETTI et al., 2017) e (CONTIKI-NG, 2021b).

Quadro 6: Comparativo de protocolos IoT nativamente suportados pela ferramenta de simulação.

		Camada do modelo OSI			
Simulador	**Especializado Redes sensores sem fio ?**	**Física Enlace**	**Rede**	**Transporte**	**Aplicação Apresentação Sessão**
NS-2	Não	Wi-Fi 802.11	-	UDP	Sim
NS-3	Não	Wi-Fi IEEE 802.15.4	-	UDP	-
OMNET++	Não	WiFi IEEE 802.15.4	-	UDP	-
CupCarbon	Sim	WiFi IEEE 802.15.4 Lora	ZigBee	-	-
Castalia	Sim	IEEE 802.15.4	ZigBee	-	-
Cooja	Sim	IEEE 802.15.4	6LowPan RPL	UDP	COAP MQTT

Fonte: adaptado pelo autor, de acordo com (OJIE; PEREIRA, 2017) e (CONTIKI-NG, 2021b).

B Interfaces da API

Nesta seção são apresentados as interfaces dos componentes da API discutidos na seção 4.

B.1 Componente de Definição de Imagens

```
typedef enum {
        UNCOMPRESSED,
        JPG_IMG
} tImageFormat;

typedef struct tImage{
        unsigned int height;
        unsigned int width;
        unsigned int bitDepth;
        unsigned int channels;
        unsigned char ** data;
        unsigned int id;
} tImage;
```

B.2 Componente repositório de imagem

```
typedef struct tImageDb {
```

```
                tImage ** images;
                int size;
                tImage * (*get)(struct tImageDb*, int);
                int (*load)(struct tImageDb*);
                void (*save)(struct tImage*);
} tImageDb;

tImageDb * tImageDb_create();
int tImageDb_destroy(tImageDb * imgDb);
```

B.3 Componente de Dados Comprimidos

```
typedef struct tCompressedData {
        unsigned int size; // in unsigned chars
        unsigned char * data;
} tCompressedData;
```

B.4 Componente de detecção de mudança

```
typedef enum {
        NO_CHANGE=1,
        FULL_CHANGE=2,
        PARTIAL_CHANGE=3
} tChangedStatus;

typedef struct tChangeDetection {
        void(*Init)(struct tImage *);
        tCompressedData*(*changeDetect)(struct tChangeDetection*, tImage*);
        void(*setImage)(struct tImage*);
        tImage*(*setChange) (struct tCompressedData*);
}tChangeDetection;

void Init(tImage *image);
tCompressedData* changeDetect(tChangeDetection* changeDetection, tImage* img);
void setImage(tImage* Img);
tImage * setChange (tCompressedData* compresseddata);
```

B.5 Componente de Compressão

```
typedef struct tCompression{

        tCompressedData*(*compressImage)(struct tCompression*, tImage*);
        tImage*(*uncompressImage) (struct tCompressedData*);
} tCompression;
```

```
tCompression * tCompression_create();
void tCompression_destroy(tCompression *compression);
tCompressedData * tCompression_compressImage(tCompression*, tImage* img);
tImage * uncompressImage(tCompressedData* compresseddata);
```

B.6 Componente de fragmentação

```
typedef struct tSlicedData {
        unsigned int slices;
        unsigned int sliceSize;
        unsigned int lastSliceSize;
        unsigned int * filled;
        unsigned char ** data;
} tSlicedData;

typedef struct tSlicer {
        tSlicedData* (*sliceImage)(tImage *);
        tSlicedData* (*sliceCompressedData)(tCompressedData *);
        int (*isFilled)(tSlicedData*);
        unsigned char ** (*toImgData)(tSlicedData *, int, int, int);
        unsigned char * (*toCompressedData)(tSlicedData *, int);
        unsigned char * (*getSlice)(tSlicedData *, int);
        int (*addSlice) (tSlicedData *, unsigned char *, int);
} tSlicer;

tSlicer * tSlicer_create();
void tSlicer_destroy(tSlicer * slicer);
```

B.7 Componente de Transferência UDP

```
#define TRANSFER_BUFFER_SIZE     90
#define TRANSFER_DATA_BUFFER_SIZE
         TRANSFER_BUFFER_SIZE-16-4

typedef struct tRawReceivedMsg \{
        unsigned char * buffer;
        int receivedSize;
} tRawReceivedMsg;

void tRawReceivedMsg_free(tRawReceivedMsg *);

typedef struct tTransferUDP {
        struct simple_udp_connection connection;
        int (*registerComm)(struct tTransferUDP *, int, int,
        void (callback)(struct simple_udp_connection *, const     uip_ipaddr_t *, uint16_t , const
```

```
        uint16_t , const uint8_t *, uint16_t));
        int (*send)(struct tTransferUDP *, unsigned char *, unsigned int);
} tTransferUDP;

tTransferUDP * tTransferUDP\_create();
void tTransferUDP_destroy(tTransferUDP* server);
```

B.8 Componente de Protocolo de Aplicação

```
typedef enum {
        HANDSHAKE_MSG_ID=1,
        HANDSHAKE_REPLY_MSG_ID=2,
        IMG_SLICE_HEADER_MSG_ID=3,
        SLICE_HEADER_MSG_ID=4,
        SLICE_DATA_MSG_ID=5,
        REPLY_MSG_ID=6,
} Message_ID;

typedef enum {
        REPLY_OK=1,
        REPLY_FAIL
} Reply_STATUS;

typedef struct tImgSliceHeaderMsg {
        Message_ID id:4;
        unsigned int height:16;
        unsigned int width:16;
        unsigned int bitDepth:16;
        unsigned int channels:16;
        unsigned int slices:16;
        unsigned int sliceSize:16;
        unsigned int lastSliceSize:16;
} tImgSliceHeaderMsg;

typedef struct tSliceDataMsg {
        Message_ID id:4;
        unsigned int position:16;
        unsigned int dataSize:16;
        unsigned char data[TRANSFER_DATA_BUFFER_SIZE];
} tSliceDataMsg;

typedef struct tHandShakeMsg {
        Message_ID id:4;
} tHandShakeMsg;

typedef struct tReplyMsg{
        Message_ID id:4;
        unsigned int status:4;
```

```
} tReplyMsg;

typedef struct tAppProtocol {
        int (*isHandShakeMsg)(tRawReceivedMsg *);
        int (*isHandShakeReply)(tRawReceivedMsg *);
        int (*isReplyOK)(tRawReceivedMsg *);
        int (*getMsgId)(unsigned char *);
        tImgSliceHeaderMsg * (*encodeImgSliceHeaderMsg)(tImage *, tSlicedData *);
        tSliceDataMsg * (*encodeSliceDataMsg)(unsigned char *, int, int);
        tHandShakeMsg * (*encodeHandShakeMsg)();
        tReplyMsg * (*encodeReplyOk)();
        tImgSliceHeaderMsg * (*decodeImgSliceHeaderMsg)(tRawReceivedMsg *msg);
        tSliceDataMsg * (*decodeSliceDataMsg)(tRawReceivedMsg *msg);
        tHandShakeMsg * (*decodeHandShakeMsg)(tRawReceivedMsg *msg);
        tReplyMsg * (*decodeReply)(tRawReceivedMsg *msg);
} tAppProtocol;
tAppProtocol * tAppProtocol_create ();
void tAppProtocol_destroy(tAppProtocol*);
```

Referências

4DSYSTEMS. *4dsystems web page*. 2020. Disponível em: <https://4dsystems.com.au/ucam-iii>.

ABDULLAH, J.; ALDOORI, A.; JAMIL, A. Image transmission scheme with joint routing and hwt-based compression in wireless sensor networks. In: *2019 9th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. [S.l.: s.n.], 2019. p. 152–157.

ACM. *ACM Digital Library*. 2021. Disponível em: <https://dl.acm.org>.

ADELANTADO, F. et al. Understanding the limits of lorawan. *IEEE Communications Magazine*, v. 55, n. 9, p. 34–40, 2017.

ALLIANCE, W.-F. *Wi-Fi Alliance*. 2019. Disponível em: <https://www.wi-fi.org>.

ALTAYEB, M.; SHARIF, S.; ABDELLA, S. The Internet-of-Things and Integration with Wireless Sensor Network Comprehensive Survey and System Implementation. *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering, ICCCEEE 2018*, IEEE, p. 1–6, 2018.

BHARATH, K.; PADMAJADEVI, G. International Journal of Computer Science and Mobile Computing Hybrid Compression Using DWT-DCT and Huffman Encoding Techniques for Biomedical Image and Video Applications. *Ijcsmc*, v. 2, n. 5, p. 255–261, 2013. Disponível em: <www.ijcsmc.com>.

BOUGUEZEL, S.; AHMAD, M. O.; SWAMY, M. N. S. Binary discrete cosine and hartley transforms. *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 60, n. 4, p. 989–1002, 2013.

CHANG, K. H.; MASON, B. The IEEE 802.15.4g standard for smart metering utility networks. *2012 IEEE 3rd International Conference on Smart Grid Communications, SmartGridComm 2012*, p. 476–480, 2012.

CHEN, T.; EAGER, D.; MAKAROFF, D. Efficient image transmission using lora technology in agricultural monitoring iot systems. *Proceedings - 2019 IEEE International Congress on Cybermatics: 12th IEEE International Conference on Internet of Things, 15th IEEE International Conference on Green Computing and Communications, 12th IEEE International Conference on Cyber, Physical and Social Computing and 5th IEEE International Conference on Smart Data, iThings/GreenCom/CPSCom/SmartData 2019*, IEEE, p. 937–944, 2019.

CONTIKI-NG, P. *Contiki-ng Project Git-Hub*. 2021. Disponível em: <https://github.com/contiki-ng/contiki-ng>.

CONTIKI-NG, W. *Contiki Wiki*. 2021. Disponível em: <https://github.com/contiki-ng/contiki-ng/wiki/Toolchain-installation-on-Linux>.

COSTA, D. G. Visual sensors hardware platforms: A review. *IEEE Sensors Journal*, v. 20, n. 8, p. 4025–4033, 2020.

DEEPTHI, A. S.; RAO, S. E.; PRASAD, G. M. N. Image transmission and compression techniques using SPIHT and EZW in WSN. *Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018*, IEEE, n. Icisc, p. 1146–1149, 2018.

D´ANGELO, G.; FERRETTI, S.; GHINI, V. Modeling the internet of things: a simulation perspective. In: *2017 International Conference on High Performance Computing Simulation (HPCS)*. [S.l.: s.n.], 2017. p. 18–27.

ELSEVIER. *Scopus Preview*. 2021. Disponível em: <https://www.scopus.com/home.uri>.

FELEMBAN, E.; NASEER, A.; AMJAD, A. Priority-based routing framework for image transmission in visual sensor networks: Experimental analysis. *International Journal of Advanced Computer Science and Applications*, v. 11, n. 1, p. 668–677, 2020. ISSN 21565570.

FERRETTI, S. et al. The quest for scalability and accuracy: Multi-level simulation of the internet of things. *arXiv*, p. 2–9, 2017. ISSN 23318422.

GARTNER, I. *Gartner Identifies Top 10 Strategic IoT Technologies and Trends*. 2018. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends>.

GOPALUNI, J. et al. Graphical User Interface for OpenThread. *HONET-ICT 2019 - IEEE 16th International Conference on Smart Cities: Improving Quality of Life using ICT, IoT and AI*, IEEE, p. 235–237, 2019.

HEYNE, B. et al. A computationally efficient high-quality cordic based dct. In: *2006 14th European Signal Processing Conference*. [S.l.: s.n.], 2006. p. 1–5.

IEEE. *IEEE Xplore*. 2021. Disponível em: <https://ieeexplore.ieee.org/Xplore/home.jsp>.

IETF, I. E. T. F. *Internet Engineering Task Force*. 2021. Disponível em: <https://tools.ietf.org>.

ITU. *ITU EXPLAINERS, INTERNET OF THINGS*. 2018. Disponível em: <https://www.gp-digital.org/wp-content/uploads/2018/02/ITU_IOT2.pdf>.

JAVAID, M. et al. Sensors for daily life: A review. *Sensors International*, v. 2, p. 100121, 2021. ISSN 2666-3511. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666351121000425>.

KISHK, A. et al. Hybrid compression algorithm for wireless sensor network. *Journal of Advances in Computer Networks*, v. 2, p. 147–150, 06 2014.

KOUADRIA, N. et al. Low complexity dct for image compression in wireless visual sensor networks. *Electronics Letters*, v. 49, p. 1531–1532, 11 2013.

KUMAR, T.; MANE, P. B. ZigBee topology: A survey. *2016 International Conference on Control Instrumentation Communication and Computational Technologies, ICCICCT 2016*, p. 164–166, 2017.

LECUIRE, V.; MAKKAOUI, L.; MOUREAUX, J.-M. Fast zonal dct for energy conservation in wireless image sensor networks. *Electronics Letters*, v. 48, p. 125–127, 01 2012.

LIN, J. et al. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, v. 4, n. 5, p. 1125–1142, 2017.

MAJDOUL, S.; SRIFI, M. N. Wireless sensors networks challenges. In: *Proceedings of the Mediterranean Symposium on Smart City Application*. New York, NY, USA: Association for Computing Machinery, 2017. (SCAMS '17). ISBN 9781450352116. Disponível em: <https://doi.org/10.1145/3175628.3175637>.

MATIN, M. A.; ISLAM, M. M. *Overview of Wireless Sensor Network*. 2012. Disponível em: <https://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/overview-of-wireless-sensor-network>.

MEHTA, S.; PATEL, A.; MEHTA, J. Ccd or cmos image sensor for photography. In: *2015 International Conference on Communications and Signal Processing (ICCSP)*. [S.l.: s.n.], 2015. p. 0291–0294.

MENDELEY. *Mendeley Desktop*. 2021. Disponível em: <https://www.mendeley.com/download-reference-manager/windows>.

MINU, S.; SHETTY, A. A Comparative Study of Image Change Detection Algorithms in MATLAB. *Aquatic Procedia*, Elsevier B.V., v. 4, n. Icwrcoe, p. 1366–1373, 2015. ISSN 2214241X. Disponível em: <http://dx.doi.org/10.1016/j.aqpro.2015.02.177>.

MITTAL, A. et al. Entropy based image segmentation with wavelet compression for energy efficient lte systems. In: *2016 23rd International Conference on Telecommunications (ICT)*. [S.l.: s.n.], 2016. p. 1–6.

MOCNEJ, J. Network traffic characteristics of the iot application use cases. In: . [S.l.: s.n.], 2018.

MOZAMMEL, M.; CHOWDHURY, H.; KHATUN, A. Image compression using discrete wavelet transform. *International Journal of Computer Science Issues*, v. 9, 07 2012.

NAUMAN, A. et al. Multimedia internet of things: A comprehensive survey. *IEEE Access*, v. 8, p. 8202–8250, 2020. ISSN 21693536.

NOVELLI, L. et al. Application protocols and wireless communication for iot: A simulation case study proposal. In: *2018 11th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*. [S.l.: s.n.], 2018. p. 1–6.

NXP. *Kinetis® W Series KW41Z MCUs Block Diagram*. 2020. Disponível em: <https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/kinetis-cortex-m-mcus/w-serieswireless-conn.m0-plus-m4/kinetis-kw41z-2.4-ghz-dual-mode-bluetooth-low-energy-and-802.15.4-wireless-radio-microcontroller-mcu-based-on-arm-cortex-m0-plus-core:KW41Z>.

NXP. *KW41Z: Kinetis® KW41Z-2.4 GHz Dual Mode: Bluetooth® Low Energy and 802.15.4 Wireless Radio Microcontroller (MCU) based on Arm® Cortex®-M0+ Core*. 2020. Disponível em: <https://www.nxp.com/products/wireless/thread/kinetis-kw41z-2-4-ghz-dual-mode-bluetooth-low-energy-and-802-15-4-wireless-radio-microcontroller-mcu-based-on-arm-cor KW41Z?tab=Buy_Parametric_Tab\#/>.

OJIE, E.; PEREIRA, E. Simulation tools in internet of things: A review. In: *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2017. (IML '17). ISBN 9781450352437. Disponível em: <https://doi.org/10.1145/3109761.3158400>.

OMNIVISION. *OV7675 - CMOS VGA (640x480) Image Sensor with OmniPixel®3-HS Technology*. 2021. Disponível em: <https://www.ovt.com/sensors/OV7675>.

ORACLE. *OpenJDK Project*. 2021. Disponível em: <https://openjdk.java.net>.

PATEL, N.; CHAUDHARY, J. Energy efficient WMSN using image compression: A survey. *IEEE International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2017 - Proceedings*, n. Icimia, p. 124–128, 2017.

PATEL, N. R.; KUMAR, S. Wireless sensor networks' challenges and future prospects. *Proceedings of the 2018 International Conference on System Modeling and Advancement in Research Trends, SMART 2018*, IEEE, p. 60–65, 2018.

PHAM, C. Low-cost, low-power and long-range image sensor for visual surveillance. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, v. 03-07-Octo, p. 35–40, 2016.

POKHREL, S. R.; WILLIAMSON, C. Modeling compound tcp over wifi for iot. *IEEE/ACM Transactions on Networking*, v. 26, n. 2, p. 864–878, 2018.

RAZA, U.; KULKARNI, P.; SOORIYABANDARA, M. Low power wide area networks: An overview. *IEEE Communications Surveys Tutorials*, v. 19, n. 2, p. 855–873, 2017.

SARKAR, N.; HALIM, S. Simulation of computer networks: Simulators, methodologies and recommendations. In: . [S.l.: s.n.], 2008.

SILVA, E. L.; MENEZES, E. M. *Metodologia da pesquisa e elaboração de dissertação 4. ed.* Florianópolis: UFSC, 2005. 138 p., 2005. Disponível em: <www.posarq.ufsc.br/download/metPesq.pdf>.

STANLEY, D. *IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS, The Working Group for WLAN Standards*. 2019. Disponível em: <http://grouper.ieee.org/groups/802/11/>.

WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computação 6. ed.* [S.l.: s.n.], 2009.

ZEITZ, K. et al. Designing a micro-moving target IPv6 defense for the internet of things. *Proceedings - 2017 IEEE/ACM 2nd International Conference on Internet-of-Things Design and Implementation, IoTDI 2017 (part of CPS Week)*, p. 179–184, 2017.

ZOLERTIA. *Zolerti*. 2021. Disponível em: <http://wiki.zolertia.com/wiki/index.php/Main_Page>.