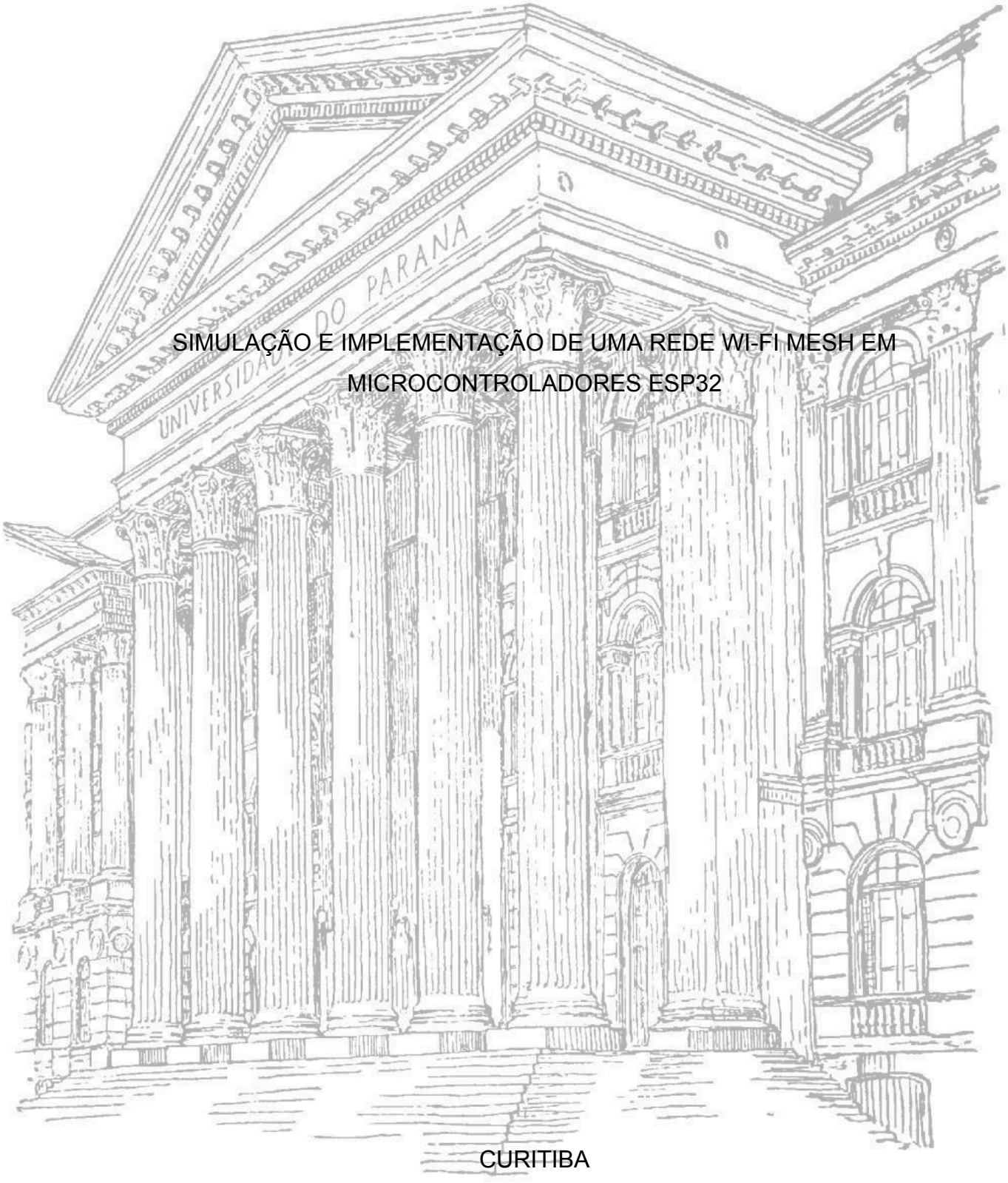
UNIVERSIDADE FEDERAL DO PARANÁ

LORENZO GALLASSINI PICININ

SIMULAÇÃO E IMPLEMENTAÇÃO DE UMA REDE WI-FI MESH EM MICROCONTROLADORES ESP32



CURITIBA

2025

LORENZO GALLASSINI PICININ

SIMULAÇÃO E IMPLEMENTAÇÃO DE UMA REDE WI-FI MESH EM MICROCONTROLADORES ESP32

Monografia de trabalho de conclusão de curso, disciplina TE348, apresentada como requisito parcial à obtenção do título de Engenheiro Eletricista, curso de Engenharia Elétrica, setor de Tecnologia, Universidade Federal do Paraná.

Orientador: Prof. Dr. Eduardo Parente Ribeiro.

CURITIBA

2025

TERMO DE APROVAÇÃO

LORENZO GALLASSINI PICININ

SIMULAÇÃO E IMPLEMENTAÇÃO DE UMA REDE WI-FI MESH EM MICROCONTROLADORES ESP32

Trabalho de Conclusão de Curso apresentado à disciplina TE- 348 – Trabalho de Conclusão de Curso II do curso de Graduação em Engenharia Elétrica com Ênfase em Eletrônica e Telecomunicações da Universidade Federal do Paraná, como requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica, pela seguinte banca examinadora:

_____

Prof. Dr. Eduardo Parente Ribeiro
Orientador

_____

Prof. Dr. Edson José Pacheco
UFPR - DELT

_____

M. Sc. Polyana Camargo de Lacerda
UFPR - DELT

CURITIBA, 3 de julho de 2025.

AGRADECIMENTOS

Agradeço à minha mãe Angela Gallassini Picinin, pelo suporte em toda minha trajetória, pela força nos momentos difíceis e pelo amor que sempre dedicou à mim e minha irmã.

Agradeço à minha irmã e meus relativos, que sempre acompanharam meus passos e me proporcionaram muitos momentos de felicidade.

Agradeço aos meus professores e professoras, pela dedicação e domínio com os quais transferem seus conhecimentos para os alunos.

Agradeço a todos os meus colegas e amigos, que fizeram parte da minha jornada.

Agradeço a todos aqueles, que com respeito e sabedoria, trabalham na construção de um futuro melhor.

Dedico este trabalho ao meu falecido pai Maurício Picinin.

RESUMO

O presente trabalho realizou uma pesquisa experimental a respeito do desempenho de uma rede Wi-Fi *Mesh* visando aplicações para internet das coisas, utilizando, para isso, ambientes simulados em *software* OMNet++ e testes práticos realizados com microcontroladores ESP32. Tanto a simulação quanto a prática foram realizadas em um cenário interno residencial. Na etapa de simulação foram contabilizados os efeitos de multicaminho e ruído através de um modelo de desvanecimento *Log Normal Shadowing*, além das características de rádio frequência dos microcontroladores escolhidos. Foram realizados múltiplos testes, com diferentes taxas de transferência de dados, protocolos de transporte e disposição física. As métricas analisadas foram a taxa efetiva de bits (*Throughput*), latência, medida na forma de RTT (*Round-Trip-Time*) e ICMP, e taxa de perda de pacotes. Para a implementação nos microcontroladores, foi utilizada a rede ESP-WIFI-MESH desenvolvida pela empresa Espressif *Systems*. Os resultados mostraram um desempenho similar entre a simulação e a prática com relação à taxa de dados e perda de pacotes. A latência se mostrou maior nos microcontroladores, quando comparada à simulação, em até 15 vezes. A maior taxa de transferência foi obtida no teste prático com protocolo UDP e banda HT40 no padrão 802.11n , chegando em 692 kbps para uma conexão *uplink*. Porém, a taxa de perda com protocolo UDP chegou a 74%, enquanto em conexões TCP a perda foi nula. A rede Wi-Fi *Mesh* é, portanto, uma alternativa de comunicação de alta taxa de transferência para dispositivos embarcados, se comparada com redes usualmente utilizadas para sistemas IoT (*Internet of Things*), como Thread, Zigbee e BLE.

Palavras-chave: Wi-Fi Mesh. OMNet++. ESP32. Matter.

ABSTRACT

In this work an experimental research was conducted about the performance of a Wi-Fi Mesh network with internet of things applications in mind, using, for this, simulated environments in OMNet++ software and practical tests with ESP32 microcontrollers. Both simulation and tests were conducted in an internal residential scenario. In the simulation section, effects of multipath and noise were taken into account using the Log Normal Shadowing model of fading channel and the radiofrequency characteristics of the microcontrollers. Multiple tests were made with different data rates, transportation protocols and physical displacement of the nodes. The analysed metrics were throughput, latency, measured in the form of RTT (Round-Trip-Time) and ICMP requests, and packet loss ratio. For the implementation in the microcontrollers, the ESP-WIFI-MESH network developed by Espressif Systems was used. The results showed a similar performance between the simulation and practice regarding the throughput and packet loss ratio. The latency was higher in the microcontrollers up to 15 times when compared to the simulation results. The biggest throughput obtained in the practical tests was reached with the UDP protocol and HT40 band of the 802.11n standard, reaching up to 692 kbps for uplink. In contrast, the packet loss ratio with UDP reached 74%, while with TCP connections the packet loss was null. The Wi-Fi Mesh network is a high throughput alternative for wireless communication in embedded systems, when compared to other frequently used IoT networks like Thread, Zigbee and BLE.

Key-words: Wi-Fi Mesh. OMNet++. ESP32. Matter.

LISTA DE SIGLAS

AP - *Access Point*

BLE - *Bluetooth Low Energy*

CASE - *Certificate Authenticated Session Establishment*

DHCP - *Dynamic Host Configuration Protocol*

HT40 - *High Throughput 40 MHz*

ICMP - *Internet Control Message Protocol*

IoT - *Internet of Things*

MAC - *Medium Access Control*

MIMO - *Multiple Input Multiple Output*

NAT - *Network Address Translation*

OFDM - *Orthogonal Frequency Division Multiplexing*

PASE - *Protocol Authenticated Session Establishment*

QAM - *Quadrature Amplitude Modulation*

RSSI - *Received Signal Strength Indicator*

RTT - *Round Trip Time*

STA - *Station*

TCP - *Transmission Control Protocol*

TSMC - *Taiwan Semiconductor Manufacturing Company*

UDP - *User Datagram Protocol*

WLAN - *Wireless Local Area Network*

WMN - *Wireless Mesh Network*

SUMÁRIO

1 INTRODUÇÃO....10
1.1 CONTEXTO....10
1.2 OBJETIVOS....11
1.2.1 Objetivo Geral....11
1.2.2 Objetivos Específicos....11
2 REVISÃO BIBLIOGRÁFICA....12
2.1 O PADRÃO MATTER....12
2.1.1 Arquitetura de Rede....13
2.1.2 Topologia de Rede....14
2.1.3 Comissionamento....15
2.2 REDES WI-FI MESH....16
2.2.1 Wireless Mesh Networks (WMN)....16
2.2.2 IEEE 802.11....18
2.2.3 O Padrão WLAN 802.11s para Redes Mesh....20
2.3 ESP-WIFI-MESH....24
3. MATERIAIS....30
3.1 SOFTWARE....30
3.2 HARDWARE....31
4. METODOLOGIA DE DESENVOLVIMENTO....33
4.1 SIMULAÇÃO DA REDE WI-FI MESH....33
4.1.1 O Arquivo de Configuração da Rede....33
4.1.2 Arquivo de Simulação....34
4.1.3 Simulação com Três Nós e Protocolo TCP....36
4.1.4 Simulação TCP com Alta Taxa de Dados....37
4.1.5 Simulação com 12 Nós e Protocolo TCP....37
4.1.6 Simulação com Três Nós e Protocolo UDP....38
4.2 TESTES PRÁTICOS COM A REDE WI-FI MESH....39
4.2.1 Implementação Prática da Rede Wi-Fi Mesh....40
4.2.2 Etapas do Teste Prático....41
4.3 ANÁLISE DE RESULTADOS....45
5. RESULTADOS....47
5.1 RESULTADOS DA SIMULAÇÃO....47
5.1.1 Resultados da Rede com Três Nós e Protocolo TCP....47
5.1.2 Resultados da Rede TCP com Alta Taxa de Dados....48
5.1.3 Resultados da Rede com Doze Nós TCP....49
5.1.4 Resultados da Rede com Três Nós e Protocolo UDP....53
5.2 RESULTADOS PRÁTICOS....54
5.2.1 Resultado do Teste Prático TCP com Baixo Volume de Dados....55
5.2.2 Resultado do Teste Prático TCP com Alta Taxa de Dados....56
5.2.3 Resultado do Teste Prático UDP....58
6. CONCLUSÃO....61
6.1 TRABALHOS FUTUROS....62

7. BIBLIOGRAFIA UTILIZADA..... 63
APÊNDICE 1 - ESP-IDF E ESP-WIFI-MESH..... 65
APÊNDICE 2 - OMNET++..... 67

1 INTRODUÇÃO

1.1 CONTEXTO

Avanços tecnológicos na área da Internet das Coisas (IoT) dependem de redes de comunicação eficientes, confiáveis e seguras, com suporte para muitos usuários simultaneamente e capacidade de adaptação. Para atingir estes objetivos são necessárias alternativas aos protocolos usuais de comunicação sem fio. Algumas tecnologias de comunicação sem fio emergentes para aplicações IoT são: Zigbee, *Bluetooth Low Energy*, Thread, entre outras (GROSSI et al., 2023). Porém, protocolos já amplamente utilizados, como o Wi-Fi, podem ser melhores para algumas aplicações IoT. A escolha de um determinado método de comunicação depende de diversos fatores, que frequentemente devem ser analisados em simulação e testes práticos para chegar à uma conclusão concreta.

O Wi-Fi é uma tecnologia de comunicação sem fio baseada no protocolo IEEE 802.11 e amplamente utilizada em ambientes comerciais, industriais e residenciais. A grande disponibilidade do Wi-Fi e o grande número de equipamentos que utilizam esta tecnologia, é um atrativo do ponto de vista da facilidade de implementação e utilização de recursos preexistentes. O Wi-Fi normalmente é utilizado em uma configuração de rede em topologia estrela, mas pode ser configurado para operar em uma topologia Mesh, definida oficialmente pelo protocolo 802.11s (HIERTZ G. et al., 2010), permitindo que os dispositivos da rede se comuniquem entre si e encaminhem mensagens de outros nós. Com isso, o alcance é aumentado e também a resiliência da rede. A topologia *Mesh* é vantajosa para aplicações em IoT em que muitos equipamentos estão envolvidos na rede, pois garante uma resistência à falhas maior e maior adaptabilidade.

Apesar de existir o protocolo 802.11s para redes Wi-Fi *Mesh*, esta tecnologia é pouco utilizada na prática por falta de suporte de dispositivos comerciais à este modo de operação. Com isso em vista, o microcontrolador ESP32 possui uma interface Wi-Fi 802.11 b/g/n e é frequentemente utilizado para aplicações IoT pela sua versatilidade e preço acessível. Além disso, o ESP32 suporta o protocolo ESP-WIFI-MESH, que é uma aproximação do protocolo 802.11s desenvolvido pela empresa Espressif (ESPRESSIF SYSTEMS Co., 2024). Projetos de engenharia reversa estão sendo desenvolvidos para permitir que a camada de dados Wi-Fi seja

acessada pelos desenvolvedores de *Firmware* de forma livre (JASPER, 2025), o que permitirá o desenvolvimento de uma rede 802.11s real nestes dispositivos. O protocolo oficial 802.11s abre diversas possibilidades de aplicação, inclusive a integração do protocolo de aplicação Matter com a rede Wi-Fi *Mesh*.

O padrão Matter está sendo desenvolvido de forma cooperativa pela Connectivity Standards Alliance (CSA-IoT) com o objetivo de facilitar a integração de diversos dispositivos de fabricantes diferentes (PALUMBO, F. et al., 2024). Este é um protocolo da camada de aplicação no modelo OSI e opera em redes Thread, Wi-Fi e Ethernet. O projeto foi iniciado em dezembro de 2019 sob o codinome CHIP, com o objetivo de criar um novo padrão de comunicação IP para dispositivos domiciliares inteligentes, aplicações em celulares e serviços na nuvem. Empresas como Amazon, Apple, Google e a aliança Zigbee estão participando do desenvolvimento do Matter. O protocolo ainda está em fase de desenvolvimento e, por isso, sua presença nos equipamentos de IoT ainda é baixa.

O presente trabalho visa analisar a eficiência de uma rede ESP-WIFI-MESH com simulações computacionais e testes práticos em dispositivos ESP32, identificando os pontos fortes e fracos deste método de comunicação além de fazer uma análise comparativa entre os resultados.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Simular, implementar e analisar de uma rede de comunicação sem fio Wi-Fi *Mesh*, tendo em vista aplicações na internet das coisas.

1.2.2 Objetivos Específicos

- Desenvolver uma rede Wi-Fi *Mesh* ou análoga em ambiente virtual;
- Simular e analisar os resultados da rede desenvolvida;
- Implementar a rede *Mesh* em um cenário real utilizando microcontroladores;
- Realizar testes práticos de eficiência da rede;
- Analisar os resultados de forma comparativa aos obtidos em simulação.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão apresentados os conceitos pertinentes para o desenvolvimento do trabalho, informações sobre o estado da arte da tecnologia utilizada e referências de trabalhos realizados em torno do tema abordado.

2.1 O PADRÃO MATTER

O padrão de conectividade Matter é desenvolvido pela Aliança de Padrões de Conectividade, do inglês, *Connectivity Standards Alliance* (CSA-IoT), e visa a unificação dos dispositivos inteligentes da Internet das Coisas, ou *Internet of Things* (IoT), ao aumentar a compatibilidade destes em ecossistemas variados. Empresas como Apple, Google, Amazon e a aliança Zigbee participam do desenvolvimento do padrão Matter, que simplifica o desenvolvimento de dispositivos IoT ao providenciar uma abordagem unificada para a conectividade. Ele oferece uma forma segura, resiliente e fácil para que diversos equipamentos se comuniquem e interajam independentemente dos fabricantes (BELLI, BARSOCCHI, PALUMBO, 2024).

Lançado em 2019, o protocolo Matter é um *software* de distribuição livre para automação residencial e utiliza como base o protocolo internet de comunicação, ou *Internet Protocol* (IP). O ponto forte do padrão Matter é sua interoperabilidade. Por exemplo, estabelecer uma comunicação efetiva entre um equipamento de som ativado por um assistente por voz, como o *Google Assistant*, e ao mesmo tempo, integrar este equipamento ao ecossistema do aplicativo Amazon Alexa. A implementação do padrão é facilitada pelo protocolo IP, que é amplamente utilizado.

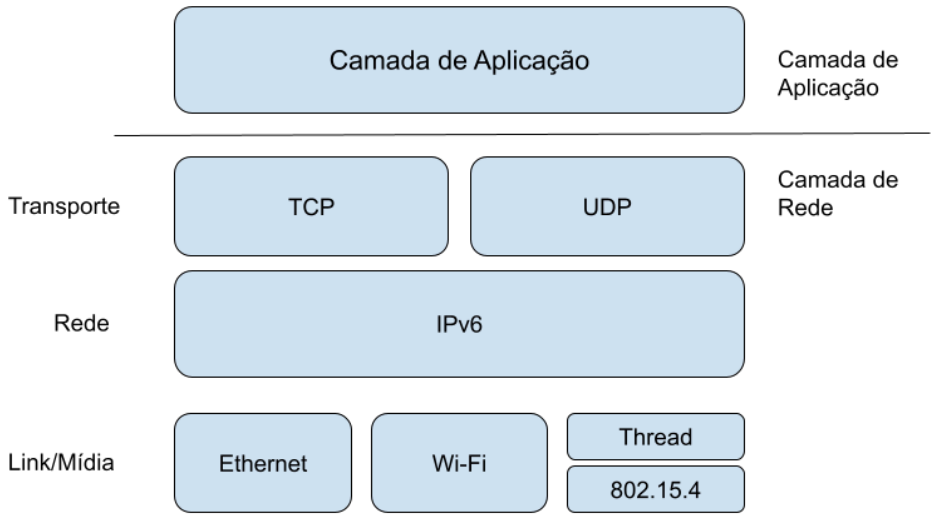
Além da sua ênfase em interoperabilidade, o Matter também prioriza a segurança e a privacidade da comunicação. Dispositivos certificados como compatíveis ao padrão, precisam corresponder a regras estritas de segurança, incluindo criptografia ponta a ponta e administração segura de chaves. Isso garante que usuários possam confiar que seus dispositivos e dados estão seguros de ações maliciosas. A norma Matter, sendo implementada na camada de aplicação do modelo *Open Systems Connection* (OSI), é desenvolvida para suportar comunicação sobre uma variedade de protocolos da camada de rede, incluindo Wi-Fi, Thread/802.15.4 e Ethernet. Permitindo desse modo uma grande flexibilidade para

os usuários escolherem a forma que eles desejam conectar seus dispositivos (SCHENK M., 2023).

2.1.1 Arquitetura de Rede

O objetivo do padrão Matter é desenvolver um protocolo de comunicação compreensivo para dispositivos Smart Home, utilizando IPv6 como base para isso. O protocolo abrange a camada de aplicação, definindo sua implementação em dispositivos na forma de Clusters, assim como em múltiplas camadas de conexão para garantir interoperabilidade. A arquitetura do protocolo é organizada em camadas distintas, oferecendo uma separação clara de responsabilidades e encapsulamento efetivo da pilha do protocolo, como é mostrado na Figura 1. O Matter em si é implementado na camada de aplicação, onde ele providencia um framework comum para desenvolvedores criarem aplicações que podem controlar e interagir com dispositivos inteligentes independente do fabricante ou da tecnologia. O Matter utiliza o protocolo de *Transmission Control Protocol* (TCP) ou o *User Datagram Protocol* (UDP) para a transmissão de mensagens.

Figura 1: Pilha do Protocolo Matter.



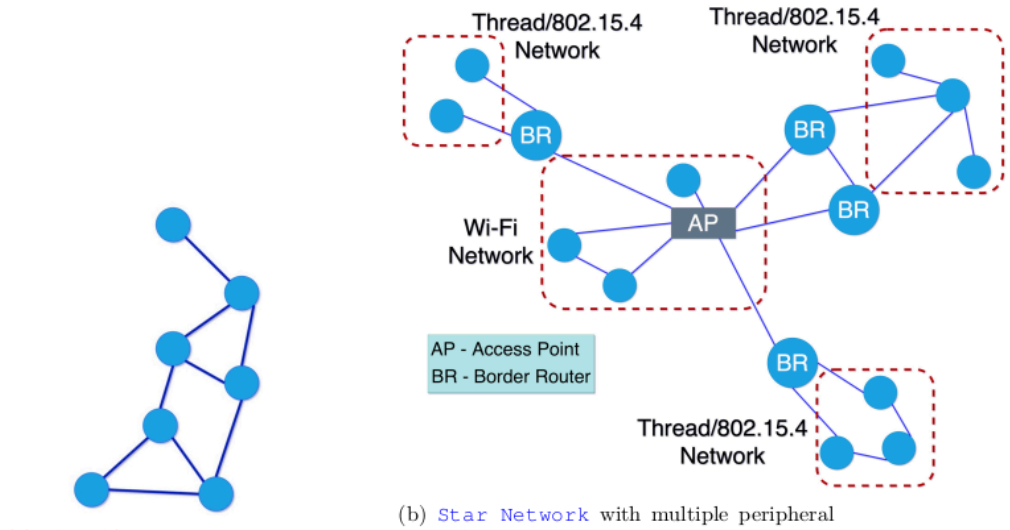
Fonte: Adaptado de Matter GitHub (2025).

O Matter estabelece uma linguagem unificada que permite que os dispositivos se entendam e respondam a comandos e requisições de aplicações. IPv6 é usado na camada de rede para providenciar uma comunicação universal para cenários Smart Home. Como o Matter suporta vários modos de conectividade (Wi-Fi, Ethernet ou Thread), diferentes topologias são também suportadas (SCHENK M., 2023).

2.1.2 Topologia de Rede

A especificação foca em três tipos de tecnologias na camada de acesso: Wi-Fi, Ethernet e Thread. O Matter adota uma abordagem de recursos compartilhados, permitindo que múltiplas redes Matter sejam estabelecidas em um mesmo universo de redes IP. O protocolo também suporta comunicação entre uma ou mais redes IPv6 através do uso de Roteadores de Borda. A especificação determina dois tipos de topologia suportadas pelo Matter: Rede Única e Rede Estrela, mostradas na Figura 2. A Rede Única é constituída por nós que formam juntos esta rede unificada, podendo assumir diversas topologias, incluindo *Mesh*. Portanto, o termo “Única” significa simplesmente que a topologia não inclui mais de uma rede independente.

Figura 2: Topologias suportadas pelo Matter.



(a) Thread/802.15.4 Single Network (b) Star Network with multiple peripheral networks

Fonte: Schenk M. (2023).

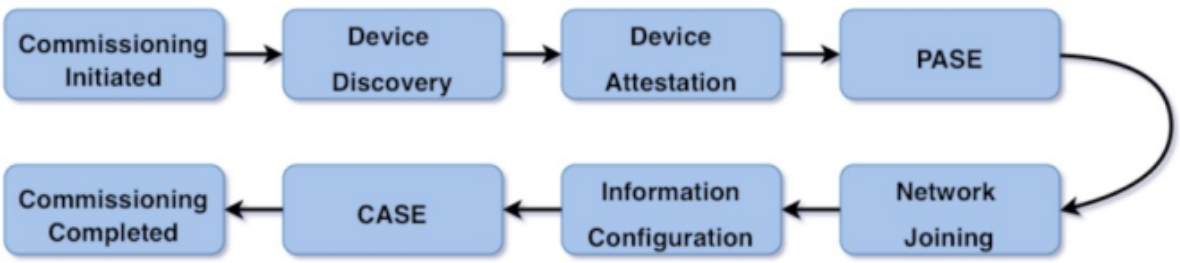
Já a Rede Estrela pode incluir diversas redes independentes entre si. Porém, estas redes independentes são conectadas a uma rede central, configurando uma topologia em estrela. As redes periféricas são conectadas à rede central através dos Roteadores de Borda, responsáveis por integrar as redes de forma eficiente. As redes periféricas podem ser constituídas por qualquer um dos protocolos de acesso já mencionados.

2.1.3 Comissionamento

O comissionamento se trata das etapas que são executadas para integrar e configurar um dispositivo na rede Matter. As etapas são representadas de forma concisa na Figura 3. Este processo é iniciado com o descobrimento de um novo dispositivo, onde o dispositivo é identificado através de um dos protocolos de comunicação suportados pelo padrão. Depois de identificado, o dispositivo é conectado a uma rede IPv6 usando uma das tecnologias de camada de acesso.

Com a conexão estabelecida, o processo continua com a configuração de segurança usando PASE (Password-Authenticated Session Establishment), ou Estabelecimento de Sessão com Autenticação por Senha. A autenticação do dispositivo é realizada e as permissões necessárias são concedidas. A configuração de informação é então realizada, atribuindo identificadores, endereços de rede e determinando credenciais de segurança. A formação da rede é determinada a partir da topologia desejada pelo usuário, podendo ser Única ou em Estrela. No próximo passo é realizada mais uma configuração de segurança com o protocolo CASE (Certificate Authenticated Session Establishment), ou Estabelecimento de Sessão com Autenticação por Certificado.

Figura 3: Etapas do comissionamento no Matter.



Fonte: Schenk M. (2023).

Quando todas as etapas do comissionamento mencionadas estiverem finalizadas sem a ocorrência de erros, o processo pode ser concluído. A conclusão é realizada com a troca de uma mensagem criptografada com com uma chave derivada do PASE, indicando sucesso no comissionamento.

Este processo de comissionamento garante que os dispositivos Matter sejam integrados com sucesso e estejam operacionais dentro do esperado para a *Smart Home*, oferecendo conexão segura e eficiente.

2.2 REDES WI-FI MESH

O protocolo de comunicação sem fio Wi-Fi é extremamente difundido, estando presente em ambientes industriais, em escritórios, em residências, aeroportos, veículos de transporte de pessoas, etc. Entre as aplicações variadas deste protocolo, está a comunicação entre dispositivos IoT e automação residencial. Sendo este um dos focos do padrão Matter.

2.2.1 *Wireless Mesh Networks* (WMN)

Redes em malha sem fio, do inglês, *Wireless Mesh Networks* (WMN), têm o potencial de entregar acesso de banda larga à internet e maior cobertura para WLAN com nós estacionários ou móveis a um baixo custo. Pois as redes WLAN comuns, dependem de infraestrutura cabeada para seu funcionamento e os APs (*Access Points*) normalmente apresentam uma cobertura limitada devido a regras do uso do espectro de rádio que limitam a potência de transmissão dos equipamentos (MICHAIL L. SICHITIU, 2012). Portanto, um usuário que deseje aumentar a cobertura de seu Wi-Fi, precisa instalar um novo AP com o devido cabeamento.

Uma das maiores vantagens da WMN é que a infraestrutura da rede são os próprios dispositivos que a utilizam. Isso é alcançado ao conceder aos nós a tarefa de encaminhamento de pacotes em uma configuração de múltiplos saltos (*multi-hop*), formando assim uma rede ad hoc (MANET). Outras vantagens da WMN são:

- Implementação rápida com infraestrutura barata;
- Fácil de providenciar cobertura em locais sem cabeamento;

- Auto regenerativa, resiliente e expansível;
- Maior alcance;
- Maior largura de banda devido a saltos mais curtos;
- Maior durabilidade da bateria devido a potências de transmissão mais baixas.

A adaptabilidade e versatilidade das redes sem fio *Mesh*, conferem a esta modalidade uma gama de aplicações diversas. Entre os cenários de uso estão:

- Uso Residencial
    - Neste caso a WMN seria utilizada para criar uma rede de baixo custo, facilmente implementável e com cobertura de alta performance ao longo da casa. Uma alta taxa de transmissão também é desejável, já que aplicações de áudio e vídeo podem exigir grandes bandas.
- Segurança Pública
    - Redes WMN para segurança pública providenciam bombeiros, policiais e paramédicos com acesso à rede. A rede pode ser usada para monitoramento, localização de emergências e dos funcionários com comunicação de voz, dados, imagens, etc.
- Militar
    - Neste caso é importante uma alta adaptabilidade e mobilidade da rede, além de uma administração automática eficiente. A rede WMN atende estes requisitos e também possui auto regeneração, permitindo comunicação mesmo com falhas de alguns nós da rede.

Apesar das redes WMN apresentarem vários pontos fortes, algumas de suas qualidades têm um preço. Entre algumas desvantagens da utilização de redes sem fio em malha, podem ser citadas:

- Sobrecarregamento do espectro com reencaminhamentos de pacotes;
- Maior complexidade do sistema;
- Maior latência devido a múltiplos saltos;
- A segurança da comunicação pode ser prejudicada com maior facilidade pela falta de um controle central.

2.2.2 IEEE 802.11

IEEE 802.11 é o padrão de comunicação sem fio para redes locais, criado pelo Instituto de Engenheiros Eletricistas e Eletrônicos, do inglês, *Institute of Electrical and Electronics Engineers*, que governa as funcionalidades da tecnologia Wi-Fi. A tecnologia foi lançada ao mercado no ano de 1997 com uma taxa máxima de transmissão de dados de 2 Mbps e com uma portadora na frequência de 2,4 GHz. Logo a tecnologia se popularizou e foi adotada por inúmeros fabricantes, entre eles a empresa Apple.

O padrão IEEE 802.11 define uma série de especificações da camada física (PHY) e camada de acesso ao meio (MAC) para implementação de redes sem fio locais (WLAN). A primeira versão do Wi-Fi permitia o uso de três tipos de modulação: infravermelho difuso, espalhamento de espectro por saltos em frequência (FHSS) e espalhamento de espectro por sequência direta (DSSS). O último acabou por se tornar o modo de modulação mais popular com o advento do IEEE 802.11b em 1999, que também usa um esquema DSSS e aumentou a taxa de transmissão para 11 Mbps (TEKTRONIX, 2016).

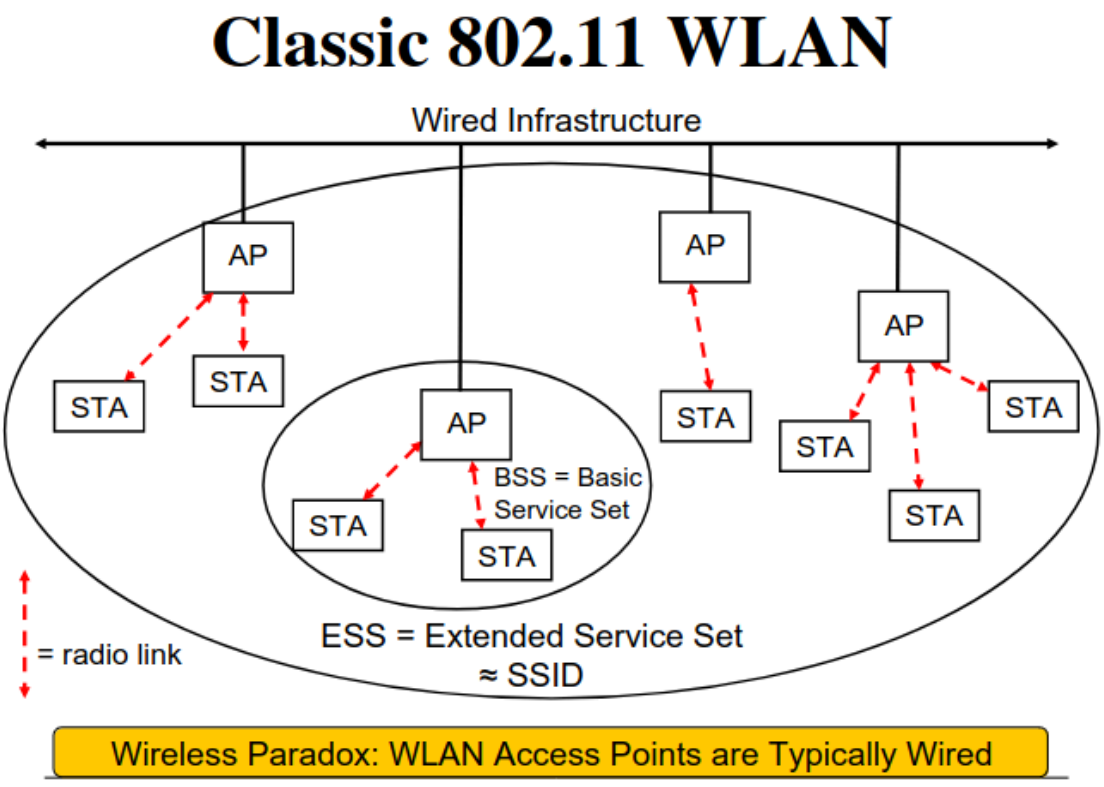
A estrutura básica de rede de um padrão 802.11 é constituída por: estações, do inglês *Stations* (STA's) e pontos de acesso, ou do inglês, *access points*. Os pontos de acesso providenciam as estações com o serviço de integração com outras STA's e com possíveis redes externas fora do escopo do 802.11. Normalmente o AP é caracterizado como infraestrutura, já que é conectado por rede cabeada a dispositivos Ethernet (GUIDO R. HIERTZ et al., 2010). Esta configuração clássica de uma rede Wi-Fi é centralizada e com topologia do tipo estrela, sendo também a configuração mais difundida de redes WLAN. A Figura 4 mostra uma representação deste tipo de rede.

Em 1999 também foi lançado o padrão IEEE 802.11a, que utiliza uma frequência de portadora de 5 GHz e um esquema de modulação de multiplexação por divisão ortogonal em frequência, ou *Orthogonal Frequency-Division Multiplexing* (OFDM), atingindo uma máxima taxa de transmissão de 54 Mbps. Além disso, a faixa de frequência mais alta do padrão 802.11a, permite que este utilize bandas maiores de canal e sofra menos com interferências. Uma de suas desvantagens é o desvanecimento maior sofrido pelo sinal e perdas maiores por reflexão e absorção

em obstáculos. A alta frequência também permite que as dimensões das antenas sejam menores e produzam maiores ganhos (TEKTRONIX, 2016).

Em janeiro de 2003 o protocolo 802.11g foi rapidamente adotado pelos consumidores devido a demanda por maiores taxas de transmissão e redução dos custos de fabricação. Este protocolo utiliza a banda de 2,4 GHz, assim como o 802.11b, e o esquema de modulação OFDM, como o 802.11a. Nessa configuração o padrão pode atingir 54 Mbps e também é compatível com equipamentos 802.11b. Apesar de suas vantagens, a adoção em massa desse protocolo e o uso da faixa de 2,4 GHz tornam a comunicação sujeita a interferências de outras redes na mesma banda. Mesmo configurando redes próximas em canais diferentes, bandas laterais podem causar interferência e prejudicar a comunicação.

Figura 4: Rede Wi-Fi na configuração usual.



Fonte: W. Steven Conner et al. (2006).

A versão 802.11n foi lançada em 2009 e inclui diversas melhorias que aumentam o alcance da rede, resiliência e a taxa de transmissão. Técnicas avançadas de processamento digital de sinais foram implementadas nas camadas física e MAC que permitem taxas de até 600 Mbps. Esta versão opera tanto em 2,4

GHz como em 5 GHz com canais de 40 MHz ou 20 MHz de largura de banda e com múltiplas entradas e saídas, do inglês *Multiple Inputs Multiple Outputs* (MIMO). A técnica MIMO permite transmissão e recepção simultâneas, além de multiplexação por divisão espacial.

Em 2013 foi lançada a versão 802.11ac com foco em aplicações de alta densidade de dados e desenvolvendo aspectos da versão anterior. O 802.11ac utiliza bandas de até 160 MHz, até 8 fluxos espaciais MIMO e modulação de alta densidade (até 256-QAM).

A versão mais recente do protocolo é o 802.11ax. Nesta versão é utilizada a modulação *Orthogonal Frequency Division Multiple Access* (OFDMA) ao invés da modulação OFDM. Com esta técnica, os canais de comunicação são divididos em subcanais, permitindo sua alocação para diversos dispositivos simultaneamente. A modulação pode chegar até 1024-QAM e também foi adicionada uma nova faixa de frequências em 6 GHz.

2.2.3 O Padrão WLAN 802.11s para Redes *Mesh*

Em 2010 a proliferação da conexão *wireless*, especialmente o Wi-Fi, e a maior demanda por redes com grandes áreas de cobertura fomentaram o desenvolvimento do protocolo 802.11s para redes *Mesh*. Sendo parte da família de protocolos 802, este padrão suporta interoperabilidade com os outros padrões citados. O 802.11s garante entrega transparente de mensagens *uni, multi e broadcast* para destinos dentro e fora da rede *Mesh* ao adicionar estes serviços na camada MAC do padrão original 802.11. Os dispositivos que integram a rede são chamados de estações *Mesh* (*Mesh STA's*), ou simplesmente nós *ad-hoc*, e fazem o redirecionamento de pacotes de forma *wireless*, mas apenas para outras estações *Mesh* (GUIDO R. HIERTZ et al., 2010). Um aspecto chave da arquitetura *Mesh*, é a presença de conexões sem fio *multi-hop*, ou seja, a conexão entre dois nós na rede pode ser realizada pelo intermédio de outros nós, que por sua vez realizam o roteamento da mensagem através de vários "saltos", até que ela atinja seu destino.

Para que o encaminhamento de mensagens em uma rede mesh seja realizado, é necessário o uso de um protocolo de roteamento. O padrão 802.11s possui um protocolo de roteamento padrão, denominado *Hybrid Wireless Mesh*

Protocol (HWMP), mas o usuário pode escolher usar qualquer outro protocolo de roteamento *ad-hoc*.

As estações *Mesh* podem também coexistir com APs do padrão 802.11, permitindo assim a comunicação entre elementos da rede mesh com STAs comuns. Um portal 802.11 pode também ser implementado na rede atuando como um *gateway*, permitindo o acesso a uma ou mais redes 802.11.

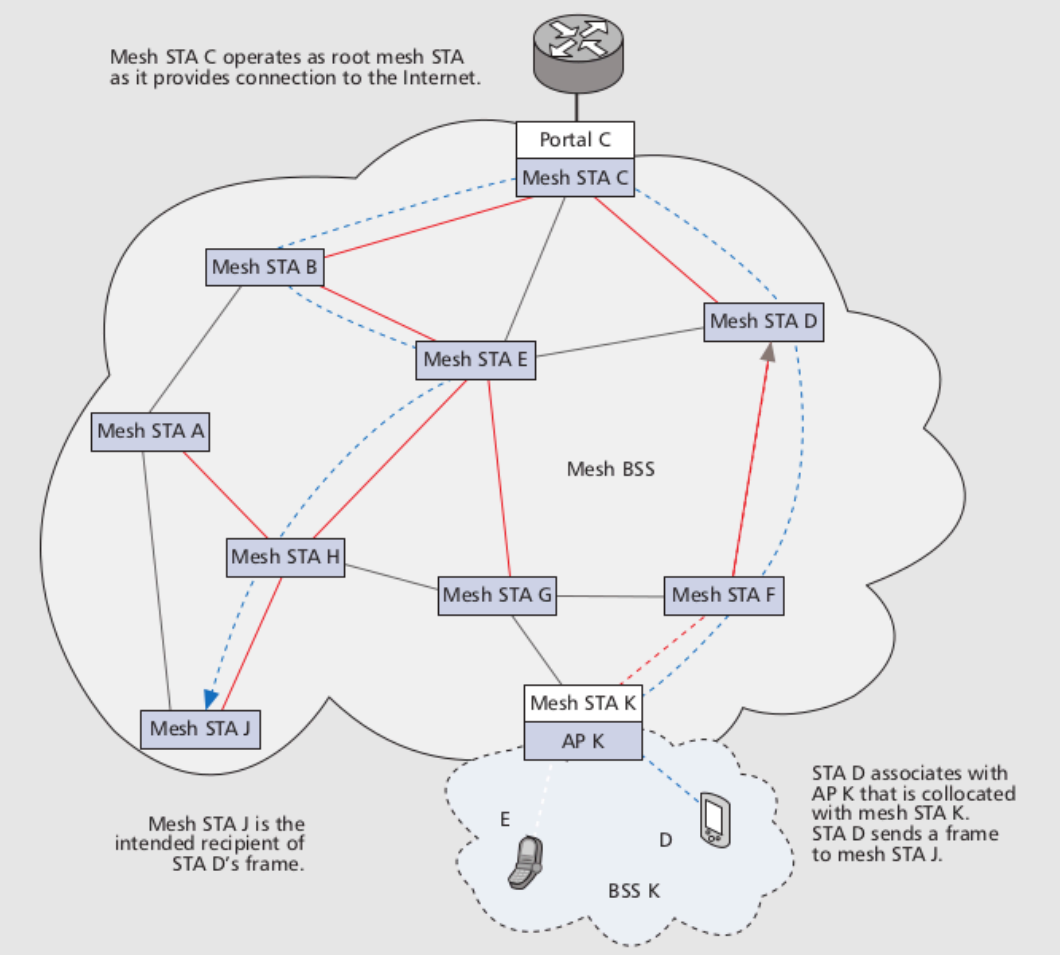
A Figura 5 mostra uma rede Wi-Fi *Mesh*. Na estação STA C é implementado um portal em uma das interfaces do dispositivo, conectando a rede a um roteador. Pelo fato da estação *Mesh* STA C estar conectada a um roteador, provendo a rede mesh com acesso à internet, ela é denominada como nó raiz, ou *root node.* Na estação mesh STA K, uma das interfaces é configurada como AP, conectando a uma outra rede típica no padrão 802.11. Com isso, quadros da rede 802.11 podem ser enviados para a rede *Mesh* através da conexão com a estação STA K. O envio de um pacote de STA D para *Mesh* STA J é representado na imagem com uma linha tracejada. Observa-se que o pacote passa por vários saltos, ao ser encaminhado de um nó para outro até chegar ao destino final.

O padrão 802.11s foi desenvolvido para suportar redes *Mesh* de pequeno a médio porte, aproximadamente 32 nós. Este é um fator limitador das redes Wi-Fi mesh importante em relação a outros protocolos de comunicação voltados para IoT. O descobrimento de rotas é realizado de forma dinâmica e considerando as capacidades e características de rádio do ambiente. A segurança da rede é realizada com o padrão 802.11u, responsável pela autenticação de dispositivos sem conexão prévia.

A formação da rede e, portanto, da topologia *Mesh* é realizada de forma automática pelos dispositivos utilizando um *Beacon* que carrega informações da rede. Cada estação *Mesh* detecta uma outra a partir de uma varredura passiva (observação de quadros beacons) ou ativa (transmissão de quadro de teste). Os quadros beacon possuem uma identificação da rede (*Mesh ID*), um elemento de configuração que indica os serviços da rede e parâmetros suportados pela *Mesh* STA transmissora. Essa funcionalidade permite que estações mesh procurem por parceiros consistentes. Quando tal parceiro é identificado, a estação *Mesh* utiliza o protocolo de gerenciamento de link mesh, ou *Mesh Peer Link Management,* para estabelecer uma conexão com outra estação *Mesh* (GUIDO R. HIERTZ et al., 2010).

A segurança na rede 802.11s é realizada através da autenticação simultânea de iguais, ou *Simultaneous Authentication of Equals* (SAE). Além de autenticação mútua, o SAE fornece às duas estações *Mesh* uma chave mestre de pares, *Pairwise Master Key* (PMK), que elas usam para criptografar seus pacotes. Com este sistema, cada par de estações é independentemente protegido e, como consequência, o 802.11s não oferece criptografia de ponta a ponta (GUIDO R. HIERTZ et al., 2010).

Figura 5: Rede Wi-Fi *Mesh* 802.11s.



Fonte: Guido R. Hiertz et al. (2010).

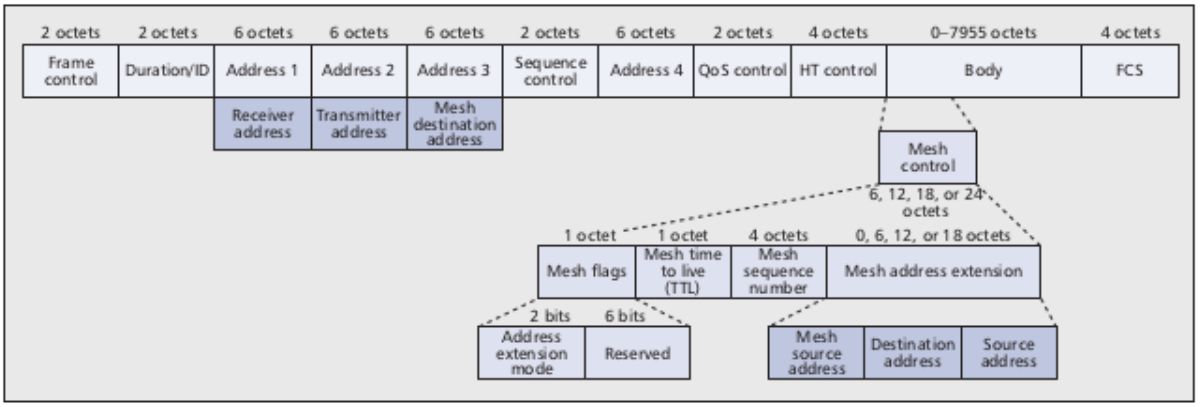
O padrão 802.11s também implementa uma versão modificada do controle de acesso ao meio (camada MAC) para evitar o congestionamento de mensagens. Isso se torna necessário pois a rede mesh é formada por vários links de capacidades diferentes. Além disso, links formados entre nós de hierarquia mais alta tendem a ser mais congestionados do que pontos extremos da rede. No protocolo MAC da

rede padrão 802.11, os nós enviam o máximo de pacotes possíveis sem considerar quantos chegam ao destino (W. Steven Conner et al., 2006). Por estes motivos, foram adicionados ao protocolo MAC original:

- Monitoramento Local de congestionamento
    - Cada nó monitora a utilização do canal local;
    - Se congestionamento for detectado, notifica nós vizinhos e/ou nós anteriores ao salto.
- Sinalização de congestionamento
    - Requisição de controle de congestionamento (unicast);
    - Resposta de controle de congestionamento (unicast);
    - Anúncio de congestionamento na vizinhança (broadcast).
- Controle de taxa local
    - Cada nó que receber uma notificação de controle de congestionamento ajusta sua geração de tráfego de acordo;

O quadro transmitido pela rede padrão 802.11s é mostrado na Figura 6. Uma particularidade deste quadro é a presença de um endereço de transmissor. Este é o endereço do último nó a encaminhar o pacote e muda para cada salto realizado.

Figura 6: Quadro da rede 802.11s.

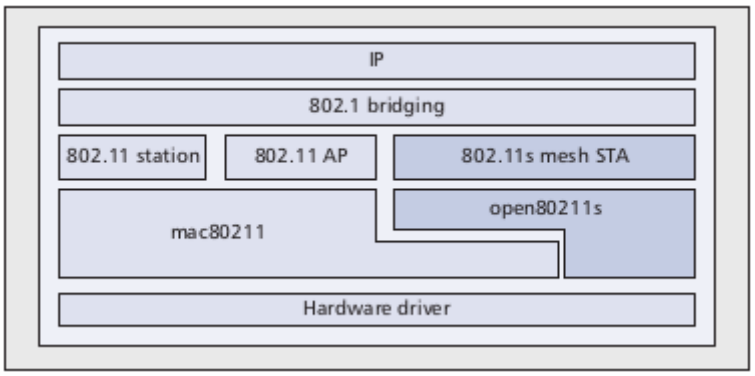


Fonte: Guido R. Hiertz et al. (2010).

O protocolo 802.11s foi implementado com sucesso no núcleo do sistema operacional Linux desde a versão 2.6.26. O projeto chamado *open80211s* possui a

pilha de software mostrada na Figura 7. Ressalta-se que o bloco IP está acima dos blocos que implementam o padrão 802.11s.

Figura 7: Pilha de *software open802.11s*.



Fonte: Guido R. Hiertz et al. (2010).

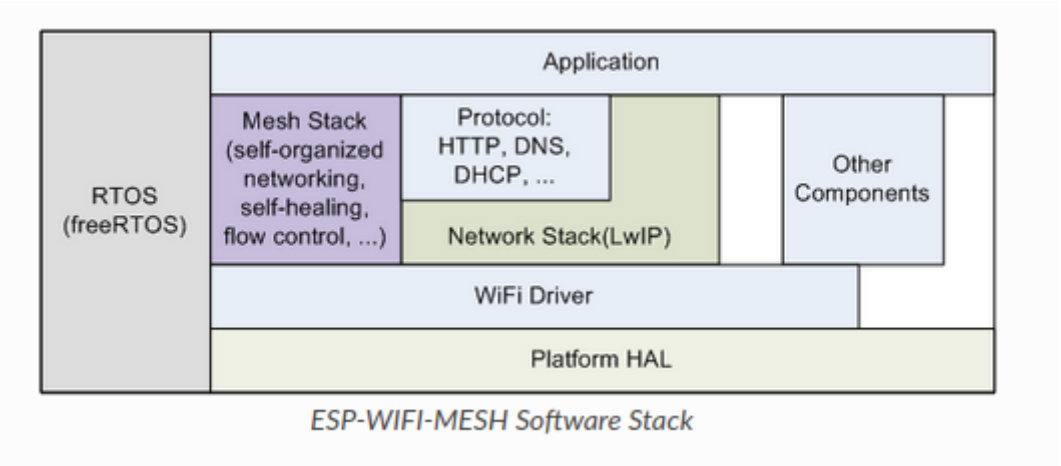
2.3 ESP-WIFI-MESH

O ESP-WIFI-MESH é uma aproximação de um protocolo 802.11s para os microcontroladores da família de produtos ESP fabricados pela empresa Espressif. Este protocolo possui, assim como o 802.11s, mecanismos de auto regeneração e auto organização. Porém, ele não é uma implementação completa do protocolo 802.11s, pois não realiza todas as funcionalidades impostas pelo padrão. O ESP-WIFI-MESH foi a rede escolhida para o projeto pois é a única forma de rede WiFi Mesh para microcontroladores ESP32 disponível no período de desenvolvimento deste trabalho. Existem pesquisas sendo realizadas no ESP32 para desbloqueio completo das configurações de Wi-Fi e de controle de dados do microcontrolador (JASPER, 2025). Este projeto tem como objetivo possibilitar aos usuários acesso irrestrito às funções da camada de acesso ao meio, o que possibilitaria a criação de um protocolo 802.11s real para os dispositivos. Atualmente, o acesso ao meio é realizado apenas através de funções pré definidas pela empresa Espressif, o que restringe as possibilidades de desenvolvimento.

O software foi desenvolvido com a pilha mostrada na Figura 8. Observa-se que o programa que caracteriza a rede *Mesh* é abreviado na literatura como protocolo de camada 2.5. Isso significa que é um programa que mescla funcionalidades da camada MAC com a camada de rede (protocolo IP), que estão

localizados no mesmo nível da pilha. Nota-se que a pilha do projeto *open80211s* (Figura 7) difere neste aspecto.

Figura 8: Pilha de *software* do ESP-WIFI-MESH.



Fonte: Espressif Systems Co. (2024).

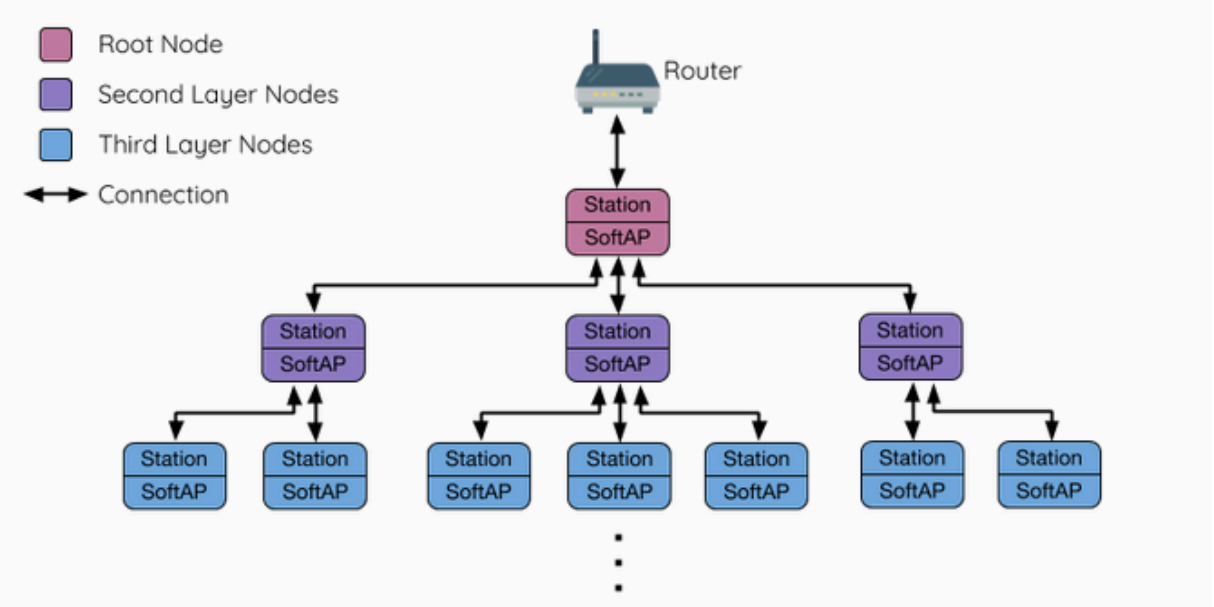
A rede ESP-WIFI-MESH é constituída por três blocos principais: o bloco *Mesh*, o bloco WiFi e o bloco LwIP. O bloco mais básico é o WiFi, que conecta o ESP32 aos drivers e ao hardware de rádio frequência. O bloco LwIP é o responsável por atribuir endereços IPv4 às interfaces do sistema. A funcionalidade do bloco LwIP só é necessária no nó central da rede, pois este é o único nó que tem acesso direto à rede externa através do roteador. Portanto, quando este nó recebe uma mensagem da rede Mesh interna destinada à rede externa, é realizada a atribuição de um endereço IP ao pacote e este é redirecionado ao roteador. Para pacotes que são recebidos da rede externa, é utilizado o protocolo NAT para redirecionar o pacote para o nó da rede interna de forma correta.

No ESP-WIFI-MESH, a topologia utilizada é em árvore, que é um tipo de rede em malha. A rede em malha completa (*full mesh*) é caracterizada por conexões diretas entre cada nó da rede. Já na topologia em árvore, há um nó raíz (*root node*) a partir do qual os outros nós se conectam em ramos. Esta topologia é realizada em camadas, onde os nós da primeira camada são conectados ao nó raiz, os nós da segunda camada são conectados aos nós da primeira camada e assim por diante. Outra característica da rede ESP-WIFI-MESH que faz a topologia ser classificada como árvore, é que cada nó é limitado a realizar apenas uma conexão com a

camada acima (*upstream*), mas pode realizar múltiplas conexões para camadas abaixo (*downstream*).

Para realizar as conexões, são usados dois tipos diferentes de interface Wi-Fi, sendo eles: *SoftAP* e *Station*. O primeiro modo é uma abreviação de *Software Enabled Access Point*, que é um ponto de acesso Wi-Fi definido por *software*. O segundo modo é o de estação mesh, que realiza uma conexão com um *SoftAP*. A Figura 9 mostra uma ilustração da rede ESP-WIFI-MESH.

Figura 9: Topologia em árvore da rede ESP-WIFI-MESH.



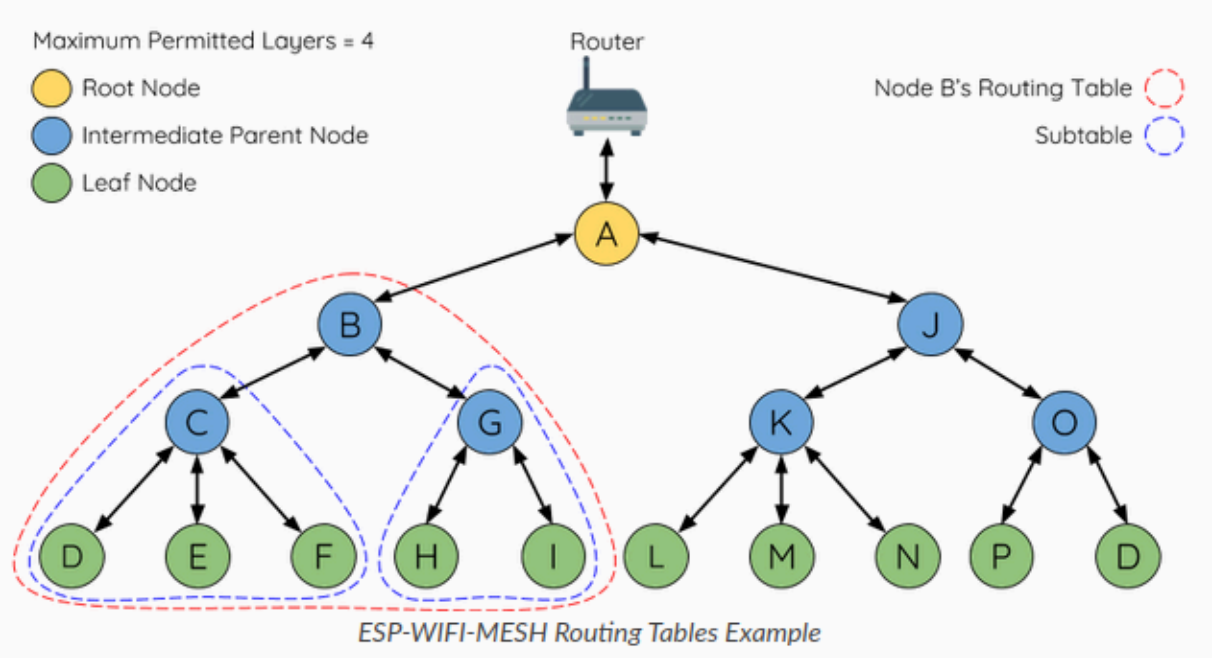
Fonte: Espressif Systems Co. (2024).

O ESP-WIFI-MESH realiza o roteamento de mensagens dentro da rede utilizando endereços MAC. Cada nó da rede mantém uma tabela de roteamento com os endereços MAC dos nós filhos. A Figura 10 mostra como os endereços da rede são armazenados para cada camada de conexão. Quando um nó recebe um pacote e o endereço de destino não está em sua tabela de roteamento, o pacote é enviado ao nó pai, que irá comparar o endereço de destino com sua tabela de roteamento. O nó central possui em sua tabela de roteamento todos os endereços MAC da rede Mesh.

Apesar da rede não utilizar endereços IP como forma de roteamento, endereços IP podem ser atribuídos pela camada LwIP às interfaces de cada um dos nós da rede. A atribuição dos endereços IP aos nós da rede *Mesh*, permite que seja

realizada comunicação efetiva entre os nós utilizando protocolos de transporte como o TCP e o UDP. Porém, o envio e encaminhamento de pacotes TCP/UDP são realizados diretamente pela camada ESP-MESH, sem passar pela camada LwIP.

Figura 10: Exemplo de tabelas de roteamento ESP-WIFI-MESH.

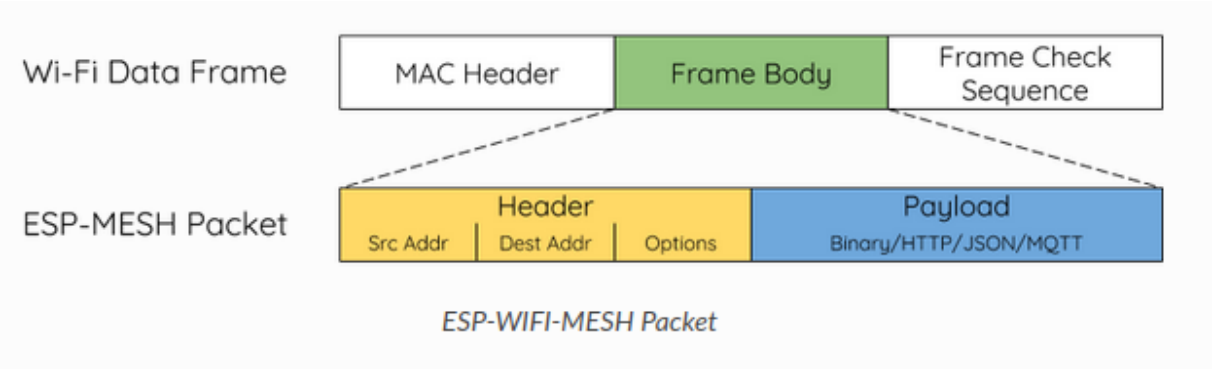


Fonte: Espressif Systems Co. (2024).

O pacote ESP-WIFI-MESH está completamente contido dentro do corpo do quadro Wi-Fi. O pacote é dividido em duas partes: cabeçalho (*header*) e carga (*payload*). No cabeçalho é armazenado o endereço MAC da fonte e o endereço MAC do destino, enquanto na carga são armazenadas informações da aplicação. No caso do pacote ser destinado para uma rede externa, o endereço de destino do cabeçalho será um endereço IP da rede externa. O nó central se encarrega de formar um pacote TCP/IP ou UDP/IP com as informações de pacotes destinados à rede externa. A Figura 11 mostra o quadro utilizado na rede ESP-WIFI-MESH.

O envio de informação bidirecional na rede ESP-WIFI-MESH é representado de forma visual na Figura 12. Observa-se que a informação parte da camada de aplicação de um nó folha na rede e é processado diretamente pela camada ESP-MESH.

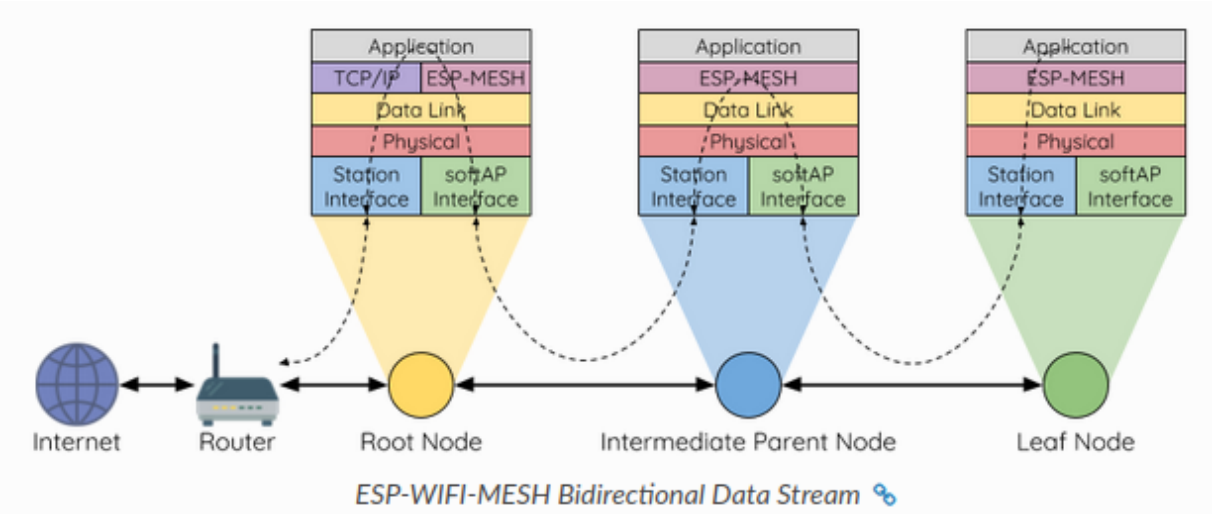
Figura 11: Pacote da rede ESP-WIFI-MESH.



Fonte: Espressif Systems Co. (2024).

O programa se encarrega de enviar a informação, que passa pela camada de *link* de dados, pela camada física e finalmente pela interface STA. O pacote é recebido pelo nó pai, que processa o quadro até atingir a camada ESP-MESH onde é realizado o roteamento baseado no endereço MAC presente no cabeçalho do pacote ESP-MESH. Ao chegar no nó raíz, o pacote com destino à rede externa é processado até a camada de aplicação. A informação é então encapsulada utilizando TCP/IP e enviada ao roteador. Portanto, o encapsulamento padrão IP só é realmente realizado pelo nó central e somente para pacotes destinados à rede externa.

Figura 12: Exemplo de fluxo de dados bidirecional na rede ESP-WIFI-MESH.



Fonte: Espressif Systems Co. (2024).

No exemplo ‘*ip_internal_network*’, disponibilizado no ESP-IDF, os endereços IP não são utilizados para o roteamento. O processo que ocorre é um simples mapeamento do endereço IP para um endereço MAC. O endereço MAC é inserido no cabeçalho do quadro ESP-MESH e é utilizado efetivamente para o roteamento entre os nós da rede.

O protocolo Matter é implementado na camada de aplicação e exige uma rede IPv6 padrão para operar corretamente. Com isso em vista, a implementação do Matter na rede ESP-WIFI-MESH não é possível sem modificações no modo como a rede faz o roteamento, que necessitaria de um roteamento IP padrão assim como na rede Wi-Fi comum ou como no padrão 802.11s.

3. MATERIAIS

Para o desenvolvimento do projeto foram utilizados programas e simuladores além de dispositivos físicos como computadores. Desse modo, os recursos utilizados podem ser divididos em *Software* (recursos digitais/virtuais) e *Hardware* (recursos físicos).

3.1 *SOFTWARE*

O trabalho utilizou recursos de *Software* para realizar a etapa de simulações e de programação dos microcontroladores.

O programa OMNeT++ foi utilizado para a simulação da rede por ser de uso acadêmico livre e possuir uma documentação extensa. OMNeT++ é um simulador modular, orientado a objetos e baseado na linguagem C++ (OWCZAREK, P.; ZWIERZYKOWSKI, P., 2014). Os projetos no simulador são organizados em módulos hierárquicos, sendo utilizada a linguagem NED para a criação de sistemas complexos com vários módulos. O programa pode ser utilizado nos sistemas Windows, Linux e Mac-os (OPENSIM LTD., 2025).

Foi utilizada a linguagem de programação Python em conjunto com os *frameworks*: Pandas, Matplotlib, NumPy e Json, para a análise dos resultados da simulação. O OMNeT++ permite que as métricas obtidas pela simulação sejam extraídas para arquivos json, que facilita a utilização de programas customizados de análise.

Para a programação dos testes práticos, foi utilizado o *framework* oficial ESP-IDF da Espressif. O ESP-IDF foi escolhido por permitir ao usuário acesso à todas as ferramentas e bibliotecas de desenvolvimento da Espressif. O Arduino IDE foi considerado como alternativa para o desenvolvimento do projeto, por possuir vários recursos para o microcontrolador ESP32 e utilizar uma linguagem mais acessível. Porém, nem todas as ferramentas disponíveis no ESP-IDF podem ser utilizadas no Arduino IDE, o que dificulta o processo de desenvolvimento de programas mais complexos. A linguagem de programação C é utilizada no ESP-IDF. Além disso, vários exemplos são fornecidos pelo ambiente e uma documentação

extensa das *API's* (*Application Programming Interface*) podem ser encontradas no site da Espressif.

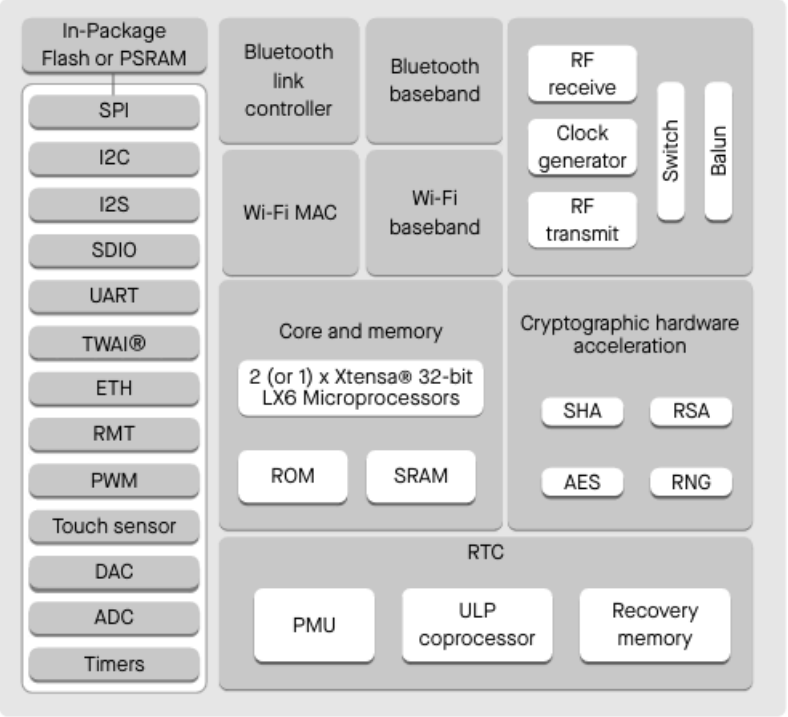
3.2 *HARDWARE*

Um computador com sistema operacional Windows 10 e processador Intel i5 de 5ª geração foi utilizado para executar as simulações e realizar o processamento dos dados.

Um *notebook* com sistema operacional Ubuntu 18.04 foi utilizado para a programação, compilação e envio dos arquivos binários para os microcontroladores.

Os microcontroladores utilizados no projeto foram três ESP32 fabricados pela Espressif, sobre placas PCB do tipo DevKit v1. O ESP32 é um único chip com Wi-Fi 2,4 GHz e Bluetooth integrados, fabricado em tecnologia TSMC de baixa potência de 40 nm. A Figura 13 mostra o bloco funcional do ESP32.

Figura 13: Bloco funcional do chip ESP32.

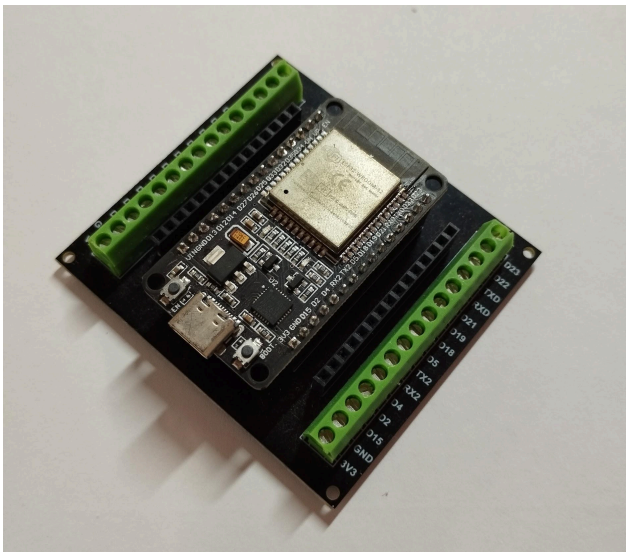


Fonte: Espressif Systems Co. (2025).

O microcontrolador ESP32 é popular por seu preço acessível, versatilidade e comunidade desenvolvedora ativa. O Wi-Fi do ESP32 suporta os modos de operação 802.11b/g/n, possibilitando uma taxa de bits teórica de até 150 Mbps.

A plataforma de desenvolvimento DevKit v1 é popularmente utilizada. O kit possui o módulo ESP-WROOM-32 com periféricos como: regulador de tensão, port USB-C, LED’s e ponte USB para UART. A Figura 14 mostra o kit de desenvolvimento utilizado no projeto.

Figura 14: DevKit v1.



Fonte: O autor (2025).

4. METODOLOGIA DE DESENVOLVIMENTO

A metodologia utilizada para o desenvolvimento do projeto pode ser classificada em uma primeira etapa de implementação e simulação da rede Wi-Fi *Mesh* em ambiente virtual e uma segunda etapa de implementação prática da rede Wi-Fi *Mesh* com microcontroladores. Os resultados da simulação e da prática são enfim comparados. Ambas as etapas envolvem pesquisa bibliográfica, definição da arquitetura da rede e dos testes e programação em C/C++.

4.1 SIMULAÇÃO DA REDE WI-FI *MESH*

A rede Wi-Fi *Mesh* foi criada no simulador de redes OMNet++ em conjunto com o Framework INET. Os arquivos de simulação são escritos em linguagem C++ e NED. A simulação é formada por dois arquivos principais, sendo eles: o arquivo de simulação (.INI) e o arquivo de configuração da rede (.NED).

4.1.1 O Arquivo de Configuração da Rede

O arquivo de configuração da rede inclui a definição dos módulos INET utilizados para a definição do Ambiente Físico (*Physical Environment*), dos nós da rede (estações *Mesh*), do ambiente de Rádio (*Radio Medium*), do configurador IP e do visualizador de dados.

O ambiente físico utilizado foi o modelo “Ieee802.11” do framework INET. Este modelo é responsável por definir obstáculos e outras características do meio físico da simulação. Objetos e obstáculos de diversos tipos de materiais e com geometrias distintas podem ser incluídos na simulação, impactando na propagação dos sinais Wi-Fi. As características do ambiente físico são definidas pelo usuário a partir de arquivos xml.

Os nós da rede representam as estações Mesh do projeto. São portanto os dispositivos responsáveis pela transmissão, recepção, roteamento, entre outras funções relacionadas à comunicação. O framework INET já inclui vários modelos de dispositivos de comunicação *Ad-hoc* capazes de se comunicar através do protocolo IEEE 802.11. Para o projeto foi utilizado o nó *DsdvRouter*, que é um *Ad-hoc Host*

que utiliza o protocolo de roteamento DSDV (*Destination Sequenced Distance Vector*). Porém, isto não influencia na simulação, já que foi utilizado roteamento estático. Além disso, os nós foram posicionados em diferentes regiões internas da casa.

O ambiente de rádio serve para configurar as características de rádio frequência da simulação. Entre elas o modelo de propagação, características da antena, potência de transmissão, interferência, ruído, etc. Todos estes parâmetros em conjunto com o ambiente físico servem para aproximar a propagação dos sinais Wi-Fi da realidade.

O configurador IP é um módulo responsável pela atribuição de endereços IP para os nós da rede, estabelecendo assim a camada de rede necessária para a utilização de protocolos TCP, UDP e implementação do Matter. Apesar do Matter utilizar endereçamento IPv6, foi utilizado um configurador IPv4 no simulador devido a limitações na implementação dos nós *Ad-hoc*.

Configurações relacionadas à visualização dos sinais, dos objetos e de outras informações na GUI do simulador são aplicadas pelo módulo Visualizador.

4.1.2 Arquivo de Simulação

O arquivo de simulação define os parâmetros que serão utilizados no cenário. Os módulos criados no arquivo da rede normalmente utilizam configurações padrão que precisam ser alteradas de acordo com o objetivo do projeto.

Neste arquivo foi definido um tempo de simulação de 120 segundos e uma área física de simulação de 25x15x10 metros em coordenadas cartesianas. Como piso, foi utilizado o modelo *FlatGround* com elevação de zero metros e com o arquivo casa.xml foram definidas as paredes e materiais que compõem o ambiente de teste, constituindo a parte interna de um edifício residencial.

O ambiente de rádio foi definido a partir do modelo INET *Ieee80211ScalarRadioMedium*, projetado para redes Wi-Fi. Foi também utilizado o modelo de perdas por propagação *LogNormalShadowing*, que é a implementação do modelo de perdas Log Normal com objetos físicos interferindo na propagação. O modelo *LogNormalShadowing* recebe dois parâmetros: alpha e sigma. O parâmetro alpha é o expoente de perda do caminho (*Path Loss*) e vem do modelo de propagação Espaço Livre. Alpha foi definido como 4, para ambientes indoor com

obstrução (DESIMONE 2015). O parâmetro sigma é o desvio padrão da variável aleatória de distribuição Gaussiana que produz o efeito de sombreamento no modelo de propagação (DESIMONE 2015). Sigma foi definido como 7, também visando uma simulação mais realista para ambientes indoor. A perda na potência do sinal gerada pelo modelo Sombreamento Log Normal é calculada pela Equação 1.

$$PL(d) = PL(d0) + 10\alpha log(\frac{d}{d0}) + \chi \quad (1)$$

Onde:

χ é uma variável aleatória de distribuição Gaussiana;

d0 e d são distâncias do transmissor;

α é o expoente de perda.

No módulo configurador IP foram definidos os endereços IPv4 dos nós da rede de forma manual utilizando o CIDR 10.0.0.0/29.

Os nós foram configurados para enviarem mensagens utilizando o protocolo TCP. O nó A foi definido como servidor TCP, abrindo uma porta para cada um dos nós da rede. Os outros nós foram configurados para se conectarem com as portas TCP do servidor, realizando 10 requisições de 120 Bytes para cada sessão estabelecida. O servidor responde cada requisição com um pacote de 1400 Bytes. O tamanho dos pacotes foram definidos usando o programa *Wireshark* para gravar os pacotes TCP ao acessar um site na internet por um navegador comercial. Para cada requisição há um tempo de espera de um segundo (*Think Time*). O tempo entre as sessões (*Idle Time)* foi definido como 10 segundos.

Por fim, foram definidos os aspectos físicos da WLAN. O modelo de rádio utilizado foi o *Ieee80211ScalarRadio* (trabalha em conjunto com ambiente de rádio escolhido). Baseado no protocolo 802.11b, utilizado para microcontroladores ESP32, foi definida uma taxa de bits de 1 Mbps para a comunicação. O valor da potência de transmissão foi definido como o máximo suportado pelo ESP32, sendo de 19,5 dBm. A sensibilidade é um parâmetro que depende do modo de operação utilizado para a comunicação. Sendo assim, foi escolhido o padrão suportado pelo *Wi-Fi Driver* do ESP-IDF, que consiste de uma taxa de bits de 1 Mbps. Neste modo a sensibilidade

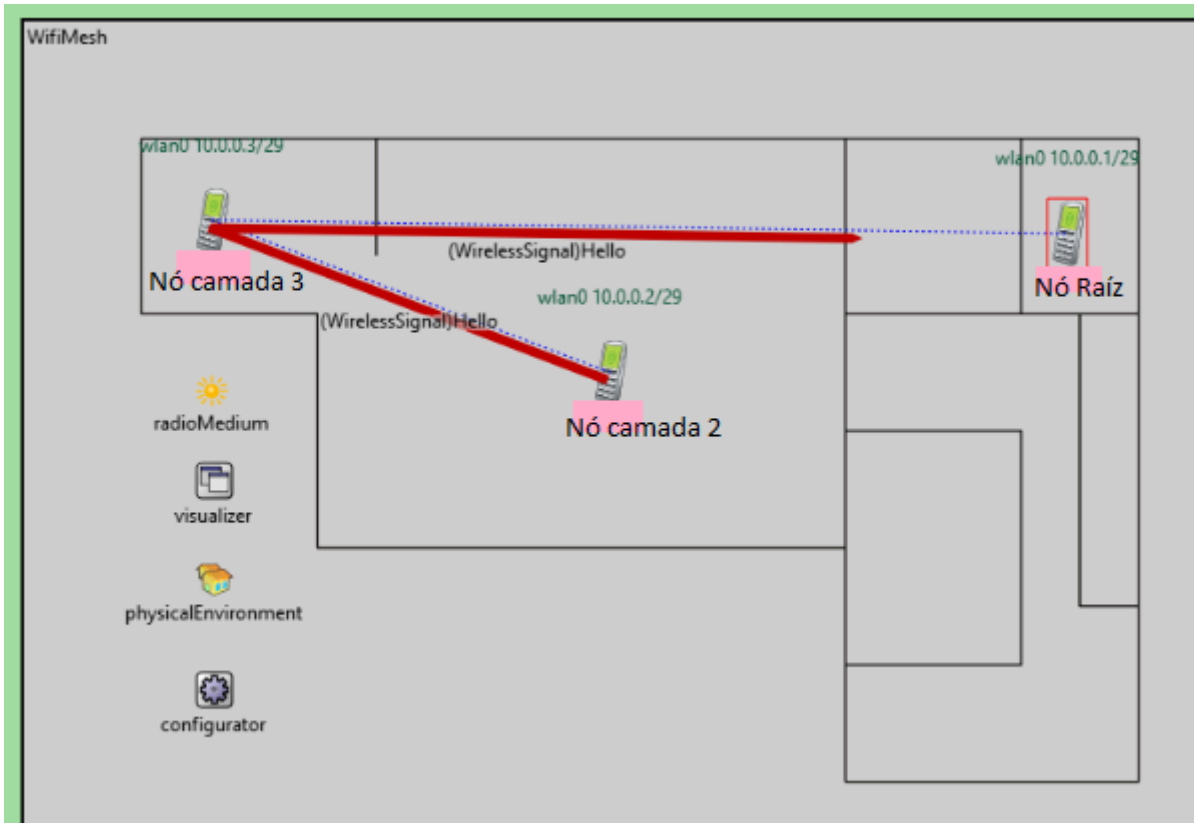
máxima é de -97 dBm. O canal de comunicação escolhido para o teste foi o canal 6 da banda Wi-Fi de 2,4 GHz.

4.1.3 Simulação com Três Nós e Protocolo TCP

Foi realizada uma simulação de 120 segundos com três nós participando da comunicação, representando os três microcontroladores ESP32, em um ambiente interno residencial modelado de acordo com o ambiente de teste físico. Nesta simulação o protocolo de transporte utilizado foi o TCP, por se tratar de um protocolo bastante utilizado em aplicações de internet e ser suportado pelo protocolo Matter.

A Figura 15 mostra a interface gráfica da simulação contendo os nós da rede e o ambiente. Durante a simulação são mostradas informações sobre os pacotes enviados entre os nós, incluindo roteamento e aplicação TCP.

Figura 15: Interface gráfica da simulação.



Fonte: O autor (2024).

4.1.4 Simulação TCP com Alta Taxa de Dados

Nesta simulação foi utilizado o mesmo cenário da simulação TCP com três nós. Entretanto, a aplicação TCP foi modificada para gerar um tráfego maior de dados, simulando, por exemplo, a transferência de vídeos dos nós da camada 2 e 3 para o servidor no nó raíz. Os dados modificados no arquivo INI foram:

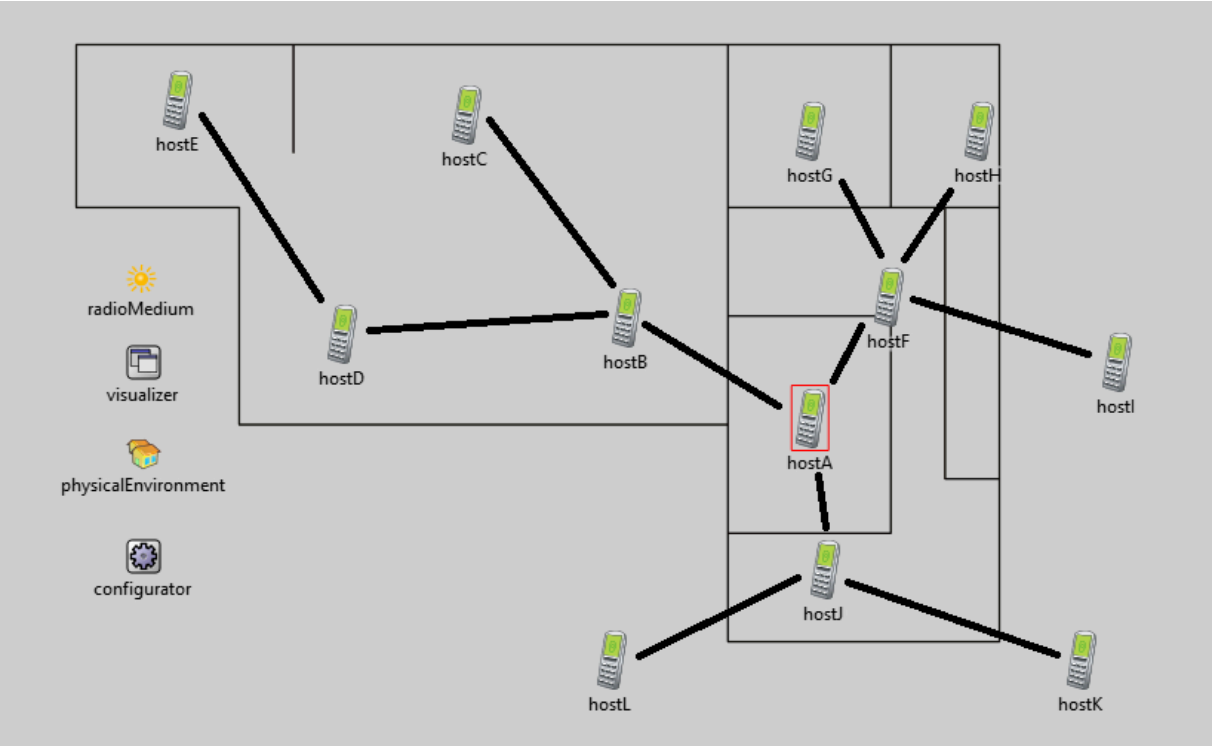
- Tamanho do pacote enviado ao servidor: 1296 Bytes;
- Tamanho do pacote de resposta do servidor: 120 Bytes;
- Período de envio dos pacotes (*Think Time*): 2,4 ms;
- Número de envios por sessão: 386;
- Tempo entre sessões: 10 segundos.

O restante das configurações foram mantidas iguais ao teste anterior. A simulação foi executada para o mesmo número de nós.

4.1.5 Simulação com 12 Nós e Protocolo TCP

Foi realizada uma simulação usando o protocolo TCP e doze nós configurados em uma topologia *Mesh* Árvore, onde o nó raiz é denominado como hostA. A topologia foi definida de forma manual configurando a tabela de roteamento de cada um dos nós da simulação. Não foi utilizado o protocolo de roteamento DSDV, pois a topologia seria definida de forma dinâmica e, considerando a proximidade e potência de transmissão, as conexões seriam realizadas de forma direta entre os nós, ou seja, sem saltos. Desse modo, foi escolhida uma forma de roteamento estático para observar o comportamento da rede com a topologia em árvore (método utilizado pela biblioteca ESP-WIFI-MESH) e com uma quantidade maior de nós envolvidos na comunicação. A Figura 16 mostra a topologia formada.

Figura 16: Topologia Mesh em Árvore da simulação com doze nós.



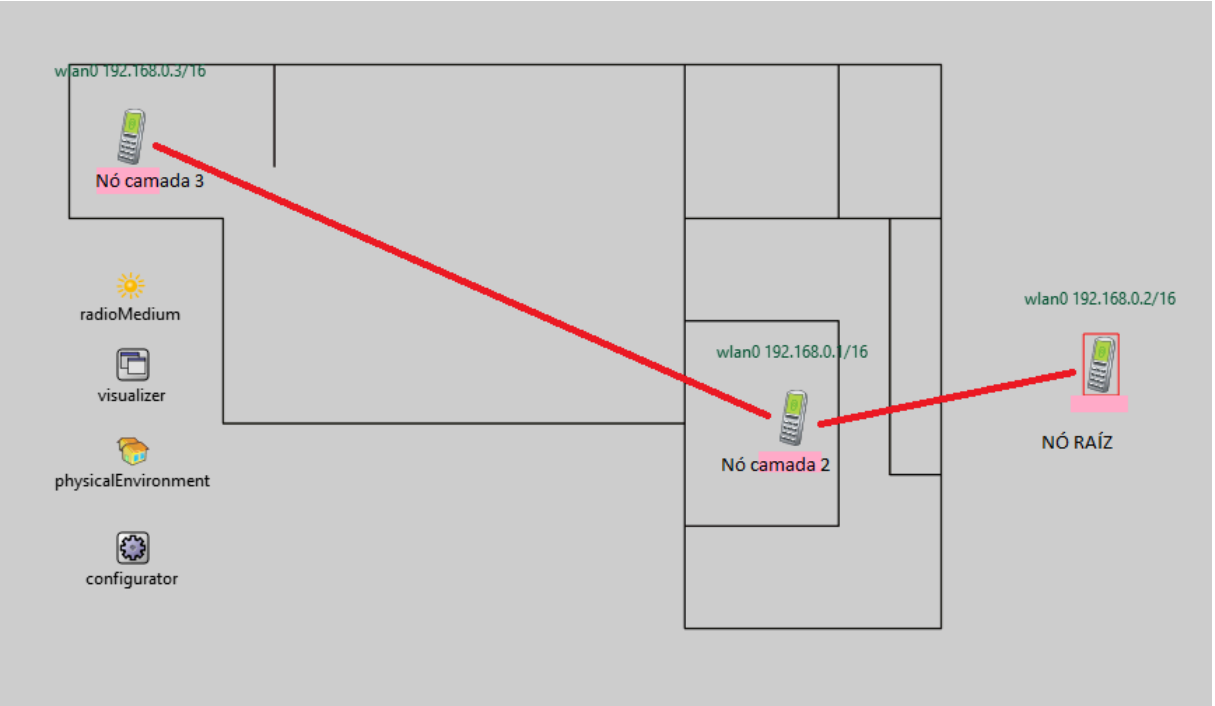
Fonte: O autor (2024).

Os links TCP foram formados conectando os nós em pares: AL, BK, CJ, DI, EH, FG. Cada nó é constituído por um servidor TCP e um cliente TCP, realizando uma comunicação bidirecional. O restante das configurações foram iguais às usadas no primeiro teste com o protocolo TCP no capítulo 3.1.3.

4.1.6 Simulação com Três Nós e Protocolo UDP

Foi realizada uma simulação com o protocolo de transporte UDP e três nós envolvidos na comunicação. O protocolo de transporte UDP é suportado pelo padrão Matter e é muito utilizado na prática com aplicações envolvendo transmissão de dados de sensores. Este protocolo difere do TCP principalmente pela ausência de sessões e controle de fluxo, os pacotes são enviados ao destino sem confirmação de recebimento e sem considerar o congestionamento na rede. Neste aspecto o protocolo UDP é interessante para testar os limites de taxa de transmissão. A Figura 17 mostra a interface gráfica da simulação com os nós e suas conexões.

Figura 17: Topologia da simulação UDP com três nós.



Fonte: O autor (2024).

A topologia utilizada foi configurada de forma manual e estática, para que o nó da camada 3 enviasse pacotes UDP com destino ao nó raíz por encaminhamento através do nó da camada 2.

A aplicação UDP foi configurada para enviar pacotes de 1000 Bytes com um período constante de 2,4 ms. O nó raíz recebe estes pacotes nas portas 5000 e 5005, sem enviar nenhum dado ou confirmação de recebimento aos transmissores, configurando uma comunicação unidirecional.

4.2 TESTES PRÁTICOS COM A REDE WI-FI *MESH*

Os testes práticos com a rede Wi-Fi *Mesh* foram realizados utilizando o microcontrolador ESP32 na placa de desenvolvimento DevKit. Para a edição do código foi utilizada a plataforma ESP-IDF da Espressif na versão Linux (ESPRESSIF SYSTEMS, 2025). O código foi escrito utilizando a linguagem C e as bibliotecas disponibilizadas pela plataforma, especialmente a biblioteca padrão "*sys/socket.h*" (Beej, B. H., 2025). Os testes foram realizados com dois códigos diferentes: um para a comunicação TCP e um para a comunicação UDP.

4.2.1 Implementação Prática da Rede Wi-Fi *Mesh*

A rede *Mesh* foi implementada utilizando o exemplo '*ip_internal_network'*, da pasta de exemplos do projeto ESP-WIFI-MESH, disponível no *GitHub*. Algumas alterações importantes feitas no código foram:

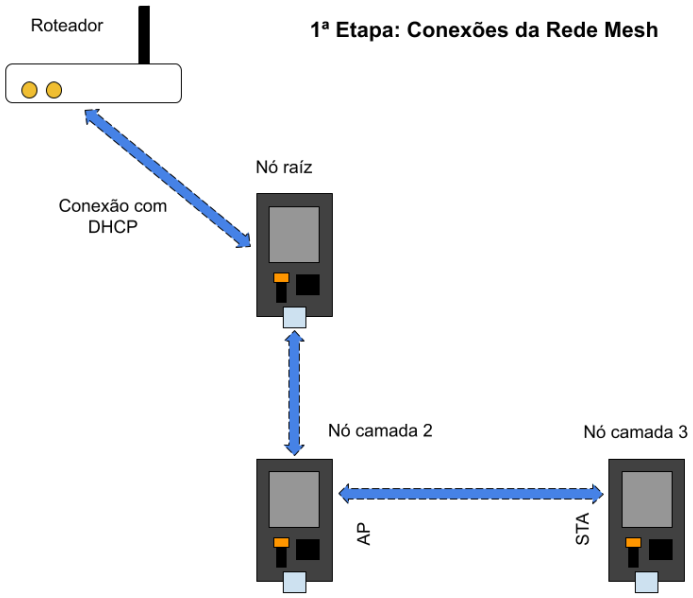
- Desativação do protocolo DHCP em todas as interfaces de rede exceto na interface 'station' do nó central, pois esta interface se conecta ao roteador.
- Configuração manual de endereços IPv4 dos nós da rede, incluindo *gateway*.
- Limitação do número de ramificações na topologia para forçar uma configuração igual à simulação.

O protocolo DHCP é ativado no exemplo *'ip_internal_network*' e elimina a necessidade de atribuir endereços manualmente aos nós da rede. Porém, a atribuição manual foi escolhida como forma de manter o teste definido de forma exata e facilitar o preenchimento do destino nos pacotes enviados na rede. Portanto, endereços foram atribuídos a todas as interfaces de rede e o protocolo DHCP foi desativado.

A limitação do número de conexões na interface AP dos nós foi utilizada para evitar que os dois nós se conectassem diretamente ao nó central. Para isso, foi criada uma variável no arquivo "Kconfig.projbuild" chamada "MESH_AP_CONNECTIONS", que define o número de conexões que podem ser realizadas na interface AP de cada nó. No arquivo "mesh_main.c", a variável "cfg.mesh_ap.max_connection" recebe o valor determinado. Com a limitação, apenas um nó fica conectado ao nó central e fica responsável pelo roteamento das mensagens do segundo nó, assim como nas simulações.

Além disso, foi utilizada a configuração de nó central manualmente definido. Portanto, não existe uma fase de votação para o nó central, como é descrito na documentação ESP-WIFI-MESH (ESPRESSIF SYSTEMS Co. 2024). Este nó central deve ser conectado a um roteador para que a rede Mesh seja estabelecida sem erros. A Figura 18 mostra a primeira etapa do teste prático, que corresponde à criação da rede WiFi *Mesh*. Nesta etapa, o nó raíz se conecta a um roteador WiFi e inicializa a rede *Mesh*.

Figura 18: Criação da rede ESP-WIFI-MESH.



Fonte: O autor (2025).

O próximo nó identifica a rede mesh e se conecta ao nó raíz. O último nó se conecta ao nó da segunda camada. Caso não seja determinado um nó pai específico, os nós darão preferência para pais em camadas menores, por isso foi imposto o limite de conexões na interface AP dos nós.

Algumas destas alterações foram realizadas tanto nos arquivos '.c' como na configuração geral do projeto, que pode ser acessada com a ferramenta 'idf.py menuconfig'. Esta ferramenta é parte do framework ESP-IDF e não está presente no Arduino IDE. Outra alteração importante foram as inclusões de funções customizadas para a comunicação TCP/UDP entre os nós e o envio de informações.

4.2.2 Etapas do Teste Prático

Para realizar os testes de eficiência da Rede *Mesh*, foram criados servidores e clientes referentes aos protocolos de transporte utilizados na etapa de simulação (TCP e UDP). No nó raíz foi criado um servidor de testes e um servidor de sinalização, cujas funções são respectivamente: receber/enviar pacotes de acordo com os testes simulados e sinalizar o início do teste para os nós da rede.

Clientes TCP/UDP foram criados nos demais nós da rede. Estes se conectam aos servidores de teste e de sinalização. No caso do teste UDP, não é realizada conexão com o servidor de testes, pois o protocolo não exige esta etapa.

A Figura 19 mostra as conexões realizadas nesta etapa do teste.

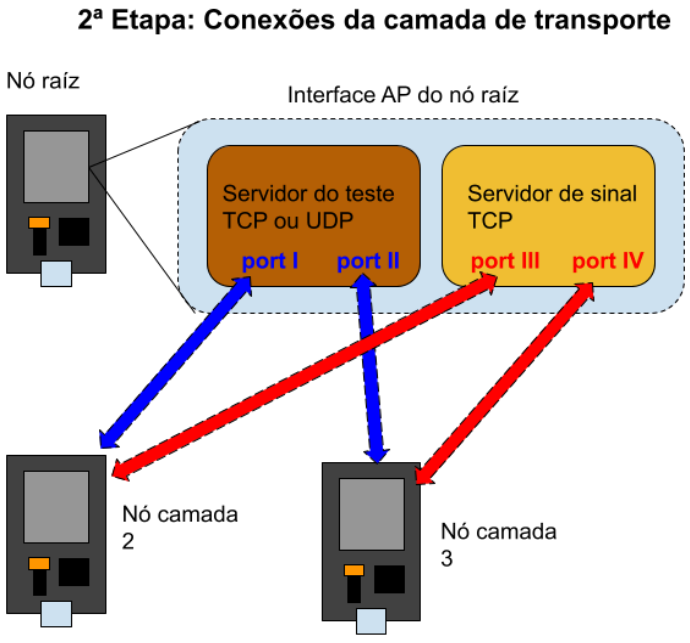
Com as conexões estabelecidas, os nós ficam aguardando o recebimento do sinal que inicia a troca de mensagens. O sinal é enviado para os nós quando o usuário aperta um botão localizado no nó raíz.

Quando um nó recebe o sinal, um cronômetro é acionado e o envio de mensagens começa. Esta operação é representada na Figura 20.

A próxima etapa consiste na troca de pacotes de forma correspondente às simulações realizadas. Para isso foram contabilizados os intervalos entre o envio de mensagens e também o intervalo entre sessões no caso do TCP. Assim como na simulação, são enviados 386 pacotes para cada sessão. Quando o limite é atingido, a sessão é fechada e uma nova é estabelecida.

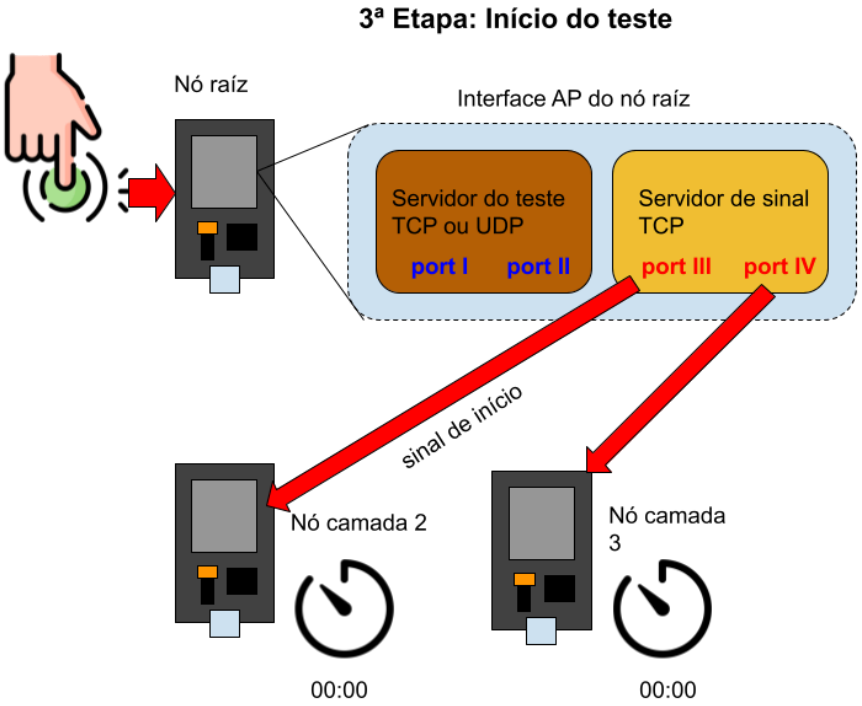
Para o protocolo UDP não são criadas conexões e não existem sessões. Portanto, o envio é realizado de forma direta, sem confirmação de entrega e com um intervalo constante de 2,4 ms entre os envios.

Figura: 19: Conexões da camada de transporte.



Fonte: O autor (2025).

Figura 20: Início do teste da rede.



Fonte: O autor (2025).

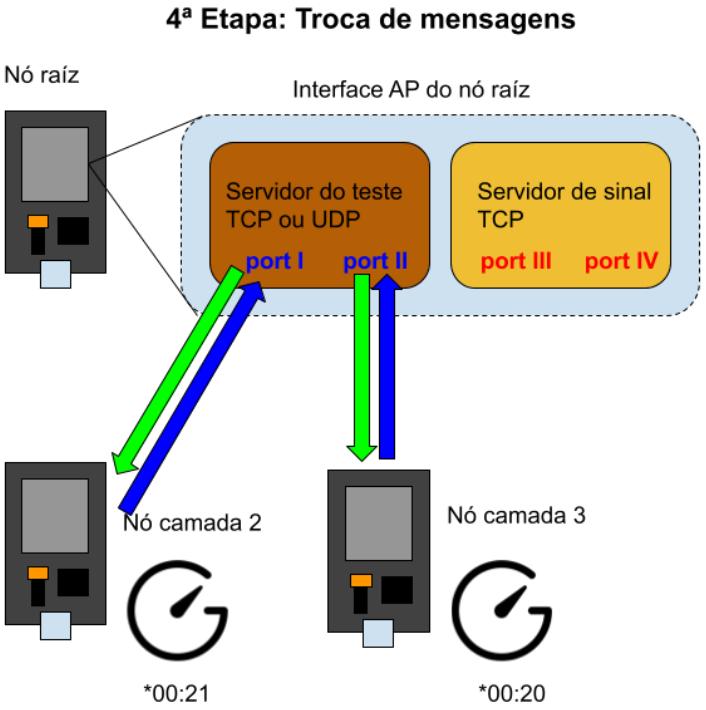
Ao passo que os pacotes são enviados e recebidos na rede, são registrados:

- O número de pacotes enviados;
- O número de pacotes recebidos;
- Quantidade de Bytes enviados;
- Quantidade de Bytes recebidos.

A Figura 21 mostra o processo de troca de mensagens realizado nesta etapa, lembrando que os pacotes do nó da camada 3 trafegam na rede através do nó da camada 1. A representação leva em conta apenas o destino e a fonte do pacote.

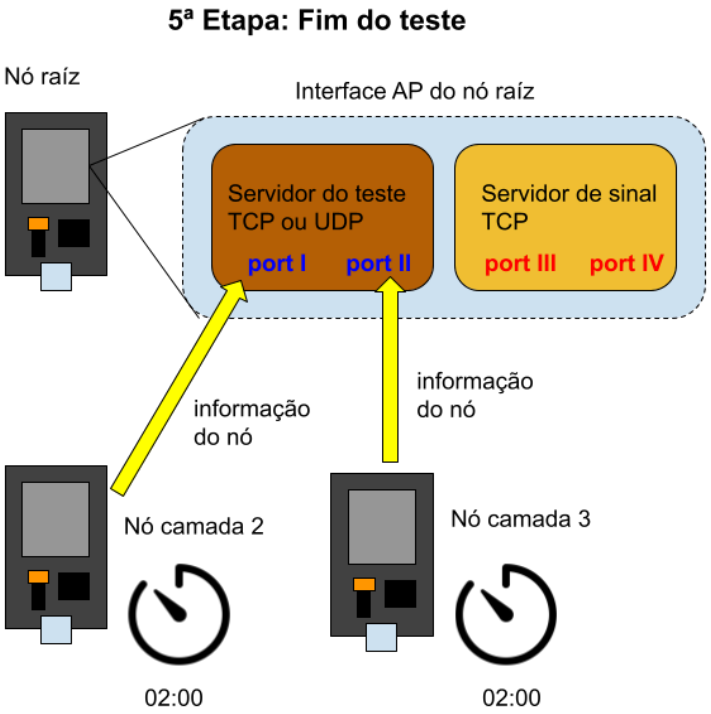
Quando os cronômetros atingem o limite de dois minutos, o envio de pacotes é encerrado pelos nós e estes enviam um último pacote de dados antes de encerrarem a conexão. O último pacote contém as informações adquiridas por aquele nó durante o teste. O servidor recebe o pacote, decodifica e apresenta as informações ao usuário. A Figura 22 mostra a etapa final do teste.

Figura 21: Execução do teste.



Fonte: O autor (2025).

Figura 22: Envio do resultado do teste.



Fonte: O autor (2025).

A aquisição da latência, na simulação, foi realizada através da métrica RTT fornecida pelos pacotes UDP e TCP. Porém, nos testes práticos, foi utilizado o comando *Ping*, ou seja, o protocolo ICMP. O resultado gerado pelo comando *Ping* é análogo ao RTT, e foi implementado com base no exemplo disponível no ESP-IDF. Assim como no teste de *Throughput*, o usuário aperta um botão físico na placa ESP32 para iniciar o envio de pacotes ICMP, sendo este o botão *built-in* 'BOOT'. A aplicação *Ping* foi configurada para enviar cinco pacotes a um destino específico, disponibilizando os resultados de latência no terminal. Foram utilizados dois enlaces para testar a latência, organizados em origem e destino:

- Nó Raíz → Nó camada 3;
- Nó camada 2 →Nó raíz.

O botão no nó de origem foi pressionado durante a execução do teste de *Throughput*, representado na Figura 21, visando aproximar os resultados de latência do caso simulado.

Foi utilizada uma amostra de 15 medidas para cada enlace em cada cenário testado. Dessas 15 medidas foram obtidas a média aritmética e o desvio padrão.

4.3 ANÁLISE DE RESULTADOS

Os resultados obtidos da simulação foram processados por um código em Python a partir dos dados em formato JSON gerados pelo OMNet++. Os módulos UDP e TCP produzem informações para cada nó da simulação, fornecendo o número total de Bytes transmitidos e o número total de Bytes recebidos. Levando em conta estas informações e o tamanho dos pacotes utilizados na comunicação, é possível determinar: o *throughput* de cada nó, a quantidade de pacotes transmitidos e recebidos e a taxa de perda de pacotes.

O *throughput* foi calculado através de duas relações: o número total de Bytes enviados por determinado nó e o tempo total de transmissão; o número total de Bytes recebidos pelo tempo total de recepção, como é indicado pelas Equações 2 e 3. O tempo de recepção e transmissão utilizados para as simulações e a prática foi de 120 segundos.

$$Throughput\ UpLink = \frac{Total\ de\ pacotes\ enviados\ (Byte) \times 8}{Tempo\ de\ simulação} \qquad (2)$$

$$Throughput\ DownLink = \frac{Total\ de\ pacotes\ recebidos\ (Byte) \times 8}{Tempo\ de\ simulação} \quad (3)$$

A taxa de perda de pacotes, ou *Packet Loss Ratio*, foi calculada a partir da Equação 4 (MEENA, P. et al. 2021). Esta equação fornece em porcentagem a quantidade de pacotes perdidos durante o teste.

$$PLR = 100 - (Total\ de\ pacotes\ Recebidos) * 100/(Total\ de\ Pacotes\ Enviados) \quad (4)$$

Os dados gerados pelo teste prático foram: a quantidade de Bytes enviados e recebidos por cada um dos nós. Os nós das camadas 2 e 3 enviam esta informação ao nó raíz que disponibiliza os dados no terminal. O resultado da latência foi obtido através do protocolo ICMP para o teste prático. O cálculo das métricas utiliza as mesmas equações da simulação.

5. RESULTADOS

Foram obtidos resultados referentes às simulações realizadas da rede Wi-Fi *Mesh* nos diferentes cenários testados. Os resultados foram analisados de forma a obter uma visão geral do desempenho, focando nas métricas: Perda de Pacotes, Latência com RTT ou Ping (ICMP) e *Throughput* (Taxa de transferência de dados).

O processamento dos dados foi realizado exteriormente ao simulador, utilizando a linguagem Python 3.11.9 e os pacotes Numpy, Statistics, Json e Matplotlib. Também foi utilizado o *Google Sheets* para criação dos gráficos comparativos entre simulação e prática.

5.1 RESULTADOS DA SIMULAÇÃO

5.1.1 Resultados da Rede com Três Nós e Protocolo TCP

Os valores de latência obtidos nesta simulação são gerados pela aplicação TCP do framework INET. Para cada sessão criada com o protocolo TCP, são gerados os dados de RTT em função do tempo, forma de dados classificada como vetorial no simulador. A informação sobre o tempo foi omitida dos resultados, já que o objetivo é a obtenção da latência na rede de uma forma geral. Além disso, os dados foram filtrados para classificar os resultados separadamente para cada nó da rede.

Os dados de RTT foram extraídos em formato JSON e processados por um código em Python. Foram obtidos os valores médios de latência e o desvio padrão para todas as sessões (23 sessões). O nó raíz foi utilizado como referência de medida e, por isso, não apresenta resultados. A Tabela 1 mostra os resultados de latência da simulação.

TABELA 1: RTT com três nós e protocolo TCP

	Nó raíz	Nó camada 2	Nó camada 3
RTT Médio	-	3,18 ms	3,25 ms
Desvio Padrão	-	0,54 ms	0,76 ms

FONTE: O autor (2024).

Observa-se que o nó da camada 3 apresenta uma latência média de 0,07 ms maior que o nó da camada 2. Resultado esperado pela maior distância entre a camada 3 e o nó raíz.

O resultado da taxa de transmissão de dados foi obtido através da aplicação TCP do simulador, que disponibiliza os dados referentes a quantidade de Bytes enviada por cada nó e quantidade de Bytes recebida por cada nó. O tempo considerado para o cálculo do *Throughput* foi o tempo total da simulação, ou seja, 120 segundos. Os dados foram calculados a partir das Equações 2 e 3 do item 4.3.

O resultado é mostrado na Tabela 2, com as quantidades representadas em bits por segundo. O nó raíz é o servidor da rede, recebendo o fluxo dos nós das camadas 2 e 3. Por isso, o resultado de *Throughput* para o nó raíz no sentido *Uplink* (saindo do nó) é igual à soma dos fluxos dos demais no sentido *Downlink* (entrando no nó). No sentido inverso, verificou-se o mesmo comportamento: a soma do *Throughput* dos nós de camada 2 e 3 no sentido *Uplink* é igual ao *Throughput Downlink* para o nó raíz. Vale ressaltar que este é o comportamento esperado apenas quando a perda de pacotes é nula.

TABELA 2: Throughput com três nós e protocolo TCP

	Nó raíz	Nó camada 2	Nó camada 3
Throughput UL	11386 bps	488 bps	488 bps
Throughput DL	976 bps	5693 bps	5693 bps

FONTE: O autor (2024)

O resultado foi de 0% de perda de pacotes, ou seja, nenhum pacote foi perdido durante a simulação. Como o TCP faz retransmissão de pacotes perdidos, é esperado que a perda seja nula.

5.1.2 Resultados da Rede TCP com Alta Taxa de Dados

O resultado para o cenário TCP com três nós e alta taxa de dados é mostrado na Tabela 3, com relação ao RTT medido em 24 sessões.

Foi notado que tanto o valor médio quanto o desvio padrão aumentaram com relação ao teste de baixa taxa de dados. A maior latência pode ser explicada no simulador pelo maior *overhead,* ou seja, maior número de preâmbulos devido à

maior quantidade de pacotes e conexões sendo realizadas. Os valores médios de latência aumentaram aproximadamente em 9 ms, enquanto o desvio padrão aumentou em 6,58 ms.

TABELA 3: RTT com três nós, protocolo TCP e alta taxa de dados.

	Nó raíz	Nó camada 2	Nó camada 3
RTT Médio	-	12,27 ms	12,43 ms
Desvio Padrão	-	7,12 ms	7,36 ms

FONTE: O autor (2024)

A Tabela 4 mostra o resultado desta simulação com relação ao *Throughput* obtido. O comportamento observado é similar ao teste anterior, onde o tráfego é concentrado na estação A. Desta vez o maior valor de *Throughput* ocorreu na recepção do nó raíz, atingindo cerca de 295 kbps. Apesar da quantidade muito maior de dados enviados, a perda de pacotes permaneceu bastante baixa, ficando com um valor de apenas 0,007%.

TABELA 4: Throughput com três nós, protocolo TCP e alta taxa de dados.

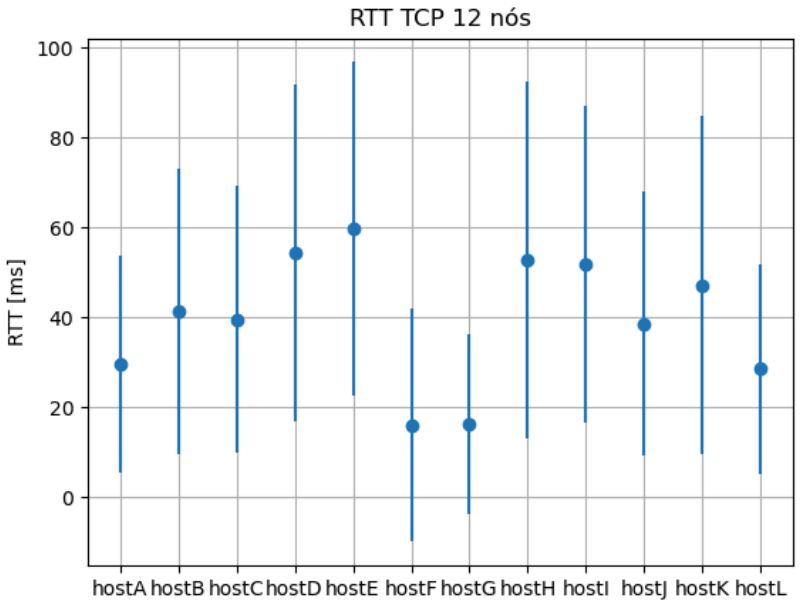
	Nó raíz	Nó camada 2	Nó camada 3
Throughput UL	27304 bps	150508 bps	144547 bps
Throughput DL	294969 bps	13928 bps	13376 bps

FONTE: O autor (2024)

5.1.3 Resultados da Rede com Doze Nós TCP

A simulação com doze nós e protocolo TCP gerou os resultados de latência mostrados no Gráfico 1. O gráfico mostra o resultado médio da latência (ponto) e o desvio padrão calculado (linhas verticais) para cada um dos nós da simulação. Observa-se que a maior parte dos valores RTT calculados ficou dentro da faixa de 20 ms a 60 ms, valores maiores comparados ao cenário com três nós. Além disso, o desvio padrão foi maior em comparação com as outras simulações, indicando uma variação maior da latência na rede. Observa-se também que a distância entre os nós é relacionada com o RTT, de modo que nós mais próximos (F e G) apresentam uma latência menor do que nós mais distantes (A e L). Os nós mais distantes (E e H) apresentaram também a maior latência.

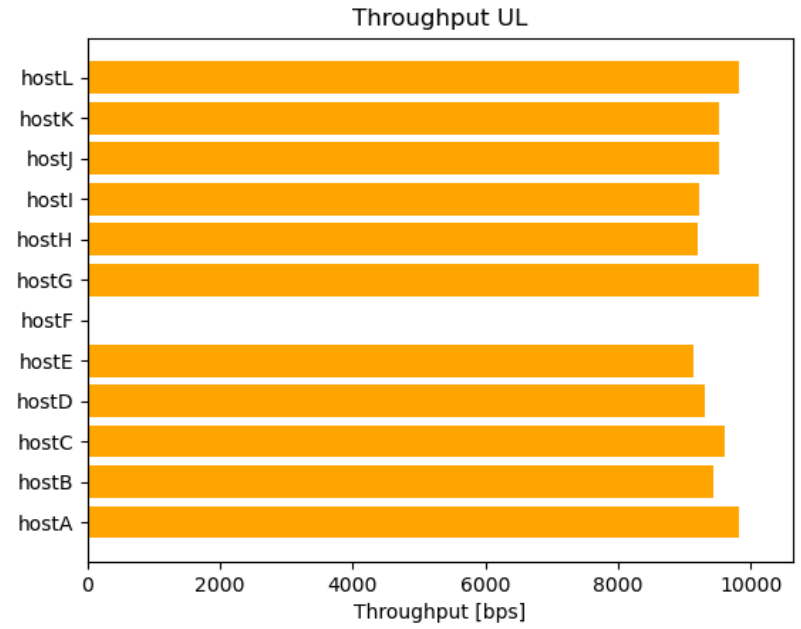
Gráfico 1: RTT da simulação TCP com 12 nós.



Fonte: O autor (2024).

Os resultados de *Throughput* apresentaram os comportamentos indicados nos gráficos 2 e 3. O Gráfico 2 mostra a taxa de dados média para as conexões *Uplink* na rede.

Gráfico 2: *Throughput* UL simulação TCP com 12 nós.



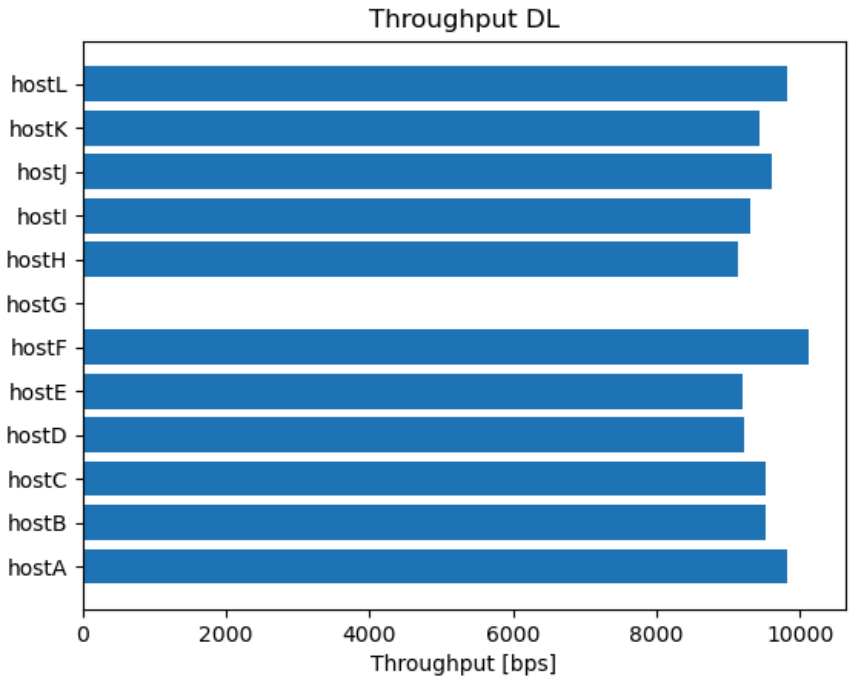
Fonte: O autor (2024).

Foi identificado que os valores de *Throughput* permaneceram na faixa de 9 kbps e 10 kbps. Além disso, nós mais distantes na topologia, apresentaram uma média menor de taxa de transferência de dados do que aqueles que estavam mais próximos.

O Gráfico 3 mostra o *Throughput* no sentido *Downlink*, que mostrou ser muito similar ao resultado UL, especialmente entre os pares que trocaram mensagens entre si.

Em ambos os gráficos foi observada uma lacuna nos resultados. Para o sentido *Uplink* esta lacuna ocorre no nó F, enquanto no sentido *Downlink* ocorre em G. Este é o par de comunicação FG, onde o nó G atua como cliente do servidor F e vice versa. Porém, os nós estabeleceram comunicação, já que os resultados aparecem para ambos os nós, apenas não simultâneamente. Este problema resultou provavelmente de alguma falha no *software*, mas não há indícios de que esta falha tenha prejudicado os resultados de forma crítica. Um teste prático com doze nós poderia ser realizado para efeito de comparação, mas neste trabalho tal análise não será realizada devido a falta de equipamentos para efetuação do teste real.

Gráfico 3: *Throughput* DL simulação TCP com 12 nós.

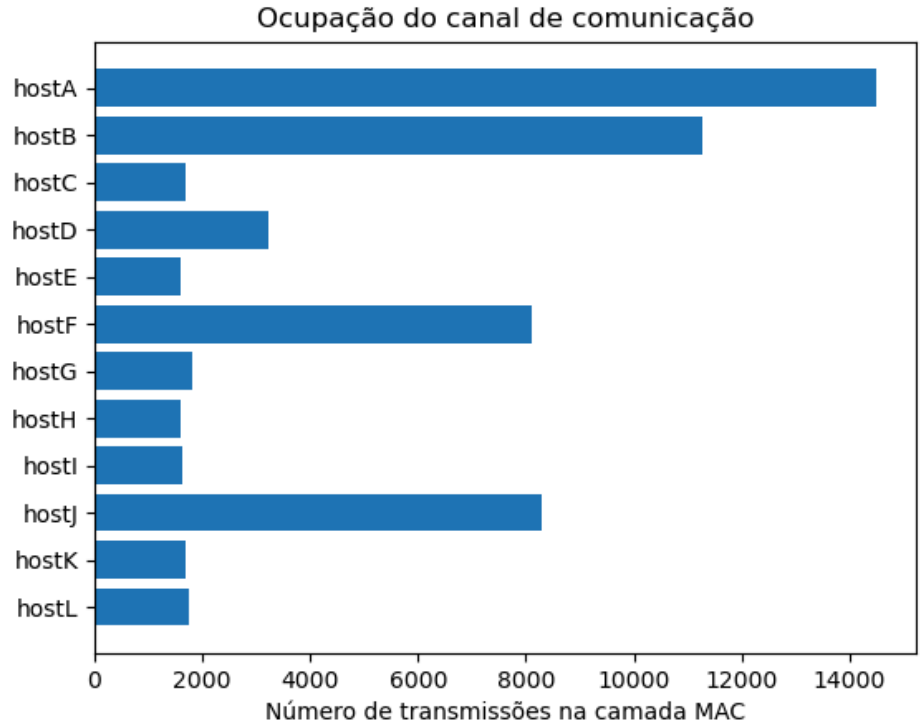


Fonte: O autor (2024).

No Gráfico 4 foi observado que o nó raiz (*hostA*) apresenta o maior número de transmissões na rede. Isso era teoricamente esperado da topologia Mesh em árvore, já que o nó A é responsável pelo roteamento de pacotes entre todas as ramificações. Além disso, os nós da segunda camada na topologia (B, F e J) apresentaram os valores mais altos de transmissão de pacotes depois do *host A*. Novamente, isso acontece pois estes nós são responsáveis pelo roteamento de todos os outros da camada abaixo, aumentando assim a necessidade deles transmitirem um número maior de pacotes. Por fim, observa-se que o nós folha ocuparam o canal de forma similar, por não realizarem roteamento de outros nós.

A taxa de perda de pacotes para a simulação TCP com doze nós foi nula, ou seja, 0%. Portanto, nenhum pacote foi perdido durante os 120 segundos de simulação.

Gráfico 4: Ocupação do canal para a simulação TCP com 12 nós.



Fonte: O autor (2024).

5.1.4 Resultados da Rede com Três Nós e Protocolo UDP

Os resultados referentes ao valor de latência observado na simulação com três nós e protocolo UDP são apresentados na Tabela 5. Observou-se que os valores de RTT permaneceram relativamente constantes, devido aos valores baixos de desvio padrão calculados. Além disso, o valor médio de latência foi baixo em comparação com o teste similar utilizando TCP. Não foram obtidos resultados de latência para o nó raíz, pois este foi utilizado como referência para a medição dos outros nós.

TABELA 5: RTT com três nós e protocolo UDP.

	Nó raíz	Nó camada 2	Nó camada 3
RTT Médio	-	2,84 ms	5,20 ms
Desvio Padrão	-	0,40 ms	0,88 ms

FONTE: O autor (2024)

Já na Tabela 6 estão contidos os resultados de *Throughput* derivados da simulação. O nó da camada 2 atingiu uma taxa média de 554 kbps e o nó da camada 3 de 303 kbps. O nó da camada 3 apresentou uma taxa menor de transmissão, provavelmente devido à sua distância relativa ao nó raíz e pelo fato de seus pacotes estarem sendo roteados pela camada 2. Portanto, conclui-se que para uma rede em *Mesh* maior, nós mais distantes de seus destinos apresentariam taxas menores de transferência de dados e latências maiores.

Nos dados gerados pelo simulador, foi verificado que a taxa de dados média atingiu o nível máximo de capacidade de 1 Mbps na camada MAC.

TABELA 6: *Throughput* com três nós e protocolo UDP.

	Nó raíz	Nó camada 2	Nó camada 3
Throughput UL	-	427088 bps	428112 bps
Throughput DL	855200 bps	-	-

FONTE: O autor (2024)

Por fim, foram calculadas as taxas de perda de pacotes na rede, indicando a eficiência da comunicação. Os resultados mostraram que a taxa de perda de pacotes para o nó da camada 2 foi de 82,75% e para o nó da camada 3 de 90,58%. O resultado total da rede foi de uma perda de 86,67% dos pacotes transmitidos.

Portanto, a eficiência da comunicação é bastante prejudicada utilizando UDP como protocolo de transporte. Novamente, observou-se que a eficiência na comunicação é maior para os nós mais próximos do destino na rede *Mesh*. Mas a distância do nó da camada 3 e o processo de roteamento de seu pacote acarretou em uma perda apenas 8% maior.

5.2 RESULTADOS PRÁTICOS

Os resultados obtidos dos testes na plataforma ESP32 da rede WiFi *Mesh* desenvolvida foram separados de acordo com as simulações realizadas. As métricas obtidas diretamente através do código foram: total de Bytes enviados e recebidos, total de pacotes, enviados e recebidos, e latência através do Ping. As estatísticas foram obtidas através destes dados utilizando o cálculo apresentado no capítulo de simulações e o tempo de simulação de 120 segundos.

Foi verificado na prática que os nós utilizam o protocolo 802.11n com banda de transmissão HT40 (40 MHz de banda), com taxa nominal de 600 Mbps com tecnologia MIMO, para a comunicação nas condições do teste. Esta configuração é a padrão para o *Wi-Fi Driver* dos dispositivos ESP32, como descrito na documentação da Espressif. Além disso, é descrito na documentação do módulo *Wi-Fi Driver* que a taxa nominal padrão é de 1 Mbps. Esta taxa pode ser alterada com o comando “esp_wifi_config_80211_tx_rate()”, porém foi utilizada a taxa *default* de 1 Mbps por corresponder ao limite imposto nas simulações.

Os resultados práticos foram obtidos utilizando as mesmas configurações dos protocolos TCP e UDP das simulações. Para o teste TCP com baixa taxa foram configuradas sessões de 386 pacotes com *Think Time* de 1 segundo e *Idle Time* de 10 segundos. O teste TCP de alta taxa utilizou sessões de 386 pacotes com *Think Time* de 2,4 ms e *Idle Time* de 10 segundos. O tamanho do *payload* (dados de aplicação) dos pacotes também foi o mesmo das simulações: 120 Bytes e 1400 Bytes para baixa taxa e 1296 Bytes e 120 Bytes para alta taxa. O teste UDP utilizou pacotes com 1000 Bytes de *payload* enviados dos nós da rede para o nó central com 2,4 ms de *Think Time*.

5.2.1 Resultado do Teste Prático TCP com Baixo Volume de Dados

O resultado do teste prático para o teste com protocolo TCP para a taxa de dados obtida de cada nó na rede WiFi *Mesh* são apresentados na Tabela 7.

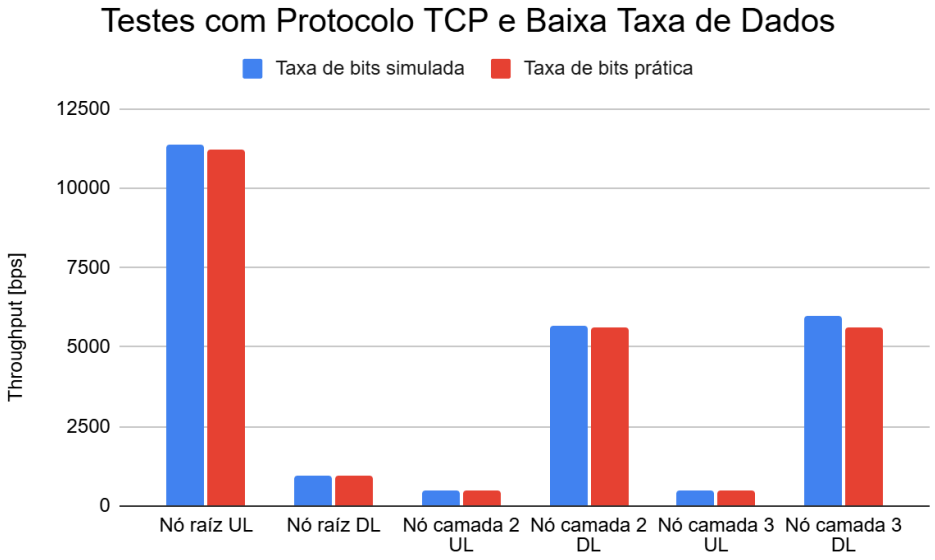
TABELA 7 - *Throughput* do teste TCP prático de baixa taxa.

	Nó raíz	Nó camada 2	Nó camada 3
Throughput UL	11200 bps	480 bps	480 bps
Throughput DL	960 bps	5600 bps	5600 bps

FONTE: O autor (2025)

Comparado com os valores obtidos via simulação os desvios foram mínimos, mostrando que a simulação previu com acurácia o fluxo de dados para a rede *Mesh*. O Gráfico 5 mostra a comparação entre os resultados obtidos por simulação e os obtidos pelo teste prático.

Gráfico 5: *Throughput* do teste TCP com baixa taxa.

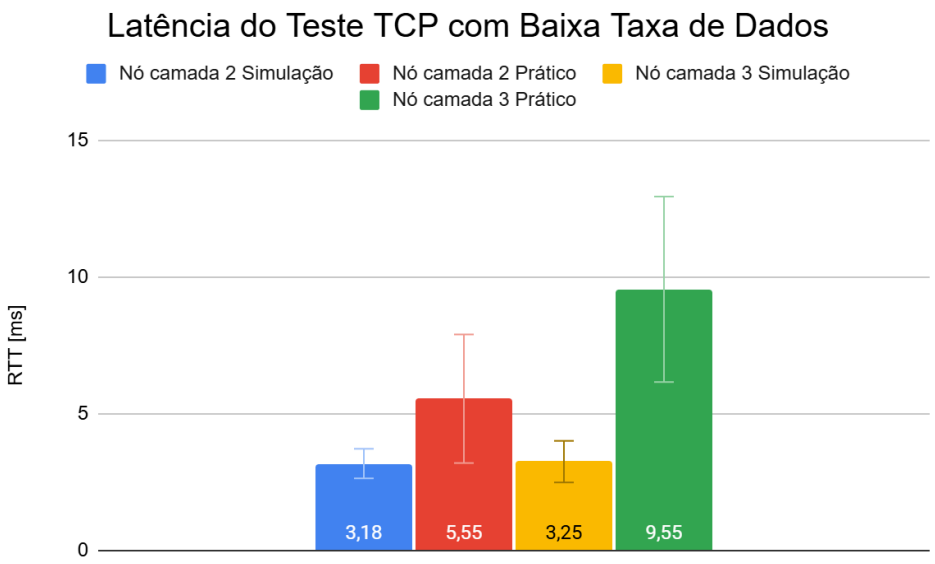


Fonte: O autor (2025).

Os resultados obtidos através do comando PING para o teste com protocolo TCP e baixa taxa de dados indicaram que a latência real foi maior do que a obtida por simulação. Além disso, o desvio padrão dos resultados medidos também foi

maior. O Gráfico 6 mostra os resultados de latência média e os desvios padrão obtidos para os nós da camada 2 e 3.

Gráfico 6: Latência do teste TCP com baixa taxa.



Fonte: O autor (2025).

5.2.2 Resultado do Teste Prático TCP com Alta Taxa de Dados

O resultado obtido com o protocolo TCP durante os testes práticos com alto volume de dados são mostrados na Tabela 8.

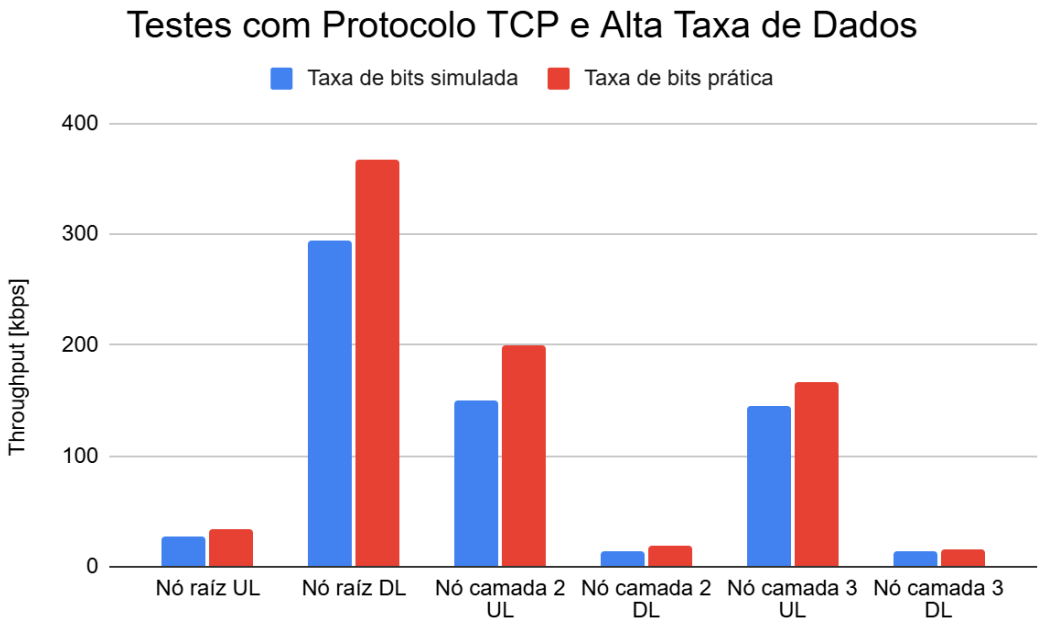
TABELA 8 - *Throughput* do teste TCP prático de alta taxa.

	Nó raíz	Nó camada 2	Nó camada 3
Throughput UL	33968 bps	200102 bps	166752 bps
Throughput DL	366854 bps	18528 bps	15440 bps

FONTE: O autor (2025)

A taxa de dados obtida através do teste prático com alta taxa é muito próxima dos resultados da simulação com alta taxa. Portanto, a simulação e a prática foram novamente muito próximas. Mas, em geral, o teste prático apresentou taxas de transferência maiores do que a simulação. O Gráfico 7 mostra o resultado comparativo entre simulação e prática para este cenário.

Gráfico 7: *Throughput* do teste TCP com alta taxa de dados.



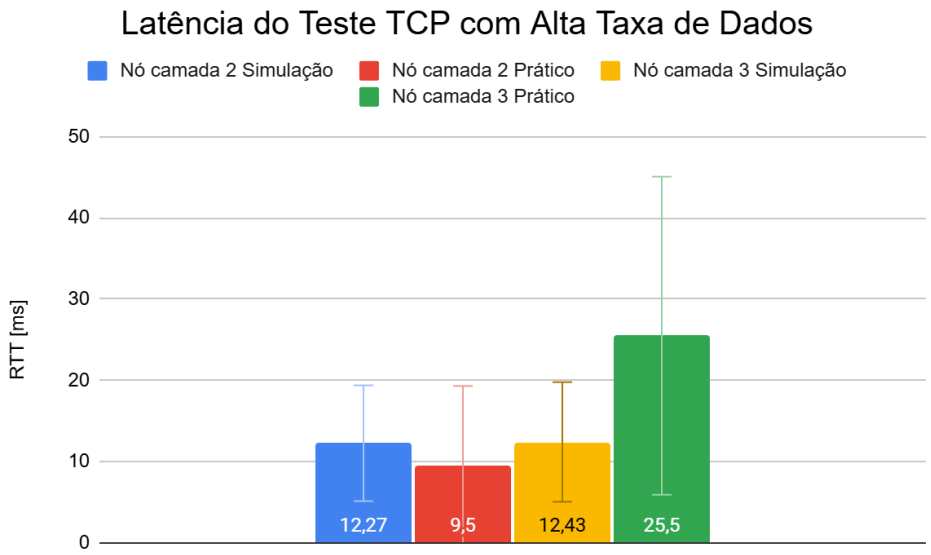
Fonte: O autor (2025).

A diferença entre os resultados práticos e os simulados pode ter sido causada pela diferença entre o modelo de propagação teórico Sombreamento Log Normal e o canal real. Os parâmetros alpha e sigma do modelo poderiam ser ajustados na simulação para aproximar a perda da realidade. Também poderia ser criado um ambiente físico (planta da casa) mais próximo da realidade. Além disso, outros modelos de propagação como o Nakagami poderiam ser utilizados na simulação.

Os resultados obtidos para a latência do teste TCP com alta taxa de transferência são mostrados em conjunto com os resultados simulados no Gráfico 8.

Foi observado que o nó da camada 2 apresentou um RTT médio menor do que na simulação, porém com um desvio padrão maior. Já o nó da camada 3 apresentou um RTT médio bem maior do que o obtido por simulação, além de um desvio padrão grande. Isto indica que há uma maior variação da latência no teste real. Isto pode ser causado por limitações de processamento do ESP32, além de variações do modelo real e simulado.

Gráfico 8: Latência do teste TCP com alta taxa.



Fonte: O autor (2025).

5.2.3 Resultado do Teste Prático UDP

O resultado obtido com o teste prático do protocolo UDP é mostrado na Tabela 9 abaixo.

Foi observado que o teste prático apresentou uma performance superior ao resultado simulado. O teste prático alcançou uma taxa de 692 kbps enquanto a simulação atingiu 428 kbps para o nó da camada 2. Para o nó raíz foram atingidos aproximadamente 1,27 Mbps de tráfego efetivo Downlink na prática. O *Throughput Downlink* do nó raíz foi calculado simplesmente como a soma dos *Throughputs Uplink* dos nós da camada 2 e da camada 3. É interessante observar que 1,27 Mbps é maior do que a taxa nominal utilizada pelo *Wi-Fi Driver* de 1 Mbps. Isto é resultado do modo como a taxa no nó raíz foi calculada, que não considera se os nós estão transmitindo simultâneamente ou não durante o intervalo de teste.

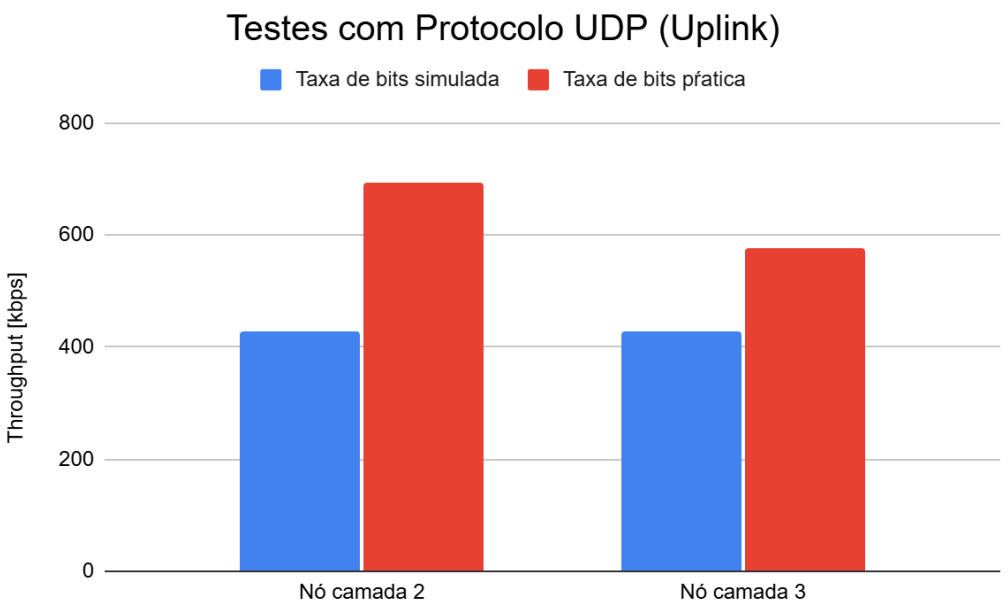
TABELA 9 - *Throughput* do teste UDP prático.

	Nó raiz	Nó camada 2	Nó camada 3
Throughput UL	-	692200 bps	575067 bps
Throughput DL	1267267 bps	-	-

FONTE: O autor (2025)

A taxa de perda de pacotes calculada para o teste prático UDP foi de 74%, ou seja, apenas 26% dos pacotes enviados chegaram ao destino. Já o teste simulado obteve uma taxa de perda de 86%, ou seja, 12% maior que o teste prático.

Gráfico 9: *Throughput* dos testes UDP.



Fonte: O autor (2025).

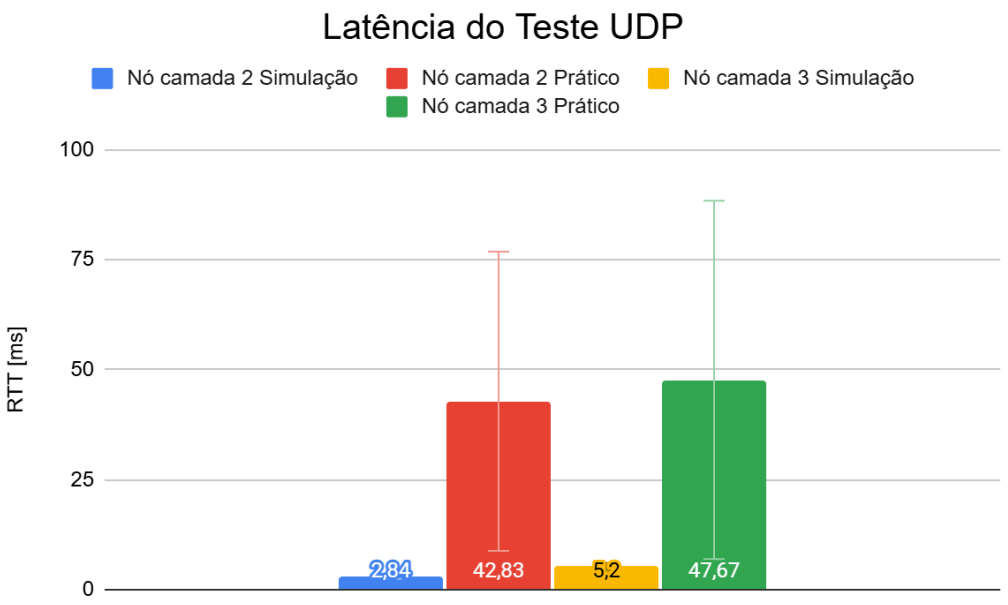
A taxa de dados *Uplink* comparativa entre os testes é mostrada no Gráfico 9. Observou-se que a simulação apresentou um desempenho pior com relação à prática. Estas diferenças na taxa de transferência estão associadas à aproximações realizadas pela simulação utilizando modelos teóricos. Porém, o ambiente real apresenta características dinâmicas e muitos detalhes que são desconsiderados para realização da simulação.

Os resultados de latência obtidos pelo teste prático com o protocolo UDP apresentaram valores muito mais altos do que os testes simulados. Além disso, o desvio das amostras foi também muito maior, demonstrando um grau de incerteza no tempo de transmissão. O Gráfico 10 mostra o resultado do RTT com o protocolo UDP para a simulação e a prática.

Novamente um aumento na taxa de transmissão de dados causou uma diferença maior entre o modelo simulado e a prática, provavelmente devido ao acúmulo de erros ao passo que é aumentado o número de pacotes transmitidos.

Além disso, configurações do modelo simulado, com relação ao ambiente físico, podem ter causado o acréscimo de erros e redução do *Throughput*.

Gráfico 10: Latência dos testes UDP.



Fonte: O autor (2025).

6. CONCLUSÃO

A simulação de uma rede Wi-Fi mesh foi realizada com o software OMNet++ utilizando quatro cenários diferentes variando em quantidade de estações *Mesh*, disposição no ambiente, protocolo de transporte e taxa de dados utilizada. As métricas analisadas foram a latência, *throughput* e perda de pacotes. Na segunda etapa do projeto, foi realizada a implementação da rede ESP-WIFI-MESH nos microcontroladores ESP32 e o desenvolvimento dos testes de acordo com os cenários simulados. Tanto os testes práticos quanto os simulados foram realizados no mesmo ambiente residencial, considerando a planta da casa e o material das paredes.

Os resultados obtidos mostraram que houve diferenças pequenas entre a simulação e a prática nas métricas de *throughput* e perda de pacotes. A maior taxa de transferência atingida para uma conexão foi de 692 kbps para o protocolo UDP no teste prático com uma banda HT40 do protocolo 802.11n na faixa de 2,4GHz e com taxa nominal de 1 Mbps, no cenário avaliado. A perda de pacotes neste caso ficou em torno de 74%, mostrando que o protocolo UDP não é eficiente neste cenário. Isto também indica que o limite de taxa de transferência de dados na rede ESP-WIFI-MESH para estas configurações foi próximo de 700 kbps. Apesar de ser um valor maior do que redes Thread, cujo limite teórico é de 250 kbps, ainda foi consideravelmente baixo se comparado com redes Wi-Fi 802.11n padrão, com limite teórico de 600 Mbps. A utilização do protocolo TCP com a rede Wi-Fi *Mesh* se mostrou eficiente, com perda de pacotes nula, como esperado, e *throughput* máximo de 367 kbps por nó.

Com relação à latência, foi observada uma diferença entre a simulação e a prática, com a última tendendo a valores maiores e também com maior dispersão dos valores medidos. Foi observada uma relação de proporcionalidade entre o *throughput* do teste prático e a latência medida por *ping*, sendo que quanto maior a taxa de dados maior foi a latência média, com uma possível causa sendo o *overhead* gerado por mais transmissões. Considerando apenas os testes práticos, o valor médio mínimo de latência observado foi de 5,5 ms, enquanto o máximo foi de 47,7 ms.

6.1 TRABALHOS FUTUROS

A implementação do protocolo Matter não foi realizada neste projeto pela falta de suporte da rede ESP-WIFI-MESH. No entanto, o trabalho de engenharia reversa da pilha Wi-Fi dos microcontroladores ESP32 (JASPER, 2025), possibilitará o desenvolvimento de uma rede 802.11s para estes microcontroladores e, consequentemente, a implementação do Matter em uma rede Wi-Fi *Mesh* com estes dispositivos.

Outra possibilidade de implementação do Matter em redes Wi-Fi *Mesh* é possível utilizando o projeto *open80211s*, que faz parte do Linux *Kernel*. Microprocessadores com sistema Linux embarcado como o Raspberry Pi são alternativas para implementação ao invés dos microcontroladores ESP32. Um ponto negativo da utilização de um sistema operacional Linux é a maior complexidade do chip (exige maior memória e processamento), além do maior consumo de energia.

Trabalhos futuros poderiam ainda explorar aplicações conjuntas do Matter e da rede Wi-Fi mesh em cenários distintos, estabelecendo comparações de desempenho com outros tipos de rede *Mesh*, como o Thread e o BLE.

7. BIBLIOGRAFIA UTILIZADA

BEEJ, B. H. **Beej's Guide to Network Programming v3.2.10**, 2025. Disponível em: <https://beej.us/guide/bgnet/html//index.html>. Acesso em: 17 jun. 2025.

CONNER, W. S.; KRUYS, J.; KIM, K. J.; ZUNIGA, J. C. IEEE 802.11s Tutorial, Overview of the Amendment for Wireless Local Area Mesh Networking. **IEEE 802 Plenary**, Dallas, 2006.

DESIMONE, R., BRITO, B. M., BASTON J. Model of indoor signal propagation using log-normal shadowing. **Long Island Systems, Applications and Technology**, Farmingdale, 2015.

ESPRESSIF SYSTEMS Co. **ESP-WIFI-MESH**. ESP-IDF Programming Guide, Shanghai, 2024. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/esp-wifi-mesh.html> . Acesso em: 07 dez. 2024.

ESPRESSIF SYSTEMS Co. **ESP-IDF LINUX INSTALLATION GUIDE.** Shanghai, 2025. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/linux-macos-setup.html#get-started-get-esp-idf>. Acesso em: 07 jun. 2025.

GROSSI,E. U.; GARMUS, G. P. **Desenvolvimento de um sistema de comunicação Thread para controle de iluminação**. Curitiba, 2023.

HIERTZ, G. R.; DENTENEER, D.; MAX, S.; TAORI, R.; CARDONA, J.; BERLEMANN, L.; WALKE, B. IEEE802.11s: The WLAN Mesh Standard. **IEEE Wireless Communications**, 2010.

INET FRAMEWORK. **User’s Guide**. Release 4.5.4. Disponível em: <https://inet.omnetpp.org/docs/users-guide/> . Acesso em: 28 nov. 2024.

JASPER. **esp32-open-mac**. Disponível em: <https://github.com/esp32-open-mac/esp32-open-mac>. Acesso em: 11/06/2025.

MEENA, P., JASMINDER, S. A Systematic Review of Quality of Service in Wireless Sensor Networks using Machine Learning: Recent Trend and Future Vision. **Journal of Network and Computer Applications**, Volume 188, 2021.

OPENSIM LTD. OMNeT++ Website. Disponível em: <https://omnetpp.org/>. Acesso em: 09/06/2025.

OWCZAREK, P.; ZWIERZYKOWSKI, P. Review of Simulators for Wireless Mesh Networks. **Journal of Telecommunications and Information Technology**, Poznan, 2014. Disponível em: <https://core.ac.uk/reader/235207400>. Acesso em: 28 nov. 2024.

PALUMBO, F.; BARSOCCHI, P.; BELLI, D. **Connectivity Standards Alliance Matter: State of the art and opportunities**. Internet of Things, vol. 27, Editora ELSEVIER, 2024. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2542660523003281>. Acesso em: 28 nov. 2024.

SCHENK, M. **Matter and the Web of Things**. Vienna, 2023.

SICHITIU, M. L. **Wireless Mesh Networks: Opportunities and Challenges**. Raleigh USA.

TEKTRONIX. **Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements**. 2016.

ZEGEYE, W.; JEMAL, A.; KORNEGAY, K. Connected Smart Home over Matter Protocol. **IEEE International Conference on Consumer Electronics**, Baltimore, MD, 2023.

APÊNDICE 1 - ESP-IDF E ESP-WIFI-MESH

Para programação dos microcontroladores e utilização da rede ESP-WIFI-MESH foram realizados os seguintes passos em um sistema operacional Ubuntu 18.04.6:

- **Passo 1:**

Para instalar o ESP-IDF foi utilizado o software Anaconda para Linux. O Anaconda permite a criação de um ambiente isolado dentro do sistema operacional. Isso é vantajoso, pois bibliotecas e outras dependências do ESP-IDF não se misturam com dependências de outros programas. Um guia de instalação e comandos básicos pode ser encontrado na web procurando por “Anaconda getting started”.

- **Passo 2:**

Foi criado um ambiente chamado “esp-idf” com o Anaconda. Neste ambiente foi realizada a instalação manual do ESP-IDF segundo o guia “*Standard Toolchain Setup for Linux and macOS*”. Os comandos no terminal são:

- Instalar dependências:

sudo apt-get install git wget flex bison gperf python3 python3-pip python3-venv cmake ninja-build ccache libffi-dev libssl-dev dfu-util libusb-1.0-0

- Obter ESP-IDF (v5.4.2):

mkdir -p ~/esp

cd ~/esp

git clone -b v5.4.2 --recursive https://github.com/espressif/esp-idf.git

- Instalar ferramentas para o chip ESP32:

cd ~/esp/esp-idf

./install.sh esp32

- **Passo 3:**

Para utilizar os comandos do ESP-IDF, ou seja, “idf.py”, o seguinte comando deve ser executado no terminal sempre que o ambiente for aberto:

. $HOME/esp/esp-idf/export.sh

- **Passo 4 (Utilizando o ESP-IDF)**:

Com isso, o ambiente de programação está instalado. A pasta “cd ~/esp” deve ser utilizada como “workspace” dos projetos. Aqui podem ser copiados projetos

prontos da pasta “esp-idf/examples”, que contém o exemplo “ip_internal_network” da rede ESP-WIFI-MESH.

Ao copiar um projeto para o diretório “/esp”, o usuário pode entrar no diretório do projeto copiado e compilar usando “idf.py build”.

Configurações do projeto podem ser alteradas utilizando “idf.py menuconfig”. Após alterar as configurações, é necessário compilar novamente.

Ao conectar um ESP32 ao computador via USB, o projeto compilado pode ser escrito na memória Flash do microcontrolador com “idf.py -p /dev/ttyUSB0 flash”.

A saída do microcontrolador pode ser observada com o comando “idf.py -p /dev/ttyUSB0 monitor”. Caso a saída não seja legível, aborte a operação com “Ctrl+]” e execute “idf.py -p /dev/ttyUSB0 monitor” novamente.

O código deste trabalho, referente aos testes práticos, pode ser acessado por completo em: <https://github.com/lorenzopicinin/TCC>.

APÊNDICE 2 - OMNET++

Foi utilizada neste trabalho a versão 6 do software OMNeT++ para ambiente Windows. No site oficial foi lançada a versão 6.1.0 em 10/09/2024, dísponível para múltiplos sistemas operacionais.

Um guia de instalação detalhado para cada sistema operacional pode ser encontrado no site como “OMNeT++ Installation Guide v6.1”. Neste guia há duas formas de instalação para Windows: Usando o compilador MinGW64 ou Subsistema Windows para Linux (WSL). Neste trabalho foi utilizado o compilador MinGW64.

O site oficial inclui documentação extensa de todas as funcionalidades do OMNeT++ e *Framework* INET. Além disso, tutoriais e vídeos explicativos podem ser encontrados na internet.