Solução de Deteção e Dispersão de Aves para Proteção de Colheitas - Parte 2

Projeto I

Daniel Reis Carvalho

Nº 20191904

Fábio Alexandre Barata Cardoso

Nº 20200701

Orientadores

João Manuel Leitão Pires Caldeira

Vasco Nuno da Gama de Jesus Soares

Trabalho de Projeto apresentado à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco para cumprimento dos requisitos necessários à obtenção do grau de Licenciado em Engenharia Informática, realizada sob a orientação científica do Doutor João Manuel Leitão Pires Caldeira, do Instituto Politécnico de Castelo Branco.

Outubro 2024

Composição do júri

Presidente do júri

Mestre, José Luís Silva Tavares da Cruz

Professor Adjunto do Instituto Politécnico de Castelo Branco

Vogais

Doutor, João Manuel Leitão Pires Caldeira

Professor Adjunto do Instituto Politécnico de Castelo Branco

Mestre, Paulo Alexandre Correia da Silva Neves

Professor Adjunto do Instituto Politécnico de Castelo Branco

Dedicatória

Daniel Carvalho

Este trabalho é dedicado aos meus pais, avós e restantes familiares. Agradeço profundamente pelo apoio constante e pelo incentivo inabalável que sempre me ofereceram. Aos meus pais, por me ensinarem o valor da educação e do esforço contínuo. Aos meus avós, por serem um exemplo de sabedoria e perseverança. E aos restantes familiares, pela compreensão e paciência durante esta jornada desafiadora. Sem vocês, este projeto não teria sido possível. Minha eterna gratidão a todos.

Fábio Cardoso

Dedico este projeto aos meus pais, ao meu irmão e às minhas avós.

Aos meus pais, pelo apoio incondicional às minhas decisões, mesmo quando não concordam com elas, e pelo imenso esforço que fazem diariamente para me proporcionarem uma vida melhor e todas as oportunidades possíveis, mesmo à custa de sacrifícios pessoais. A vossa dedicação e amor são a base do meu sucesso e crescimento. Ao meu irmão, por ser uma fonte constante de inspiração. Por me mostrar que nunca é tarde demais para alcançar os nossos objetivos e que, mesmo quando a vida não segue o rumo esperado, a verdadeira força está em saber levantar a cabeça e lutar. Agradeço-te por seres o meu modelo a seguir e por me motivares a ser sempre melhor.

Dedico ainda à minha avó Lurdes e, em memória, à minha avó Graça, que sempre insistiram para que eu estudasse e tivesse bons resultados, estando presentes e a apoiar-me até não poderem mais.

Agradecimentos

Expressamos o nosso agradecimento aos professores Vasco Nuno da Gama de Jesus Soares e João Manuel Leitão Pires Caldeira por toda a orientação e apoio técnico que nos ofereceram e pela infinita paciência demonstrada ao longo de todo o desenvolvimento deste projeto, ajudando-nos a manter o foco nos objetivos propostos.

Resumo

Os ataques das aves continuam a representar um dos maiores desafios enfrentados pelos agricultores, afetando diretamente a rentabilidade e sustentabilidade das explorações agrícolas. Com o avanço da agricultura inteligente, torna-se inevitável integrar tecnologias em um ambiente que, muitas vezes, carece das condições mínimas necessárias para a sua implementação eficaz. Um dos principais obstáculos nesse contexto é a acessibilidade à rede, devido à localização remota das explorações e à extensa cobertura requerida para monitorizar eficazmente as áreas agrícolas.

Uma solução promissora para superar essas dificuldades é a utilização de redes mesh, que permitem uma comunicação eficiente entre sensores distribuídos. Essas redes são projetadas para funcionar de forma robusta em ambientes rurais, oferecendo a flexibilidade necessária para se adaptar às particularidades do terreno. No entanto, as redes mesh também apresentam desafios próprios, como congestionamento, escalabilidade e latência, que exigem uma análise cuidadosa de soluções para otimizar seu desempenho. Neste sentido, pretende-se realizar essa análise com o objetivo de criar uma rede que mitigue ou elimine esses problemas, garantindo o envio bem-sucedido das mensagens necessárias para a dispersão dos pássaros. Ao abordar as questões de conectividade e eficiência, procuramos proporcionar uma solução eficaz para os agricultores, melhorando a sua capacidade de monitorizar e controlar as populações de aves, contribuindo assim para a sustentabilidade das suas explorações.

Palavras-chave

Dissuasão de Aves, Proteção de Colheitas, Agricultura Inteligente, Internet das Coisas, Rede *Mesh*, Sensores, Comunicação de Dados, Otimização de Rede, Avaliação de Desempenho

Abstract

Bird attacks continue to represent one of the biggest challenges facing farmers, directly affecting the profitability and sustainability of agricultural holdings. With the advance of smart agriculture, it is becoming inevitable to integrate technologies into an environment that often lacks the minimum conditions necessary for their effective implementation. One of the main obstacles in this context is network accessibility, due to the remote location of farms and the extensive coverage required to effectively monitor agricultural areas.

A promising solution to overcome these difficulties is the use of mesh networks, which allow efficient communication between distributed sensors. These networks are designed to work robustly in rural environments, offering the necessary flexibility to adapt to the particularities of the terrain. However, mesh networks also present their own challenges, such as congestion, scalability and latency, which require careful analysis of solutions to optimize their performance. In this sense, we intend to carry out this analysis with the aim of creating a network that mitigates or eliminates these problems, guaranteeing the successful sending of the messages necessary for bird dispersal. By addressing the issues of connectivity and efficiency, we aim to provide an effective solution for farmers, improving their ability to monitor and control bird populations, thus contributing to the sustainability of their farms.

Keywords

Bird Deterrence, Crop Protection, Smart Agriculture, Internet of Things, Mesh Network, Sensors, Data Communication, Network Optimization, Performance Evaluation

Índice geral

Índice de figuras

Lista de abreviaturas, siglas e acrónimos

AODV - Ad hoc On-Demand Distance Vector

ARQ - Automatic Repeat Request

CRC - Cyclic Redundancy Check

DSR - Dynamic Source Routing

FEC - Forward Error Correction

GPIO - General-purpose input/output

IDE - Integrated Development Environment

KB - Kilobytes

Kbps - Kilobits per second

MB - Megabytes

MHz - Megahertz

MP3 - MPEG Audio Layer 3

NC - Network Coding

OR - Opportunistic Routing

PPR - Partial Packet Recovery

RREP - Route Reply

RREQ - Route Request Packets

SRAM - Static Random-Access Memory

1 Introdução

A dispersão de pássaros nas colheitas representa um desafio significativo para a rentabilidade das explorações agrícolas. Os métodos tradicionais utilizados atualmente para afastar aves frequentemente revelam-se pouco eficientes, levando à necessidade de soluções mais inovadoras e tecnológicas. Nos últimos anos, têm surgido abordagens que incorporam tecnologias de monitorização e comunicação, prometendo melhorar a eficácia na gestão das populações de pássaros que atacam as culturas. Contudo, a implementação destas tecnologias em ambientes rurais apresenta desafios consideráveis. Em zonas remotas, onde a conectividade é limitada, a adoção da agricultura inteligente, que envolve o uso de sensores e dispositivos conectados para monitorizar o solo, as culturas e o ambiente, exige uma infraestrutura de rede que assegure a transmissão eficaz de dados em tempo real. No entanto, as características geográficas e a extensão das áreas agrícolas frequentemente dificultam o acesso a redes convencionais, tornando a comunicação entre dispositivos um dos principais obstáculos à sua implementação.

As redes *mesh*, caracterizadas pela sua capacidade de auto-organização e distribuição dinâmica de dados entre vários nós, surgem como uma solução promissora para combater as limitações de conectividade. Estas redes permitem a criação de uma infraestrutura descentralizada, onde cada nó da rede pode atuar como um ponto de retransmissão, garantindo que os dados percorrem distâncias longas. Contudo, a utilização de tecnologias de monitorização para a dispersão de pássaros em colheitas apresenta uma problemática específica: os ataques das aves podem causar prejuízos significativos nas explorações agrícolas, afetando diretamente a rentabilidade e a sustentabilidade das culturas.

A dispersão eficaz das aves requer uma comunicação rápida e confiável entre os dispositivos de monitorização e os mecanismos de controle. No entanto, apesar do potencial das redes mesh, estas enfrentam vários desafios, sendo os problemas mais críticos o congestionamento da rede, causado pelo tráfego de dados entre os nós, e a latência, que afeta o tempo de resposta da rede. Assim, é fundamental desenvolver soluções que garantam a transmissão eficiente de mensagens que acionem os dispositivos de dispersão de pássaros, assegurando que a agricultura inteligente funcione de maneira eficaz, mesmo em condições desafiadoras.

1.1 Âmbito

Este projeto tem como objetivo desenvolver uma solução tecnológica que permita detetar a presença de aves, junto de árvores de um pomar, e encaminhar essa informação para uma estação base ligada a um *drone*. Considerando este cenário remoto (pomar) e a dificuldade inerente no acesso a recursos tecnológicos, nomeadamente cobertura de rede, é proposto neste trabalho o desenvolvimento de uma *rede mesh* otimizada que assegure a entrega rápida e eficiente dos dados de nós distribuídos a um nó central. Pretende-se garantir não só a entrega dos dados, mas também tempo de envio reduzido, abordando desafios como a latência e o congestionamento, de modo a otimizar a comunicação entre dispositivos. Serão analisados métodos e protocolos que contribuam para a criação de uma rede robusta e efetuados testes de maneira a averiguar qual a melhor solução.

1.2 Definição do Problema e Objetivos

O projeto enfrenta o desafio de implementar uma *rede mesh* eficiente em áreas rurais que, devido às suas características, como a dispersão dos dispositivos e a ausência de uma infraestrutura adequada, tornam difícil garantir a transmissão fiável e em tempo útil dos dados recolhidos. Além disso, problemas técnicos como o congestionamento da rede, a latência elevada e a limitação na escalabilidade comprometem o desempenho da rede e, consequentemente, a eficácia das soluções de agricultura inteligente.

Neste contexto, o objetivo principal deste trabalho é desenvolver uma infraestrutura de *rede mesh* otimizada, capaz de garantir a comunicação eficiente entre os sensores, com tempos de resposta reduzidos. Para alcançar esse objetivo, será necessário identificar os principais problemas que afetam a implementação da rede, analisar os métodos e protocolos disponíveis que possam mitigar esses desafios, e implementar soluções que assegurem a fiabilidade, a escalabilidade e o desempenho adequado da rede.

1.3 Cronograma das Tarefas

Para atingir os objetivos acima identificados, prevê-se o seguinte plano de trabalhos e cronograma:

Tarefa 1 – Estudo de Protocolos: Análise dos diversos protocolos de rede existente. Esta tarefa envolve a pesquisa aprofundada sobre *redes mesh*, metodologias

de implementação e otimização, bem como a identificação de ferramentas e tecnologias relevantes para a segunda fase do projeto.

Tarefa 2 – Proposta e Implementação do Protótipo Demonstrador: Desenvolvimento do protótipo inicial da *rede mesh*, incluindo a configuração básica dos nós. Durante esta tarefa, serão implementadas as funcionalidades principais, como comunicação entre nós e gestão de tráfego, seguidas de testes preliminares para verificar a configuração e o desempenho inicial da rede.

Tarefa 3 – Demonstração e Testes Funcionais: Realização de testes avançados para avaliar o comportamento da rede sob diferentes cenários. Nesta etapa, serão feitos ajustes na configuração e otimização da rede com base nos resultados dos testes. Além disso, será realizada a integração com sistemas de captura de áudio, caso aplicável, para validar a comunicação na *rede mesh*.

Tarefa 4 – Avaliação de Desempenho: Avaliação detalhada do protótipo utilizando métricas de desempenho, como latência, taxa de perda de pacotes, e escalabilidade. A análise crítica dos resultados obtidos será realizada, comparando-os com os objetivos definidos inicialmente, e identificando áreas de melhoria e recomendações para futuras iterações.

Tarefa 5 – Escrita do Relatório: Preparação e redação do relatório final do projeto, documentando todo o processo de desenvolvimento e as conclusões alcançadas. Esta tarefa também inclui a revisão do relatório, incorporação de *feedback* e ajustes finais, além da preparação para a apresentação do projeto e submissão do relatório final.

	Mês 1	Mês 2	Mês 3	Mês 4	Mês 5
Tarefa 1					
Tarefa 2					
Tarefa 3					
Tarefa 4					
Tarefa 5					

Tabela 1 - Cronograma de tarefas

1.4 Contribuições

As principais contribuições deste trabalho são: (1) analise de protocolos utilizados para otimização de redes; (2) Proposta de implementação de uma solução (3) implementação e avaliação de desempenho da solução;

1.5 Organização do Documento

Este documento está organizado em seis capítulos. O primeiro capítulo consiste na introdução ao projeto, incluindo uma contextualização do problema, definição dos objetivos e detalhes sobre a planificação. O segundo capítulo apresenta uma descrição detalhada da rede mesh, abordando os problemas que serão considerados. O terceiro capítulo foca na investigação de métodos para otimizar a rede mesh, com uma análise detalhada de cada abordagem e as suas potenciais vantagens e desvantagens. O quarto capítulo descreve a proposta de implementação dos métodos investigados, incluindo a configuração, a integração e os testes preliminares realizados. O quinto capítulo é dedicado à realização de testes da rede, onde se avaliam os tempos de latência, a fiabilidade da comunicação, bem como a resposta da rede a diferentes cenários, incluindo sobrecarga e falhas em nós intermédios. Finalmente, o sexto capítulo apresenta as conclusões do estudo, juntamente com um conjunto de sugestões para trabalhos futuros.

2 Características das Redes *Mesh*

Uma rede *mesh* é uma topologia de rede na qual cada dispositivo, ou nó, atua como um ponto de retransmissão para os dados, criando uma estrutura de comunicação interconectada [1]. Ao invés de depender de um único ponto central, cada nó na rede *mesh* comunica diretamente com outros nós, formando uma rede descentralizada [2]. Os nós na rede *mesh* podem comunicar diretamente entre si ou através de intermediários, o que resulta numa comunicação auto-organizada e com capacidade de se adaptar [3]. Quando um nó falha ou se torna inativo, os dados podem ser redirecionados por outros caminhos disponíveis na malha, garantindo a continuidade da transmissão e minimizando a interrupção do serviço [3]. Além disso, a *rede mesh* é projetada para se ajustar dinamicamente a mudanças na topologia da rede, como a adição ou remoção de nós [4]. Esta abordagem descentralizada é particularmente benéfica em ambientes extensos e de difícil acesso onde a cobertura e a robustez da rede são essenciais para garantir a transmissão eficaz de dados [4]. A capacidade de criar múltiplos caminhos de comunicação contribui para uma maior fiabilidade e resiliência da rede, proporcionando uma solução eficaz para a transmissão de dados em tempo real [3].

2.1 Problemas Associados

Tendo em consideração o cenário de aplicação (um pomar remoto) desta solução tecnológica, um dos principais desafios é assegurar que todas as mensagens enviadas cheguem ao destino pretendido não haver uma perda de mensagens que pode ocorrer devido a falhas nos nós da rede, interferências ambientais ou problemas de transmissão [5]. Para minimizar esse risco, é crucial adotar mecanismos de retransmissão e protocolos de confirmação de receção das mensagens [6]. Estes métodos garantem que cada mensagem seja, não apenas enviada, mas também recebida e reconhecida pelo nó central, assegurando a integridade da comunicação [7].

Outro problema altamente relevante é a chegada tardia das mensagens, onde os dados chegam ao nó central fora de um intervalo temporal aceitável. Um tempo de envio longo compromete a eficácia do sistema em tempo real que é essencial. Para enfrentar este desafio, é necessário otimizar os protocolos de *routing [8]* e adotar técnicas de priorização de tráfego de maneira a garantir que as mensagens chegam num intervalo de tempo aceitável [9].

A ocorrência de falhas na transmissão de mensagens é um desafio significativo que pode surgir devido a erros ou interferências no processo de comunicação [10]. Esses problemas podem comprometer a integridade dos dados, tornando fundamental a implementação de soluções que consigam identificar e corrigir tais incidentes [10]. Para garantir a precisão das informações transmitidas, é fundamental utilizar métodos de verificação, como somas de verificação e códigos de correção de erros [11], que asseguram que qualquer mensagem afetada seja detetada e reparada antes de ser processada, evitando assim que dados incorretos sejam considerados para processamento [11].

No próximo capítulo, serão exploradas e discutidas diversas estratégias e protocolos que visam resolver ou mitigar esses problemas. Serão abordados métodos específicos para otimizar o desempenho da *rede mesh*, incluindo técnicas de *routing* avançadas e soluções para melhorar a fiabilidade e a rapidez na transmissão de dados.

3 Abordagens e Métodos para Otimização

Após a identificação e análise dos principais problemas enfrentados pelas redes *mesh*, como a perda e corrupção de mensagens e a latência elevada, o próximo passo é abordar soluções práticas para superar esses desafios. Neste capítulo, serão examinadas as abordagens e métodos destinados a otimizar o desempenho das redes *mesh*. O foco será em estratégias para melhorar a eficiência do *routing*, garantir a integridade dos dados e reduzir o tempo de entrega das mensagens.

3.1 Partial Packet Recovery (PPR)

O *Partial Packet Recovery* é uma técnica avançada de recuperação de pacotes desenvolvida para enfrentar as perdas comuns em redes sem fio, particularmente em cenários com comunicações distribuídas, como redes *mesh*. Tradicionalmente, quando um pacote é corrompido durante a transmissão, é necessária a retransmissão do pacote inteiro. No entanto, o *PPR* tem como objetivo otimizar esse processo, ao permitir a recuperação dos dados que foram corrompidos, reduzindo a necessidade de retransmissões totais. O funcionamento do *PPR* baseia-se na capacidade de recuperar e reconstituir partes de pacotes que foram corrompidos durante a transmissão. Quando um pacote é enviado através da rede, ele pode ser dividido em fragmentos ou blocos menores. O dispositivo remetente atribuiu um código *Cyclic Redundancy Check (CRC)* a cada fragmento de maneira que se pacote sofrer erros de transmissão, em vez de solicitar uma retransmissão total, o recetor usa o código *CRC* para identificar quais os fragmentos específicos que estão intactos e quais foram corrompidos e retransmite apenas os fragmentos que falharam na verificação. [12]

No contexto do nosso projeto, o *PPR* pode ser implementado para melhorar a robustez da nossa rede *mesh*. Como os sensores estão distribuídos numa área rural, a recuperação parcial de pacotes evita a sobrecarga de transmissões repetidas, melhorando tanto a latência quanto a eficiência energética. Além disso, a técnica reduz a ocupação de largura de banda ao permitir que apenas partes dos pacotes corrompidos sejam reenviadas, o que é fundamental para evitar o congestionamento da rede.

De acordo com os artigos [12] e [13] o protocolo *PPR* pode ainda ser combinado com estratégias como o *Forward Error Correction (FEC)*, Network *Coding (NC)* e

Opportunistic routing (OR) de maneira a melhorar não só a eficácia dos protocolos envolvidos mas também a da rede *mesh* em geral.

3.2 Network Coding (NC)

O *Network Coding* é uma técnica inovadora que melhora a eficiência e a robustez das redes de comunicação através da combinação de pacotes de dados durante a transmissão. Em vez de enviar pacotes individualmente e separadamente, o *NC* permite que múltiplos pacotes sejam combinados em um único pacote codificado antes de serem retransmitidos pela rede. Este método transforma os pacotes de dados em uma forma codificada que pode carregar informações de diversos pacotes originais simultaneamente. [14], [15], [16] A principal vantagem do *NC* é a redução do número total de transmissões necessárias para garantir a entrega dos dados. Em redes *mesh*, onde a comunicação entre vários nós pode ser complexa e sujeita a falhas, a combinação de pacotes permite uma utilização mais eficiente da largura de banda disponível. Isso é alcançado porque menos pacotes precisam ser retransmitidos, o que ajuda a mitigar o congestionamento e reduz a sobrecarga associada a retransmissões.[17], [18] Além disso, o *NC* aumenta a robustez da rede contra perdas de pacotes ao combinar dados de múltiplos pacotes porque a técnica proporciona uma forma de recuperação de informações mesmo quando alguns pacotes são perdidos ou corrompidos durante a transmissão o que resulta numa comunicação mais confiável e resiliente. Embora o *NC* ofereça benefícios significativos também introduz desafios específicos. Como o *NC* combina pacotes antes de enviá-los, é essencial assegurar que a sequência dos pacotes é mantida para minimizar o tempo entre a captura do som e o recebimento da resposta. Para resolver esse desafio, seria necessário implementar um método de verificação da ordem dos pacotes que poderia ser tratada através da introdução de um sistema de prioridade ou a utilização de *timestamps* para marcar a sequência dos pacotes.

Portanto, enquanto o *Network Coding* oferece uma solução promissora para melhorar a comunicação na rede *mesh*, é crucial considerar essas *caveats* e implementar soluções adequadas para garantir a integridade da ordem dos pacotes e a eficácia da comunicação.

3.3 Forward Error Correction (*FEC*)

O *Forward Error Correction* é uma técnica de correção de erros usada para garantir a integridade dos dados transmitidos que, ao contrário de outras abordagens que exigem retransmissão de pacotes quando erros são detetados, trabalha de forma proativa, adicionando dados de redundância ao fluxo da informação que permitem que o destinatário detete e corrija erros sem a necessidade de pedir a retransmissão dos pacotes danificados. A principal vantagem do *FEC* em redes *mesh*, especialmente no nosso projeto, é a redução da necessidade de retransmissões. Apesar do *PPR* reduzir a quantidade de dados a serem retransmitidos é ainda necessário retransmitir o que implica uma carga adicional na rede e uma extensão do tempo geral enquanto o *FEC* garante que as mensagens chegam ao destino com maior fiabilidade, mesmo quando ocorrem erros no caminho. Isso resulta em uma melhoria da eficiência da rede e redução da latência, pois a correção de erros é feita automaticamente sem o atraso introduzido pelas solicitações de retransmissão. [19] No entanto, o uso do *FEC* também traz algumas desvantagens, como o aumento da sobrecarga de dados transmitidos devido às informações de correção de erros adicionais. Esse fator pode afetar negativamente o desempenho da rede se não for corretamente equilibrado com os benefícios que proporciona. Portanto a implementação do *FEC* deve ser cuidadosamente pensada antes de ser implementada para maximizar a sua eficiência, garantindo que a carga maior na rede devido ao aumento de dados necessários para a correção não compromete o benefício do tempo reduzido.

3.4 Opportunistic routing (OR)

O *Opportunistic Routing* é uma técnica de encaminhamento que otimiza a transmissão de dados em redes *mesh*, aproveitando a diversidade de caminhos disponíveis para melhorar a eficiência da rede que ao invés de selecionar antecipadamente um único caminho para transmitir dados, utiliza a flexibilidade dos nós na rede para escolher dinamicamente o melhor caminho durante a transmissão. Quando um nó envia um pacote, vários nós candidatos podem recebê-lo e o mais adequado para retransmitir o pacote é escolhido com base na qualidade do *link* ou da proximidade ao destino. Esta decisão dinâmica permite que a rede se adapte em tempo real às condições de conectividade, o que aumenta significativamente a eficiência. No protocolo descrito no artigo [20], o *OR* foi implementado para otimizar a transmissão

numa rede com recursos limitados que é o cenário do nosso projeto. O protocolo escolhe inteligentemente nós de retransmissão com base nas condições momentâneas da conectividade, garantindo que os pacotes sigam o caminho mais eficiente até o destino.

Em redes *mesh*, o *Opportunistic Routing* oferece várias vantagens como a minimização do risco de falhas de transmissão ao permitir que múltiplos nós recebam o pacote. Se um nó não conseguir retransmitir o pacote, outro nó próximo pode assumir essa função, garantindo que os dados continuem a ser transmitidos assim como aumento da eficiência. Finalmente, o *OR* é altamente flexível, adaptando-se facilmente às mudanças de topologia e à variabilidade da qualidade dos *links*, tornando-o ideal para cenários de alta mobilidade e redes em que a conectividade é inconsistente.

3.5 Ad hoc On-Demand Distance Vector (AODV)

O *Ad hoc On-Demand Distance Vector* é um protocolo de *routing* utilizado em redes *ad hoc*, incluindo redes *mesh*, que calcula rotas sob demanda, ou seja, quando uma fonte precisa enviar dados para um destino. Ao contrário do que é o padrão, o *AODV* só procura e estabelece uma rota quando necessário, minimizando o tráfego de controle na rede [21]. O funcionamento do *AODV* baseia-se em dois processos principais: a descoberta de rota e a manutenção de rota. Quando um nó precisa de enviar dados para outro nó mas não conhece o percurso, ele inicia um processo de descoberta de caminho através de mensagens *Route Request Packets (RREQ)* que são transmitidas para os vizinhos. Esses nós encaminham a solicitação até que o destino seja alcançado ou um nó que conheça o caminho necessário seja encontrado. Em resposta, uma mensagem *Route Reply (RREP)* é enviada de volta ao nó de origem [22]. A manutenção dos caminhos é feita monitorizando continuamente a conectividade com os nós vizinhos. Se uma rota falhar, o *AODV* pode reiniciar o processo de descoberta [23].

No contexto da rede *mesh* do nosso projeto, o *AODV* pode ser útil para reduzir a sobrecarga de tráfego de controlo, já que as rotas só são estabelecidas conforme a necessidade, o que é importante para economizar recursos numa rede distribuída com nós de baixo consumo. No entanto, o *AODV* apresenta alguns desafios, especialmente em redes com alta densidade de nós ou tráfego intenso, o que pode resultar em congestionamento durante a fase de descoberta de rotas, uma vez que muitas solicitações podem ser geradas simultaneamente [22]. Além disso, se houver falhas

frequentes nos *links*, o tempo de recuperação pode ser elevado, já que o protocolo necessita reiniciar o processo de descoberta de rotas, o que pode aumentar a latência e afetar o desempenho da nossa rede *mesh [23]*.

3.6 Dynamic Source Routing (DSR)

O *Dynamic Source Routing* é outro protocolo de *routing* reativo projetado para redes *ad hoc*, que opera de maneira similar ao *AODV*, mas com uma abordagem baseada em fontes. No *DSR*, quando um nó precisa de enviar dados, ele descobre uma rota completa até o destino e armazena essa rota no cabeçalho de cada pacote [24]. Os pacotes são então encaminhados ao longo dessa rota predefinida. O *DSR* usa os dois mecanismos mencionados anteriormente no *AODV* sendo a principal diferença entre os dois protocolos o fato do *DSR* armazenar a rota completa nos pacotes, o que elimina a necessidade de armazenar as rotas em cada nó ao longo do caminho [25].

No nosso projeto, o *DSR* pode ser vantajoso porque não requer tabelas de *routing* extensas, o que é útil em dispositivos com pouca memória, como os sensores distribuídos na rede *mesh*. Além disso, o fato do *DSR* permitir múltiplas rotas alternativas para um mesmo destino pode aumentar a robustez da rede ao fornecer redundância caso um caminho falhe [26]. Por outro lado, como as rotas completas são armazenadas no cabeçalho dos pacotes isso pode aumentar significativamente o tamanho dos pacotes, especialmente em redes de grande dimensão, onde a rota pode atravessar vários nós [27]. Isso pode causar maior latência e sobrecarga de processamento, além de consumir mais energia. Em redes *mesh* de grande escala, essa sobrecarga pode prejudicar a eficiência da rede, comprometendo o desempenho geral.

3.7 Automatic Repeat reQuest (ARQ)

O *Automatic Repeat reQuest* é um protocolo de controlo de erro que se baseia na retransmissão de pacotes para garantir a integridade dos dados. No *ARQ*, o nó recetor verifica a integridade dos pacotes recebidos usando um código de verificação (geralmente, o *CRC*) e, se o pacote estiver correto, o recetor envia um *acknowledgement (ACK)* ao remetente. Caso o pacote esteja corrompido ou se o *ACK* não for recebido dentro de um determinado intervalo de tempo, o remetente retransmite o pacote até que ele seja corretamente recebido ou um limite de tentativas seja atingido [28]. Existem três variações principais que são o *Stop-and-Wait ARQ, Go-Back-N ARQ* e o Selective *Repeat ARQ*. O *Stop-and-Wait* é o mais simples, onde o remetente envia um

pacote e espera pela confirmação antes de enviar o próximo enquanto o *Go-Back-N* permite que o remetente envie múltiplos pacotes sem esperar por confirmações mas retransmite uma sequência de pacotes se um erro for detetado[29]. O *Selective Repeat* é mais eficiente, pois apenas retransmite os pacotes com erro diminuindo a sobrecarga na rede causada pelo reenvio de pacotes desnecessários [30].

Na nossa rede *mesh*, o *ARQ* poderia ajudar a melhorar a fiabilidade da transmissão de dados, garantindo que pacotes perdidos ou corrompidos sejam retransmitidos automaticamente [31]. O *ARQ* também pode complementar outras técnicas, como o *PPR* e o *FEC*, fornecendo um mecanismo adicional de correção de erros. Porém, a desvantagem do *ARQ* é o impacto na latência, já que a retransmissão contínua de pacotes pode introduzir atrasos significativos, especialmente se os erros forem frequentes [32]. Além disso, em redes com tráfego intenso ou com um grande número de nós, o número de retransmissões pode congestionar a rede [29]. Por isso, o *ARQ* deve ser usado com moderação, possivelmente em conjunto com outras técnicas de correção de erros para equilibrar as vantagens e desvantagens que oferece.

4 Desenvolvimento

Neste capítulo, vamos implementar e testar os protocolos descritos no capítulo 3 para determinar quais são mais eficazes para a nossa solução. A implementação será seguida por uma série de testes para avaliar o desempenho de cada protocolo em termos de eficiência, latência e taxa de retransmissões com o objetivo de identificar a solução que melhor atende às necessidades da nossa rede.

4.1 Aquisição de áudio

Para efeitos de validação da solução a propor, torna-se necessária a obtenção de áudio a analisar, para deteção e identificação de aves. Sendo que, nesta fase do desenvolvimento o foco estava na otimização do desempenho da infraestrutura de rede e por questões relativas às contingências inerentes à aquisição real desses áudios, optou-se, nesta fase, por usar um áudio pré-gravado. Para realizar os testes, foi necessário selecionar um áudio base. Entre as espécies mais conhecidas por atacar plantações agrícolas, visíveis na Figura 1, optámos pela *Pica pica*, popularmente chamada de Pega, devido à sua presença significativa na região de Castelo Branco, confirmada através do website *BioDiversityForAll*, como mostrado na Figura 2.

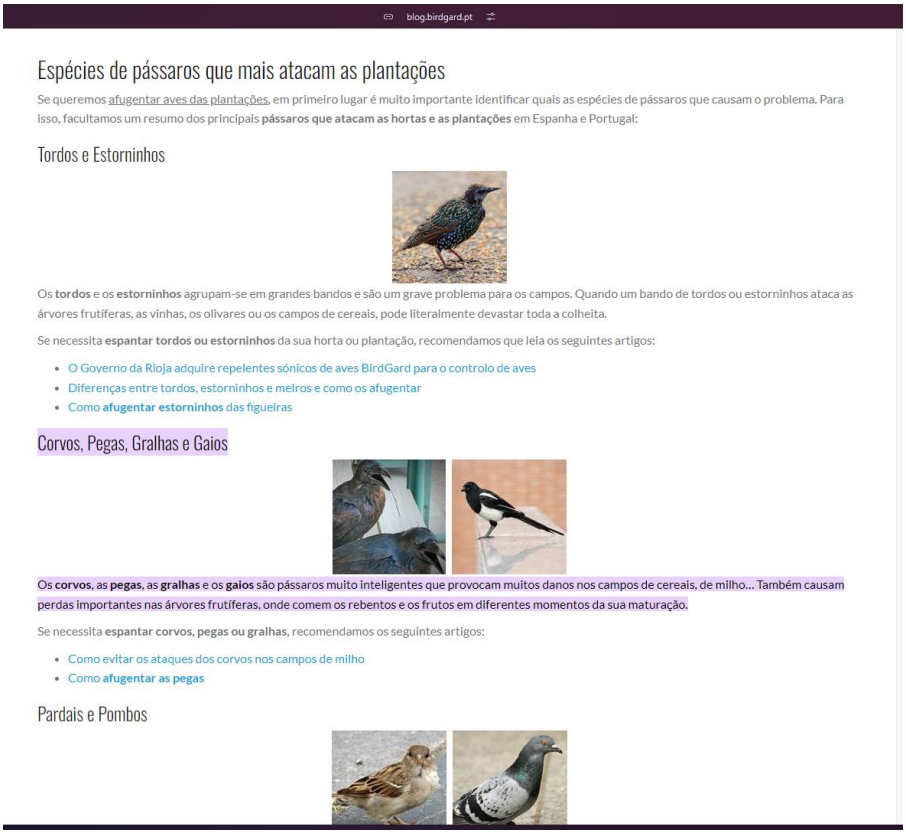

Figura 1 - Espécies que mais atacam as plantações Fonte: [33]

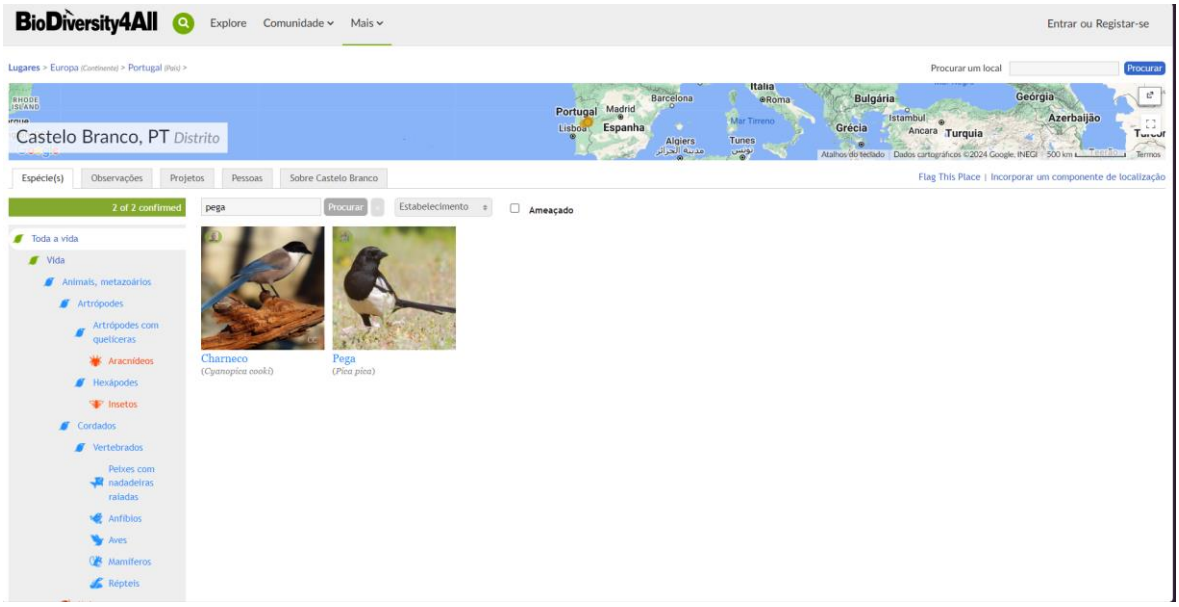

Figura 2 - Presença da espécie Pega no distrito de Castelo Branco Fonte: [34]

Utilizando a base de dados do *eBird*, que é a fonte de dados empregada pelo software *BirdNET*, realizámos uma pesquisa focada na espécie *Pica pica*, visível na Figura 3. Através do filtro geográfico, visível na Figura 4, disponível no *eBird* limitamos a nossa pesquisa para a região de Castelo Branco, o que nos possibilitou avaliar a presença da espécie nesta área especifica.

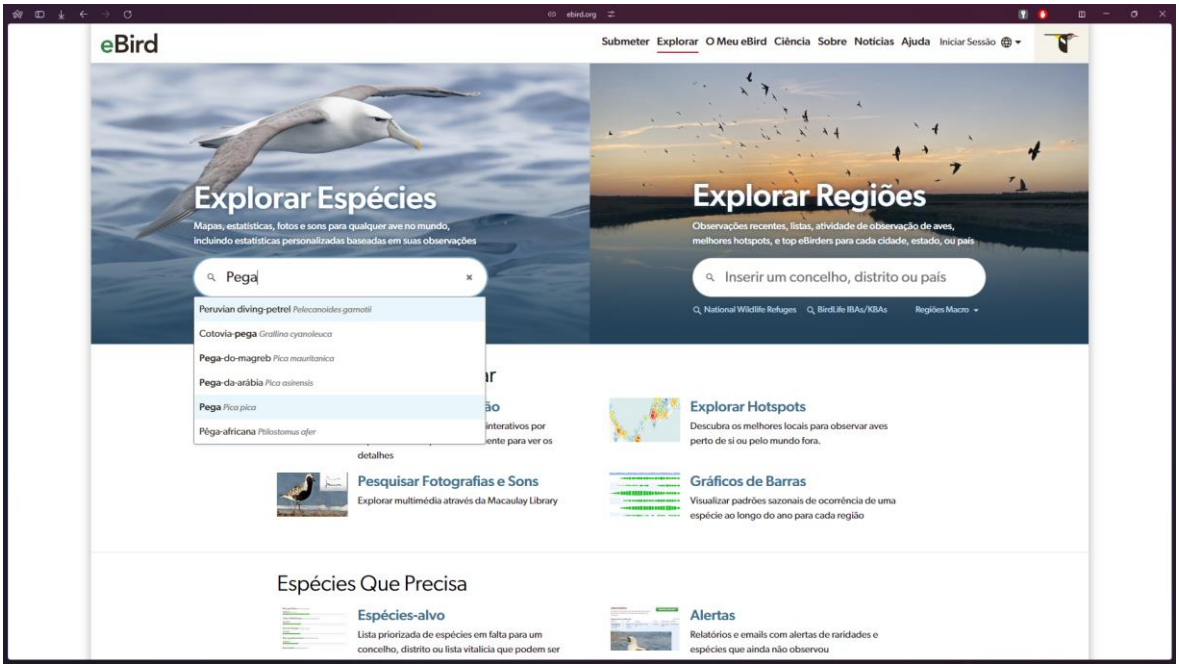

Figura 3 - Procura no eBird pela espécie Pega Fonte: [35]

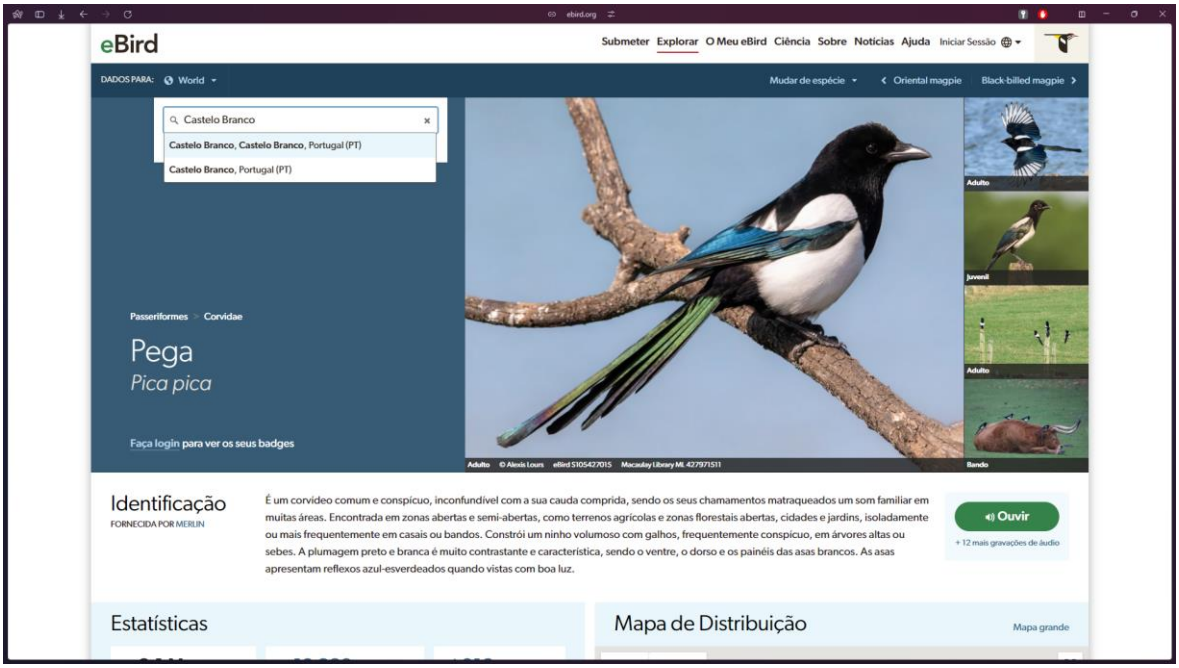

Figura 4 - Filtro pela região geográfica Fonte: [36]

Através da análise dos dados obtidos, verificámos que existem mais de 2000 observações da espécie Pega em Castelo Branco, acompanhadas de algumas imagens, como ilustrado na Figura 5. No entanto, apenas um áudio estava disponível para análise.

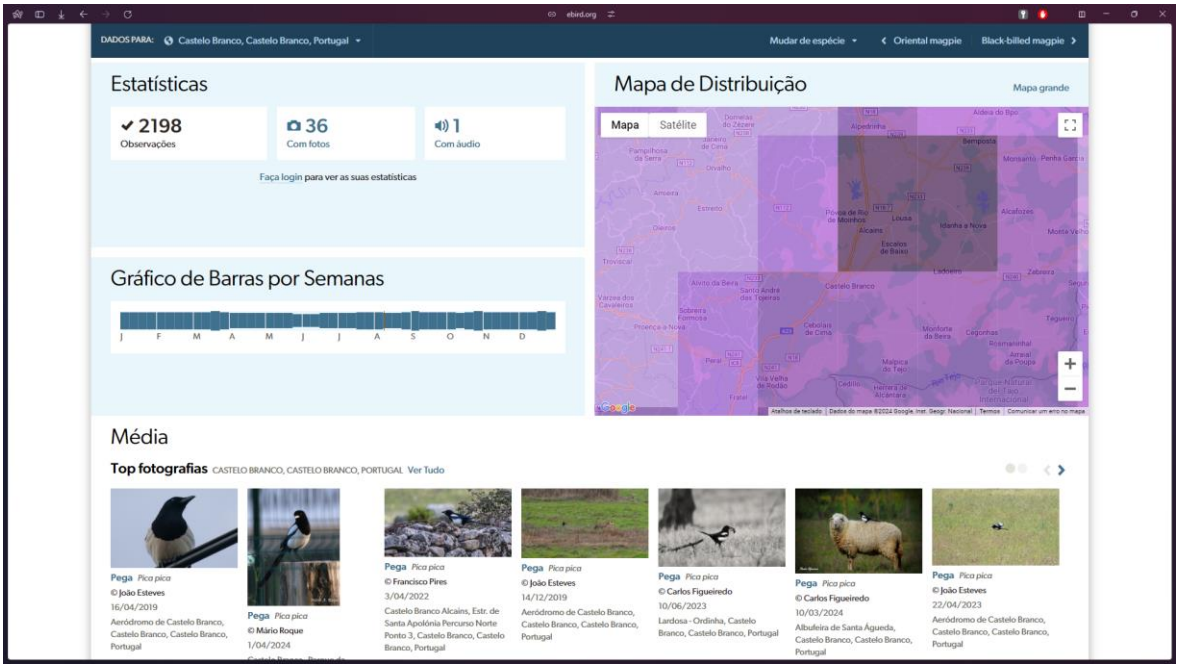

Figura 5 - Dados sobre a deteção da espécie Pega em Castelo Branco Fonte: [36]

Este áudio, visível na Figura 6, foi registado no dia 29 de abril de 2021 e recebeu uma classificação de 4 estrelas, atribuída com base em três avaliações de utilizadores. Apesar de algumas interferências, como o vento, o som da espécie é claramente audível.

Ao analisar os detalhes do ficheiro podemos observar que o mesmo é um ficheiro *MP3* com uma duração de 16 segundos e com tamanho de 501*Kb* com uma velocidade de transmissão de 256*kbps*.

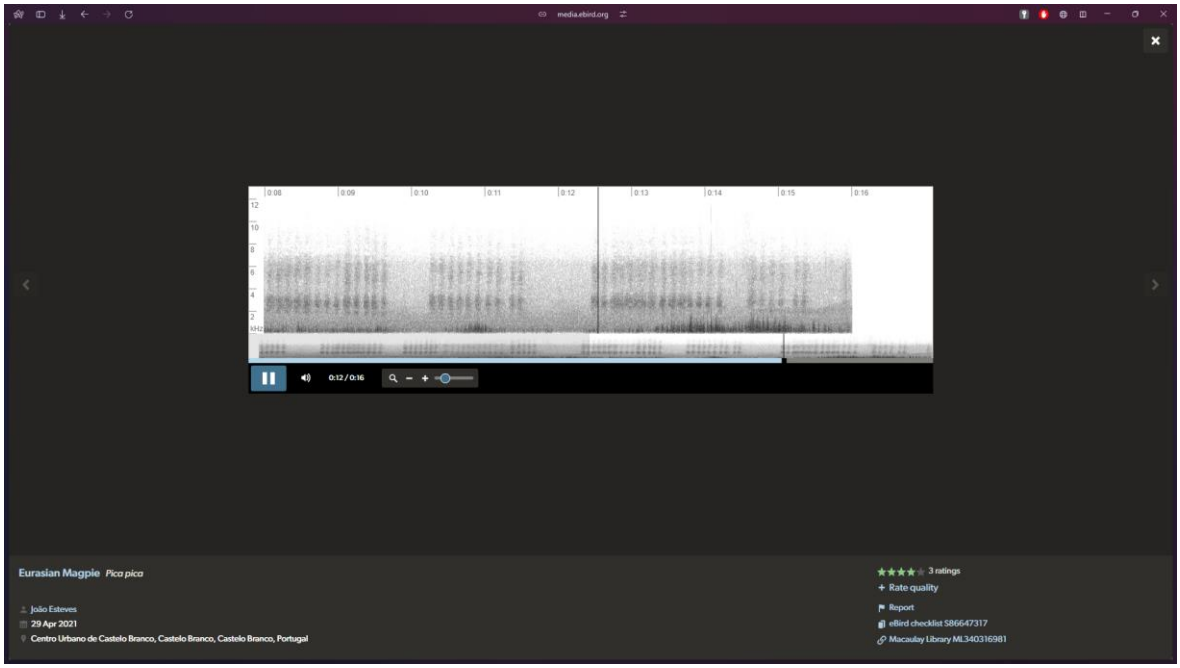

Figura 6 – Ficheiro de áudio da espécie Pega em Castelo Branco Fonte: [37]

Realizámos um teste utilizando este áudio para avaliar a capacidade de deteção da espécie pelo *BirdNET*. Os resultados do teste, visíveis na Figura 7, revelou a eficácia do sistema na identificação da Pega, mesmo na presença de ruídos ambientais, tendo detetado com uma confiança de entre 97.25 % a 99.49 %.

Selection	View	Channel	Begin File	Begin Time (s)	End Time (s)	Low Freq (Hz)	High Freq (Hz)	Species
1	Spectrogram 1	1	audio.mp3	0	3.0	0	15000	eurmag1 Eurasian Magpie 0.9725	
2	Spectrogram 1	1	audio.mp3	3.0	6.0	0	15000	eurmag1 Eurasian Magpie 0.9805	
3	Spectrogram 1	1	audio.mp3	6.0	9.0	0	15000	eurmag1 Eurasian Magpie 0.9502	
4	Spectrogram 1	1	audio.mp3	9.0	12.0	0	15000	eurmag1 Eurasian Magpie 0.9949	
5	Spectrogram 1	1	audio.mp3	12.0	15.0	0	15000	eurmag1 Eurasian Magpie 0.8750	
6	Spectrogram 1	1	audio.mp3	12.0	15.0	0	15000	bkbmag1 Black-billed Magpie	0.1131
7	Spectrogram 1	1	audio.mp3	15.0	18.0	0	15000	eurmag1 Eurasian Magpie 0.3513	
8	Spectrogram 1	1	audio.mp3	15.0	18.0	0	15000	bkbmag1 Black-billed Magpie	0.1578

Figura 7 - Resultados da análise do ficheiro da espécie Pega em Castelo Branco

Tendo agora adquirido um áudio no qual a deteção da espécie foi confirmada com um nível elevado de confiança, é possível avançar para a próxima etapa onde iremos preparar o cenário para a implementação e testes subsequentes. Esta etapa seguiu inicialmente uma abordagem (Abordagem 1) que foi perspetivada como a mais

promissora. No entanto, ao longo da sua implementação, surgiram algumas dificuldades que impediram atingir o objetivo proposto em tempo útil. Dessa forma, e na esperança de poder ser atingido esse objetivo, foi decidido seguir uma segunda abordagem (Abordagem 2) em resposta às dificuldades encontrada no desenvolvimento da anterior. Nas secções seguintes é descrito todo o desenvolvimento das soluções propostas para as duas abordagens referidas.

4.2 Abordagem 1

Nesta abordagem, os nós distribuídos são responsáveis pela captura dos ficheiros de áudio no ambiente. Estes ficheiros são transmitidos através da rede *mesh* para um nó ponte, que faz a ligação entre a rede *mesh* e o nó central. O nó central, com maior capacidade de processamento, recebe os áudios e realiza a análise para identificar as espécies presentes. Com base nessa análise, são determinadas as ações a tomar, como a dispersão de pássaros.

Para o desenvolvimento da nossa solução, nesta primeira abordagem, optámos por utilizar os módulos *D1 Mini Pro [38]* como nós distribuídos, sendo que um dos módulos foi designado para atuar como nó ponte, conforme ilustrado na Figura 8. O nó ponte desempenha um papel crucial servindo como a ligação entre a rede *mesh* e nó central que permite a comunicação entre os nós distribuídos e o nó central.

O *D1 Mini Pro*, baseado no microcontrolador *ESP8266*, é adequado para esta função devido às suas capacidades de comunicação de rede de baixa potência e alta eficiência. Com um processador de 32 *bits* operando a 80 *MHz*, 4 *MB* de memória *flash* e 80 *KB* de *SRAM*, este módulo tem o poder de processamento necessário para executar as funções de nó ponte e nó distribuído sem comprometer o desempenho da rede *mesh*. Além disso, a sua arquitetura permite a integração fácil com protocolos de rede de comunicação distribuída. No artigo [39] podemos verificar a capacidade do *ESP8266* a lidar com redes *mesh*, incluindo a sua resiliência a falhas e a eficiência na transmissão de pacotes, o que torna a escolha do *D1 Mini Pro* como nó distribuído ainda mais apropriada.

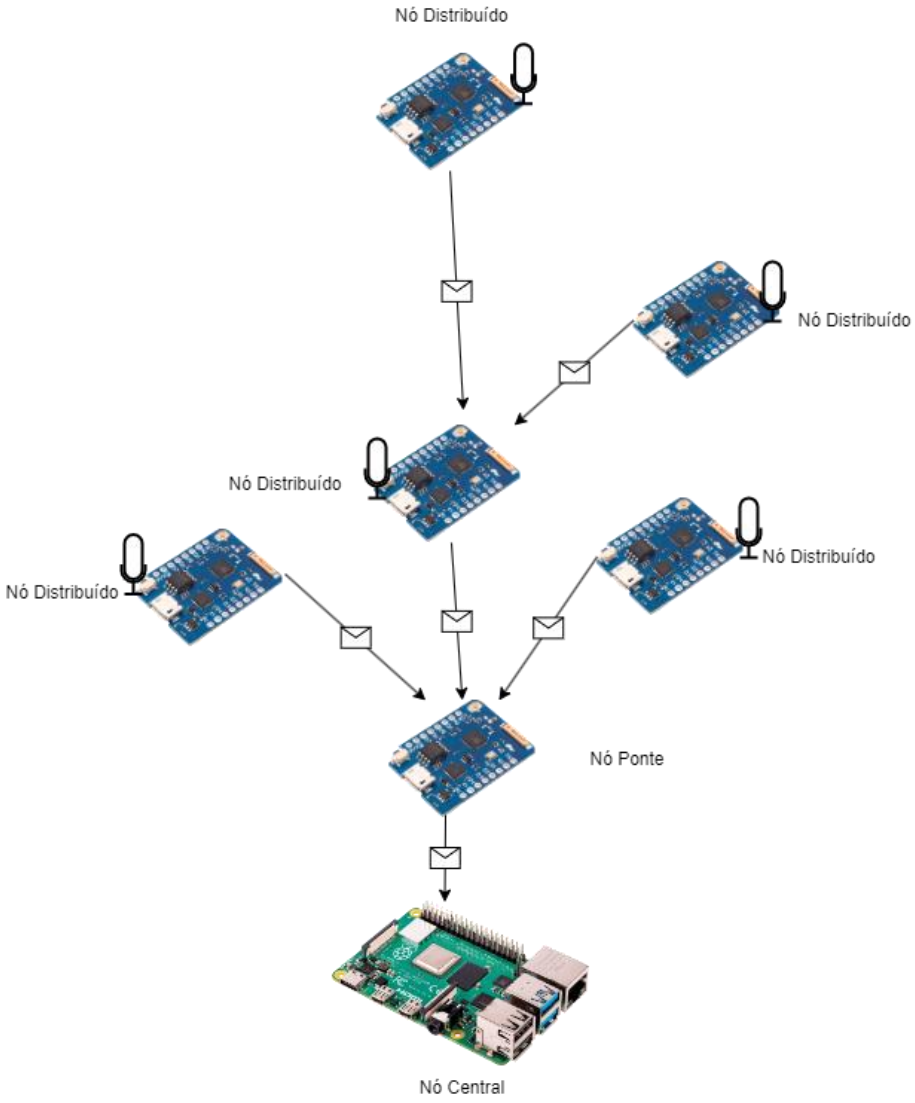

Figura 8 - Arquitetura geral da abordagem 1

A configuração dos diversos *D1 Mini Pro* como nós distribuídos foi feita através do *Arduino Integrated Development Environment (IDE*) versão 2.3.3. Para implementar a rede *mesh* foi utilizada a biblioteca *PainlessMesh* [40] versão 1.5 (Figura 9). Esta biblioteca foi escolhida pela sua capacidade de criar redes *mesh* auto-organizadas e descentralizadas com facilidade. Uma das principais vantagens da *painlessMesh* é a sua compatibilidade com dispositivos baseados no *ESP8266*, como o *D1 Mini Pro*, o que facilita a configuração e gestão da rede de forma eficiente. Além disso, a biblioteca oferece suporte nativo à reconexão automática de nós, garantindo que a rede se reorganize dinamicamente em caso de falhas ou remoção de dispositivos.

Para garantir o funcionamento adequado do sistema, instalámos também as dependências necessárias (Figura 10), sendo elas a *ArduinoJSON* [41] versão 7.0.4,

TaskScheduler [42] versão 3.8.5, e *ESPAsyncTCP* [43] versão 1.2.4, que proporcionam o suporte à troca de mensagens entre nós e a gestão eficiente das tarefas assíncronas. Além disso, o pacote *esp8266* [44] versão 3.1.2 (Figura 11), foi instalado para assegurar a compatibilidade e o reconhecimento do *D1 Mini Pro* pelo *Arduino IDE*.

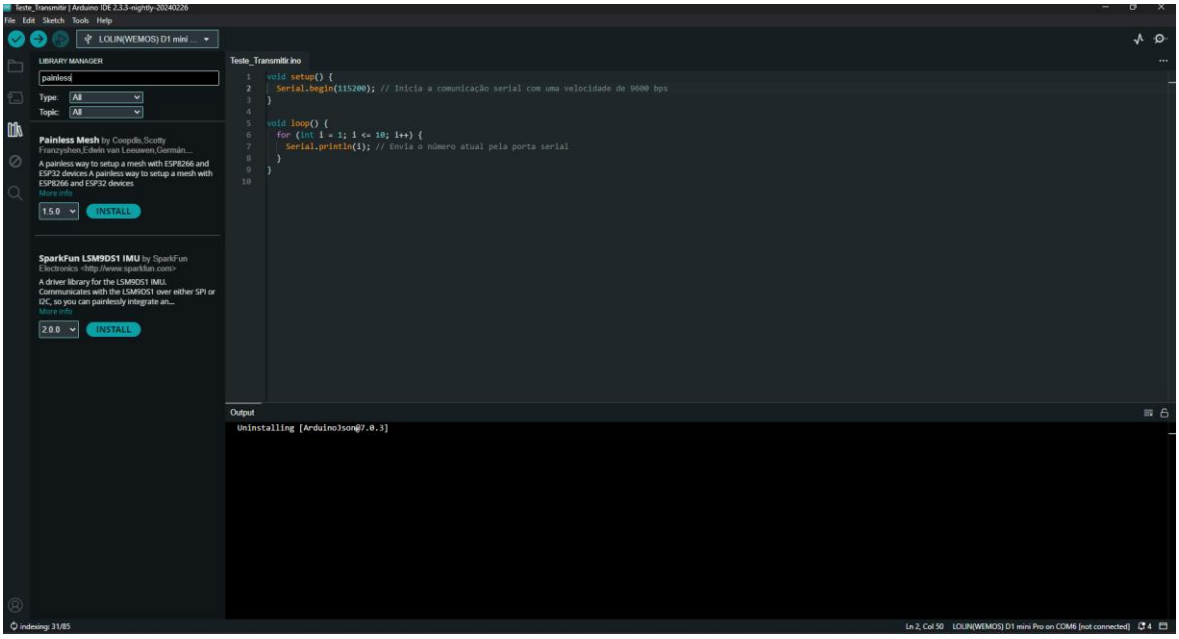

Figura 9 - Instalação da biblioteca painlessMesh

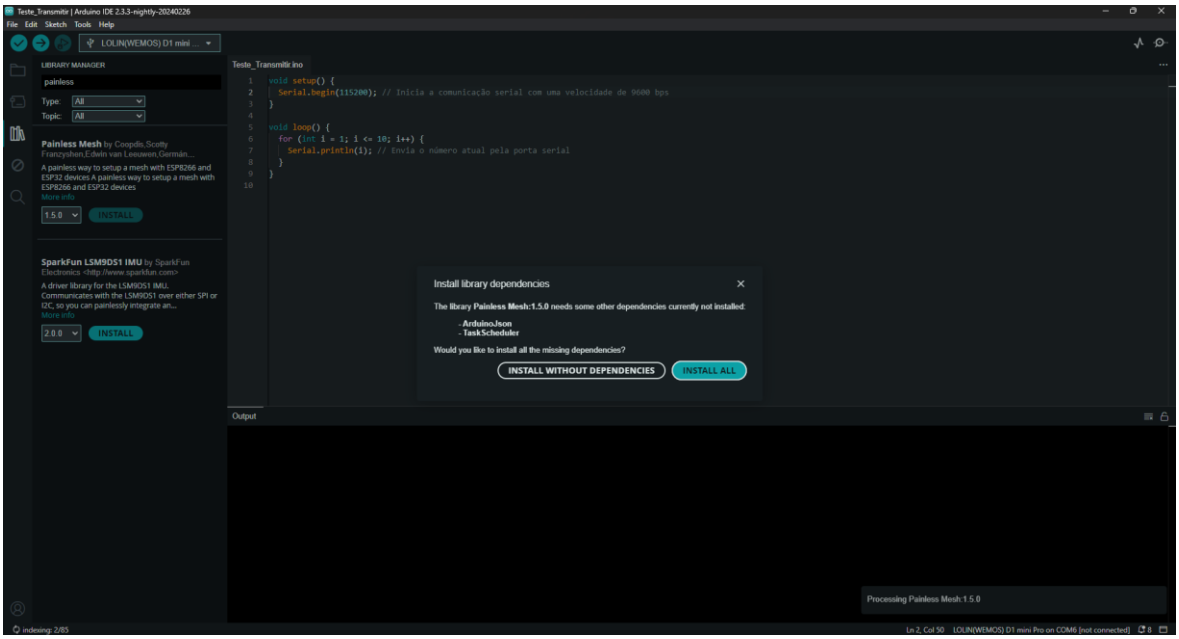

Figura 10 - Instalação das dependências necessárias para a painlessMesh

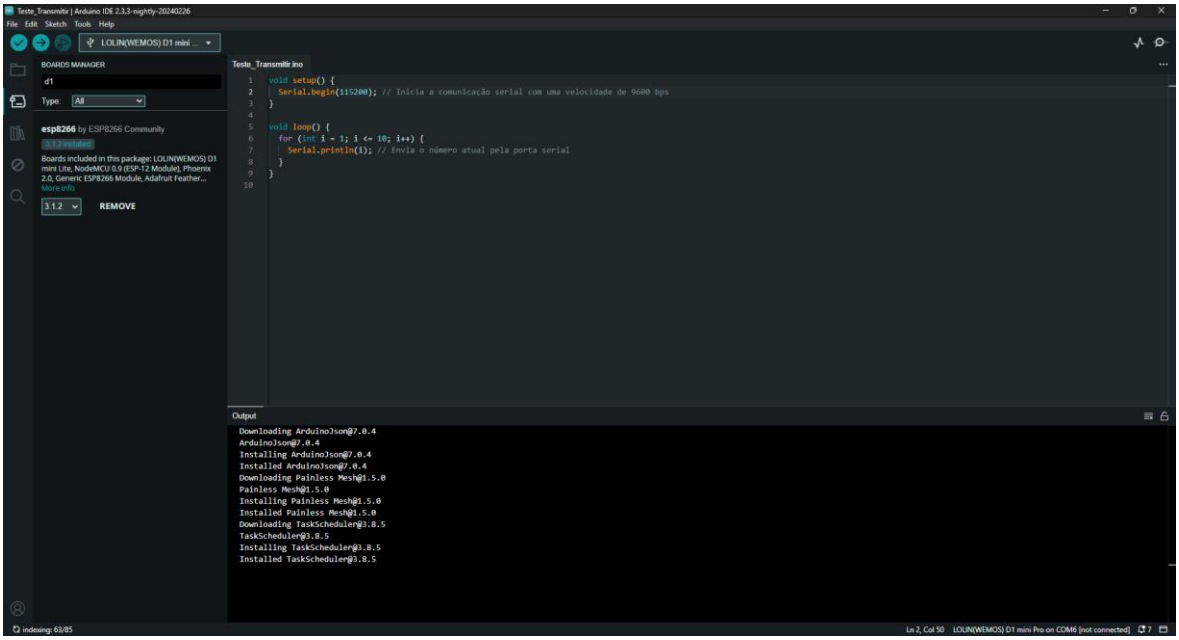

Figura 11 - Instalação da biblioteca esp8266

O nó ponte será configurado como o nó raiz da rede *mesh* e será conectado ao *Raspberry Pi 4B* através dos pinos *general-purpose input/output* (*GPIO)* de 5V, *GND*, 14 e 15 do *Raspberry Pi 4B*, visível na Figura 13, e 5V, *GND*, *RX* e *TX* do *D1 Mini Pro*, visível na Figura 14. Esta configuração permite que o nó central receba as mensagens da rede *mesh* e que as transmita para o nó central. Para a comunicação entre o nó ponte e o nó central, será utilizada a biblioteca *PySerial* [45] versão 3.5 visível na Figura 12 no *Raspberry Pi 4B*, que possibilita a receção de dados via interface *serial*.

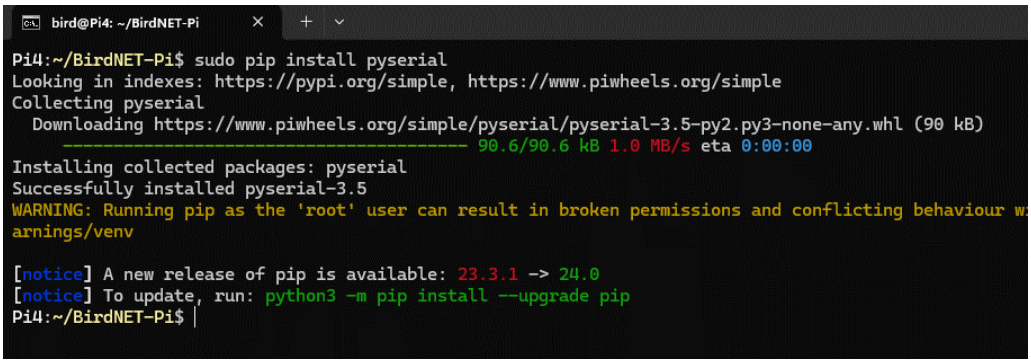

Figura 12 - Instalação da biblioteca PySerial

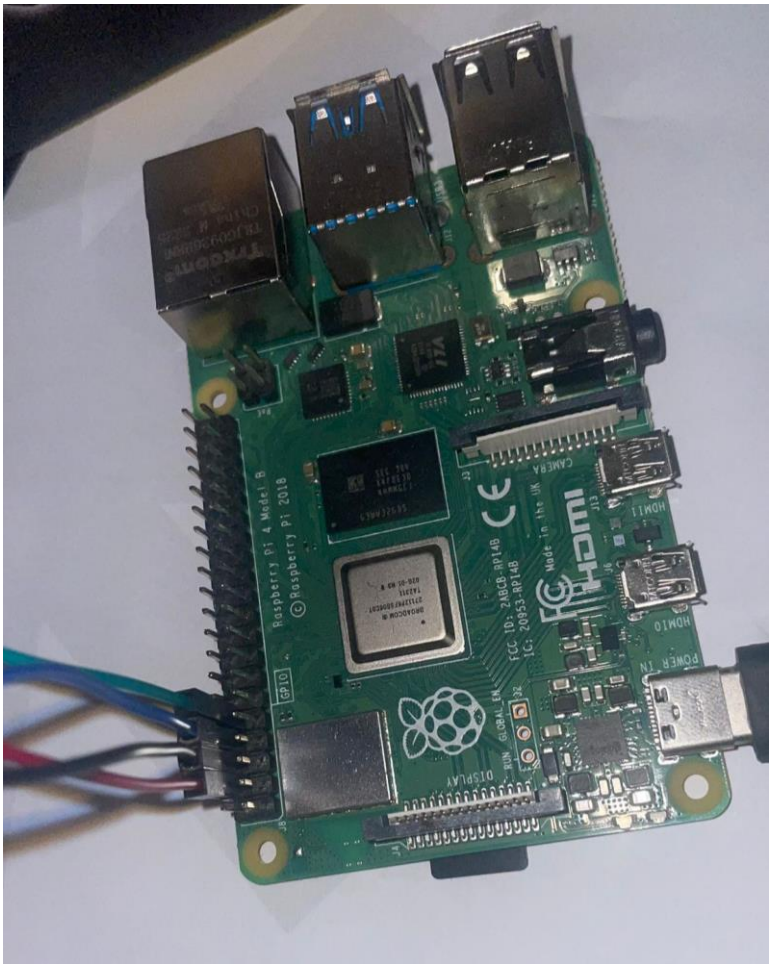

Figura 13 - Nó central com ligação GPIO para o nó ponte

Figura 14 - Nó ponte com ligação GPIO para o nó central

4.2.1 Software

Com a escolha do hardware e a obtenção de um áudio de referência, o próximo passo seria o desenvolvimento do software para a rede *mesh*. Este processo foi dividido em várias etapas de maneira a garantir a integração e o funcionamento da mesma.

Inicialmente, foi desenvolvido um *script* para transferir o ficheiro de áudio do computador para os nós distribuídos, utilizando as bibliotecas *FS* e *LittleFS* para gerir o sistema de ficheiros no *D1 Mini Pro*. Nos nós distribuídos, o sistema de ficheiros foi devidamente inicializado, com uma verificação do sucesso dessa operação de maneira que caso ocorra uma falha, uma mensagem de erro é registada. Após feita esta verificação, procede-se à identificação de uma conexão *serial* ativa. Quando uma conexão é estabelecida, é criado um ficheiro denominado 'audio.mp3' em modo de escrita, seguido de uma verificação adicional para assegurar que o ficheiro foi criado corretamente. Após esta verificação, os dados são armazenados no ficheiro, conforme ilustrado na Figura 15.

No lado do computador, uma ligação *serial* é estabelecida, abrindo-se o ficheiro de áudio em modo de leitura. Um *loop* é então implementado para monitorizar continuamente a existência de dados a serem lidos. Quando dados estão disponíveis, estes são transmitidos pela conexão *serial*, como demonstrado na Figura 16.

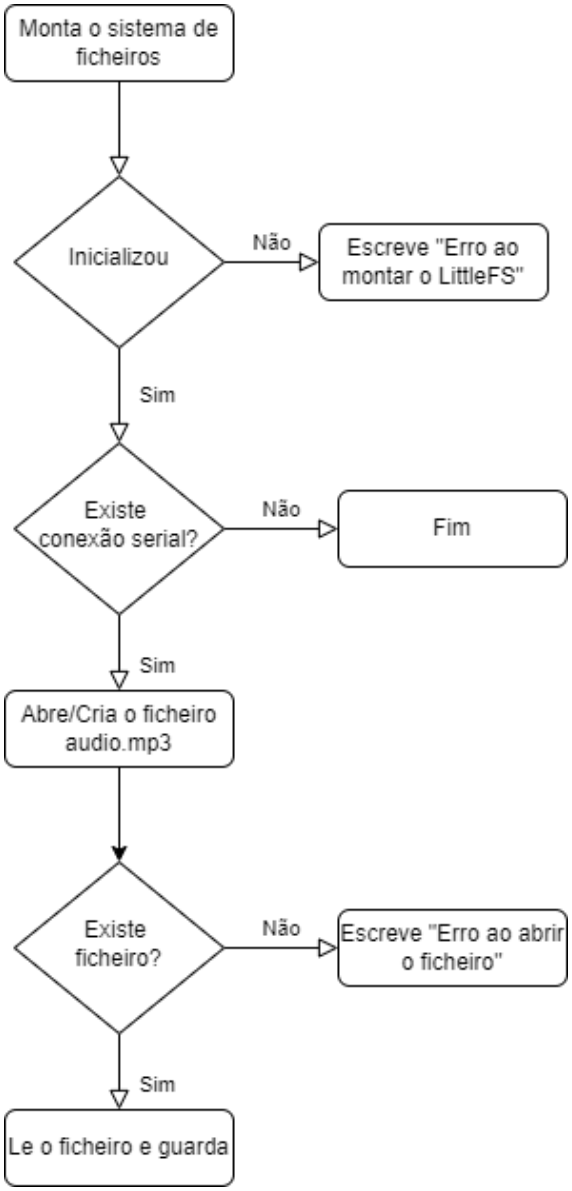

Figura 15 - Fluxograma para receção do ficheiro áudio nos nós distribuídos

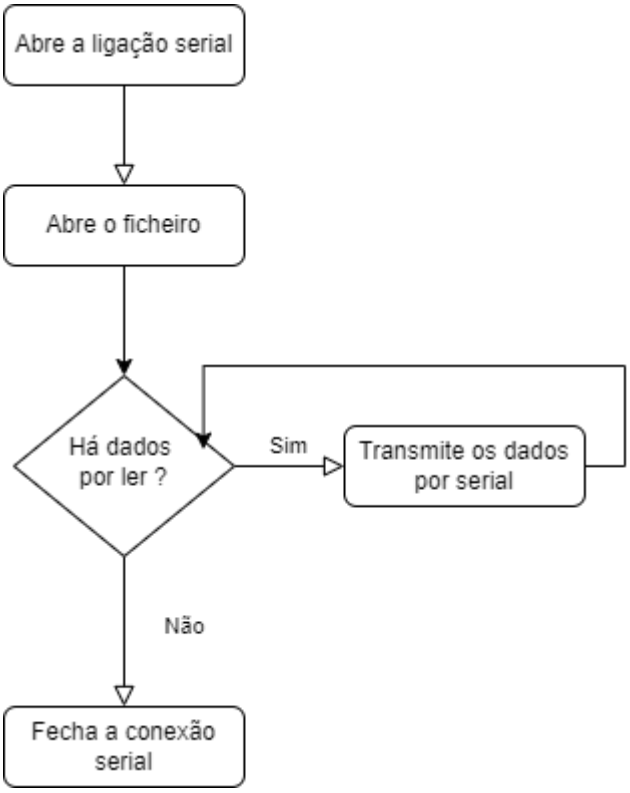

Figura 16 - Fluxograma para transmissão do ficheiro para os nós distribuídos

Em seguida, foi configurada a rede *mesh* com uma estrutura inicial simples, projetada para testar a conectividade entre os nós. Nessa configuração, cada nó enviava o seu ID para toda a rede utilizando *broadcast*, como ilustrado na Figura 17, garantindo que existia comunicação entre todos os nós. Esta configuração inicial, embora básica, permitiu verificar se havia conectividade total entre os dispositivos. Após confirmar o funcionamento correto da comunicação, o *script* foi modificado para direcionar as mensagens exclusivamente ao nó raiz, utilizando o seu ID. A Figura 18 demonstra como o comportamento do sistema após essa adaptação, com as mensagens sendo enviadas diretamente para o nó central.

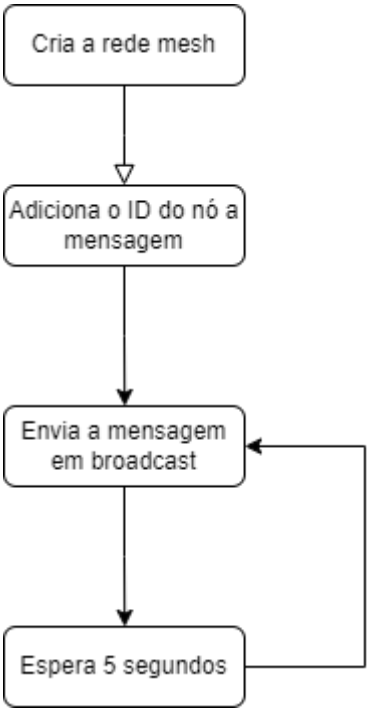

Figura 17 - Fluxograma do envio de mensagem broadcast com ID do nó distribuído

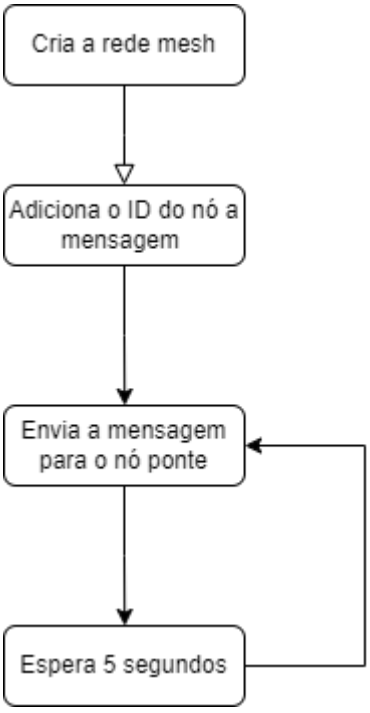

Figura 18 - Fluxograma do envio de mensagem dirigida ao nó ponte com o ID do nó distribuído

No nó ponte, o *script* foi criado de maneira a receber exclusivamente mensagens e transmiti-las via comunicação *serial*, permitindo que sejam visualizadas no nó central, conforme demonstrado na Figura 19. Já na Figura 20, é apresentado o fluxograma do *script* que corre no nó central que é responsável por receber as mensagens enviadas

pelo nó ponte e exibi-las no terminal, garantindo a monitorização das comunicações entre os nós da rede mesh e o nó central.

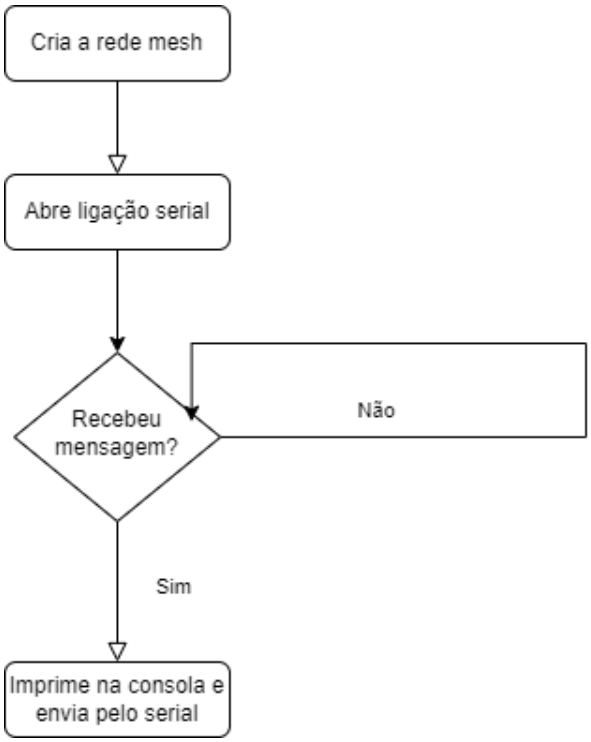

Figura 19 - Fluxograma da receção e escrita de dados no nó central

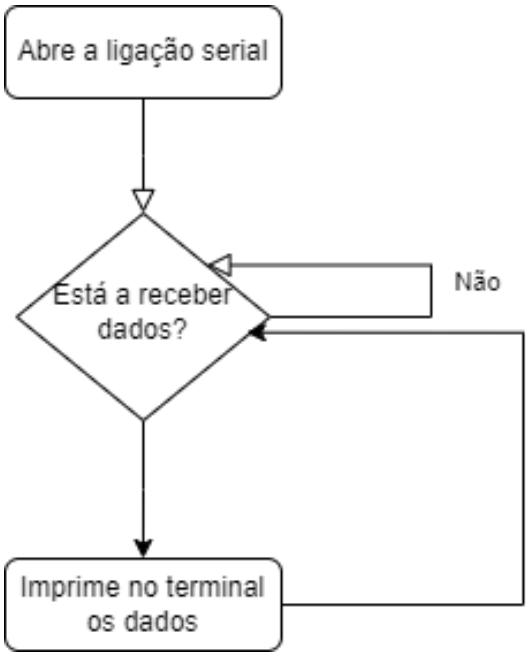

Figura 20 - Fluxograma do código nó ponte para receção das mensagens e reencaminhamento para o nó central

Após a implementação da rede conseguimos verificar que existia uma transferência de mensagens entre os nós distribuídos da rede e entre a rede e o nó central. Esta validação foi fundamental para garantir que a comunicação estava a funcionar

conforme esperado e, com base nestes resultados, procedemos à alteração dos *scripts* para permitir o envio do ficheiro de áudio previamente gravado nos nós distribuídos.

O primeiro passo no envio do áudio consistiu em dividir o ficheiro em *chunks* ou blocos. Este processo envolveu a leitura do ficheiro de áudio, a divisão dos dados em pacotes de um tamanho adequado que, devido a quantidade reduzida de *RAM* dos *D1 Mini Pro*, escolhemos 256 *bytes* e o envio sequencial de cada bloco para o nó central. O fluxograma deste processo, visível na Figura 21 ilustra claramente as etapas, desde a leitura do ficheiro até a verificação de recebimento no nó raiz.

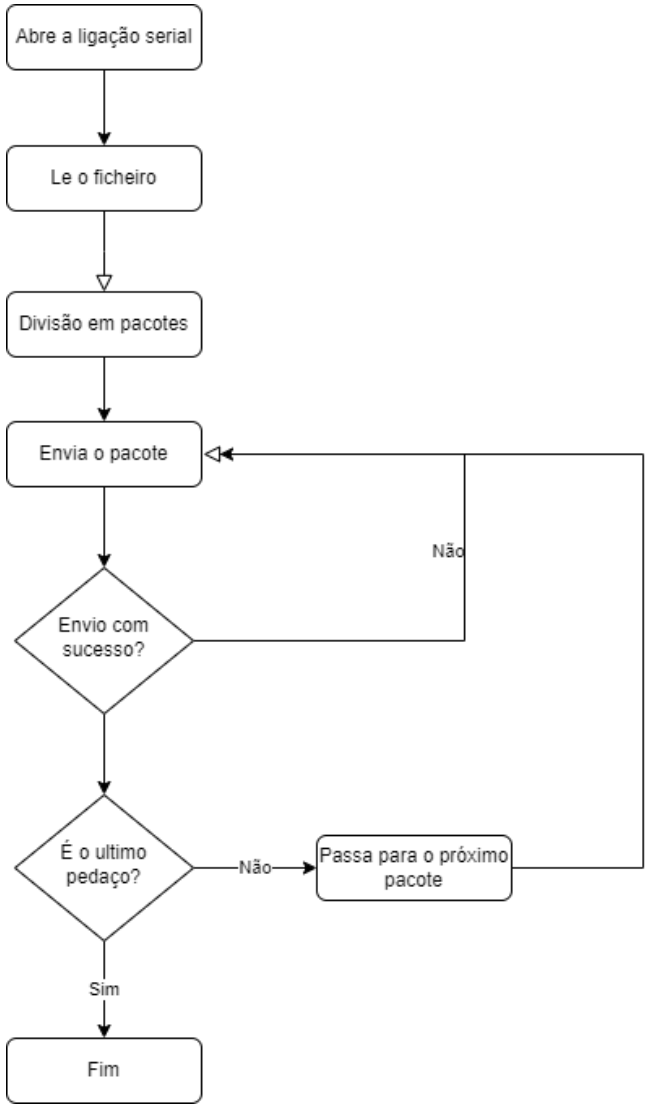

Figura 21 - Fluxograma do envio do áudio através de blocos

Em seguida, o código no *Raspberry Pi 4B* foi ajustado para receber os dados enviados pelos nós distribuídos. O *script* no *Raspberry Pi* aguardava pela receção dos blocos e, assim que todos os dados foram recebidos, tentava reconstituir o ficheiro

original a partir dos blocos. No entanto, apesar da implementação adequada, os dados não foram recebidos com sucesso. Ao verificar o ficheiro gerado, notámos que o mesmo apresentava um tamanho reduzido e, consequentemente, não funcionava como esperado.

Com o intuito de resolver o problema de transmissão, decidimos experimentar o envio dos dados em *raw binary*. Este novo método envolveu a leitura do ficheiro de áudio como dados binários que seriam enviados divididos em blocos. O fluxograma visível na Figura 22 demonstra o processo da leitura do ficheiro em formato binário e o envio para o nó central.

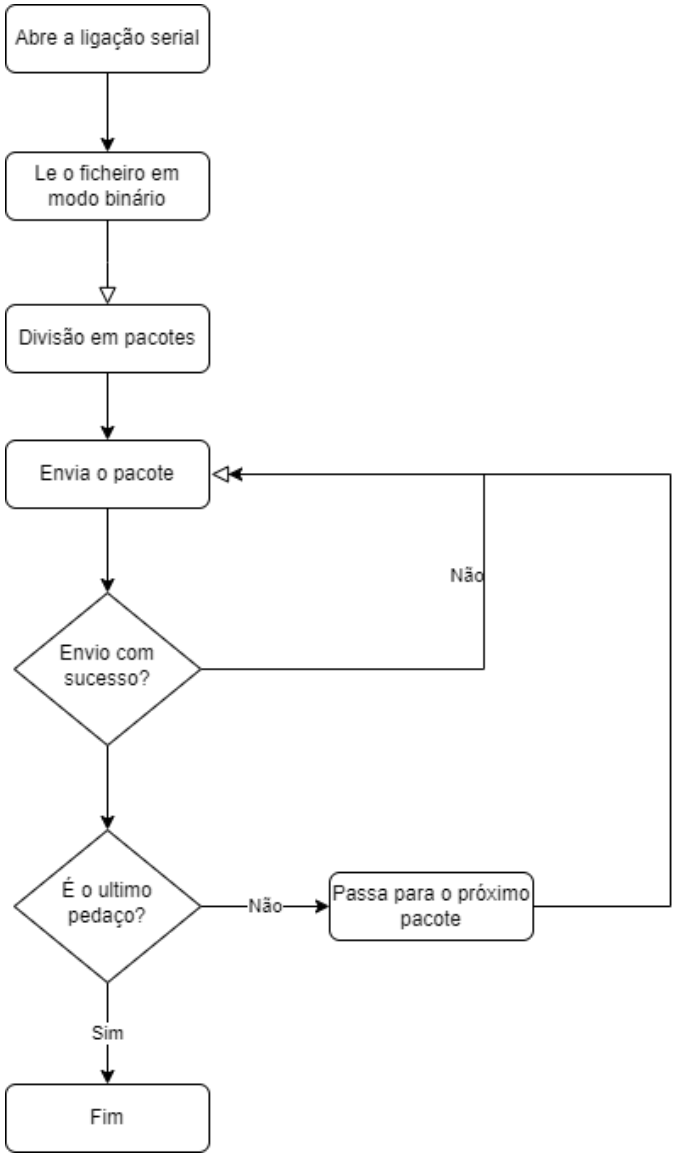

Figura 22 - Fluxograma do envio do áudio através de blocos binários

No lado do *Raspberry Pi*, o código foi novamente adaptado para receber os dados binários e gravá-los em um novo ficheiro. Embora esta abordagem tenha permitido que algo fosse recebido, a análise do ficheiro resultante revelou que apenas alguns bytes tinham sido transferidos. O tamanho final do ficheiro gerado era de apenas 1 *KB*, muito inferior ao tamanho original do áudio, e o ficheiro não era funcional.

Estávamos a enfrentar dificuldades significativas na transferência do áudio pela rede *mesh*, mesmo após diversas tentativas de resolução. A procura de uma solução funcional tornou-se uma prioridade, uma vez que estabelecemos o objetivo de criar uma rede *mesh* eficaz e otimizada. Assim, foi necessário explorar alternativas viáveis.

No decorrer do Projeto 1, identificámos duas abordagens potenciais para a transmissão e análise do áudio. Com base nos dados disponíveis na altura, decidimos que a transmissão do ficheiro de áudio completo para o nó central, onde a análise seria realizada, era a opção mais adequada. No entanto, ao depararmo-nos com problemas persistentes relacionados com a transferência, optámos por reavaliar a nossa estratégia. Assim, decidimos implementar a abordagem em que o áudio é analisado localmente nos nós distribuídos, enviando apenas os resultados da análise para o nó central.

4.3 Abordagem 2

Nesta abordagem, os nós distribuídos são responsáveis pela captura e análise local dos áudios capturados. A análise é realizada diretamente no nó distribuído, onde os resultados relevantes, como a espécie de pássaro identificada e o nível de confiança, são extraídos e enviados para os nós ponte. Estes nós ponte reencaminham as mensagens para o nó raiz, que por sua vez, transmite as informações para o nó central. O nó central utiliza esses dados para tomar as ações necessárias para dispersar os pássaros, com base na informação recebida da rede *mesh*.

Com a nova abordagem adotada, é necessário realizar alterações no setup previamente descrito, especialmente em relação aos dispositivos utilizados e à identificação dos mesmos. Como o processamento será feito localmente, é fundamental contar com nós mais robustos e capazes de realizar a análise do áudio, levando à introdução dos *Raspberry Pi 02W* nas extremidades da rede.

Com estas mudanças, para garantir uma maior clareza, os *Raspberry Pi 02W* passarão a ser os principais nós responsáveis pela análise e serão designados como nós distribuídos. Os *D1 Mini Pro*, que anteriormente desempenhavam essa função, agora atuarão como intermediários na comunicação e serão denominados nós ponte. O *D1 Mini Pro* que estabelece a ligação entre a rede *mesh* e o nó central será agora designado como nó raiz. Estas alterações podem ser observadas na Figura 23.

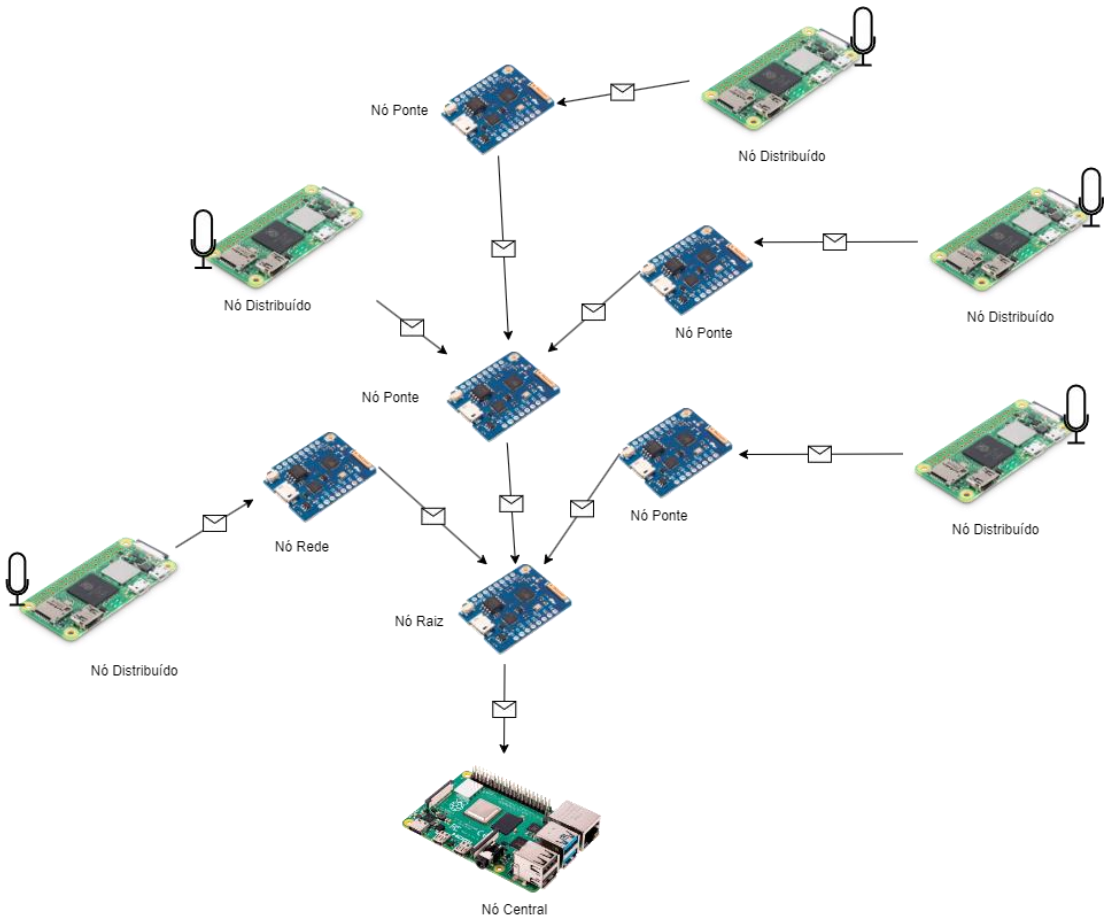

Figura 23 – Arquitetura geral da Abordagem 2

4.3.1 Software

Com a nova abordagem adotada, foi necessário realizar não apenas alterações nos dispositivos utilizados, mas também na componente de software, devido às mudanças no comportamento dos nós da rede e aos novos nós distribuídos.

Para o envio das mensagens, foi preciso definir tanto o formato quanto as informações a serem transmitidas pela rede. Baseado no protocolo previamente definido no Projeto 1, desenvolvemos um novo protocolo de mensagens com o seguinte formato:

(IdNo: <id>, Espécie: <nome>, Confianca: <confiança>, timeStamp: <timeStamp>)

Neste protocolo, a informação sobre a espécie detetada (Espécie), o nível de confiança (Confianca) e a data e hora em que foi capturada (timeStamp) são enviadas pelo nó distribuído para o nó ponte correspondente e este adiciona o seu identificador (IdNo) e encaminha a mensagem para o nó raiz. Deste modo, conseguimos identificar a posição do nó através de uma base de dados com as localizações dos nós, que embora não seja o foco principal deste projeto, permite uma visão espacial da rede. Além disso, incluímos os dados essenciais sobre a espécie, a percentagem de confiança e a data e hora com a finalidade futura de estatística.

Para assegurar que os dados são entregues corretamente ao nó central e evitar uma sobrecarga desnecessária na rede *mesh*, é crucial que os nós ponte enviem as mensagens de forma direta e exclusiva para o nó raiz. Para garantir este comportamento, utilizámos o método integrado na *API* da biblioteca *PainlessMesh* denominado *newCallbackConnection* [40], que é invocado sempre que uma nova conexão é estabelecida na rede. Este método fornece o ID do novo nó conectado e permite ao nó raiz enviar uma mensagem no formato "*Root: <id>"* para o novo nó. Do lado dos nós da rede, utilizámos o método integrado chamado *newRecievedCallback* [40] que é invocado sempre que um nó recebe uma mensagem da rede. Desta forma, quando um nó recebe uma mensagem da rede *mesh*, verifica se a mesma começa com "*Root:* " e, se for o caso, o nó extrai o ID do nó raiz e armazena-o numa variável que será usada nas comunicações futuras, garantindo que todas as mensagens sejam enviadas diretamente para o nó raiz.

Durante a fase inicial de testes, o *script* criado foi configurado com os dados da espécie e do nível de confiança pré-preenchidos com os valores *Eurasian Magpie* e 0.9949 respetivamente. Estes dados correspondem aos dados retirados da análise do ficheiro de áudio de teste no intervalo entre os segundos 9 e 12, como ilustrado na Figura 7. O fluxograma representado na Figura 24 demonstra o processo que ocorre no nó distribuído para a preparação e envio da mensagem para o nó da rede, via conexão serial enquanto a Figura 25 ilustra o processo de receção da mensagem, adição do ID do nó e retransmissão da mensagem do nó ponte para o nó raiz.

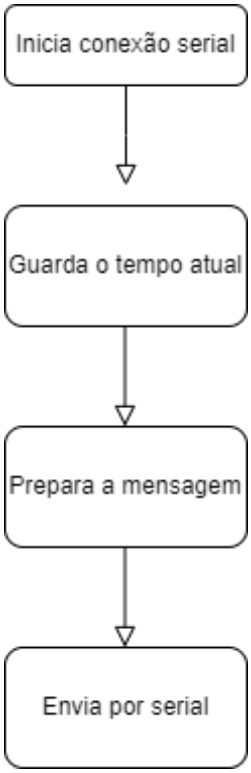

Figura 24 - Fluxograma da preparação da mensagem e envio para o nó ponte

Figura 25 - Fluxograma da receção, atualização e encaminhamento da mensagem para o nó ponte

Após termos implementado com sucesso o protocolo de mensagens na nossa rede, o próximo passo é integrar a análise de ficheiros de áudio e preencher as mensagens com dados reais extraídos dessa análise. Para isso, foi necessário modificar o *script* "analyze.py" do *BirdNET-Analyzer* de maneira a permitir que, após a análise do áudio, os resultados sejam retirados diretamente do ficheiro gerado.

Dentre os resultados, são extraídos os dados referentes à espécie identificada com o maior nível de confiança, juntamente com o *timestamp* obtido no momento em que a análise é iniciada. Estes dados são então formatados de forma a coincidirem com o protocolo de mensagens previamente definido e, finalmente, a mensagem formatada é enviada para a rede *mesh* (Figura 26).

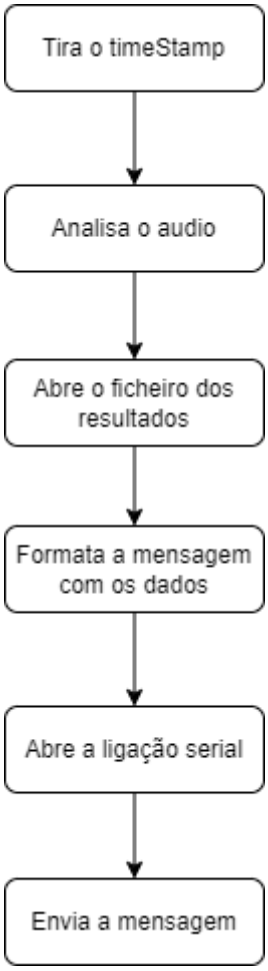

Figura 26 - Fluxograma da análise do áudio, preparação e envio de mensagem

Agora que o desenvolvimento do sistema está concluído e o protocolo de comunicação está implementado, o próximo capítulo será dedicado à análise dos testes realizados. Nesta fase, iremos avaliar o desempenho da rede *mesh* e a eficácia do sistema em transmitir os dados analisados, explorando diferentes configurações e cenários.

5 Testes e Validação

Com a implementação completa do sistema, a fase de testes é essencial para validar o desempenho e a eficácia da rede *mesh* desenvolvida. Esta etapa tem como objetivo avaliar a latência, a confiabilidade e a transmissão de dados em diferentes configurações de nós. No decorrer deste capítulo, serão apresentados os resultados dos testes realizados e as conclusões obtidas acerca do comportamento da rede em diferentes cenários.

É importante notar que os valores de análise e envio podem variar significativamente, e podem ocorrer *outliers*. Para mitigar a influência de flutuações inesperadas nos resultados, cada teste será realizado 30 vezes, e utilizaremos a média dos tempos registados como métrica principal. Os tempos que serão monitorizados incluem o tempo de análise, que representa o período necessário para a realização da análise do áudio no nó distribuído; o tempo total, que é a diferença entre o *timestamp* quando o pacote chega e o *timestamp* contido na mensagem; e o tempo de envio, que é calculado subtraindo o tempo de análise do tempo total.

Foi realizada uma série de testes de latência na rede *mesh*, iniciando com um teste base em que o nó ponte, associado ao nó distribuído, está ligado diretamente ao nó raiz, sem a presença de nós intermédios. Este primeiro teste servirá como referência para avaliar o desempenho da rede. Em seguida, iremos adicionar nós intermédios e alterar a estrutura da rede para observar como estas modificações impactam os tempos de latência.

Na Figura 27, podemos observar alguns dos cenários da rede que pretendemos testar. Nos cenários 1, 2 e 3, o objetivo é verificar o impacto da adição de diversos nós intermédios na latência da rede. O cenário 4 tem duas finalidades: primeiro, determinar qual dos nós seria escolhido para a comunicação e, segundo, testar a capacidade de recuperação da rede em caso de falha de um nó. Por fim, os cenários 5 e 6 serão utilizados para realizar testes de stress, onde pretendemos avaliar o impacto na latência quando todas as mensagens são enviadas por um único nó intermédio em comparação com envios realizados por nós intermédios diferentes.

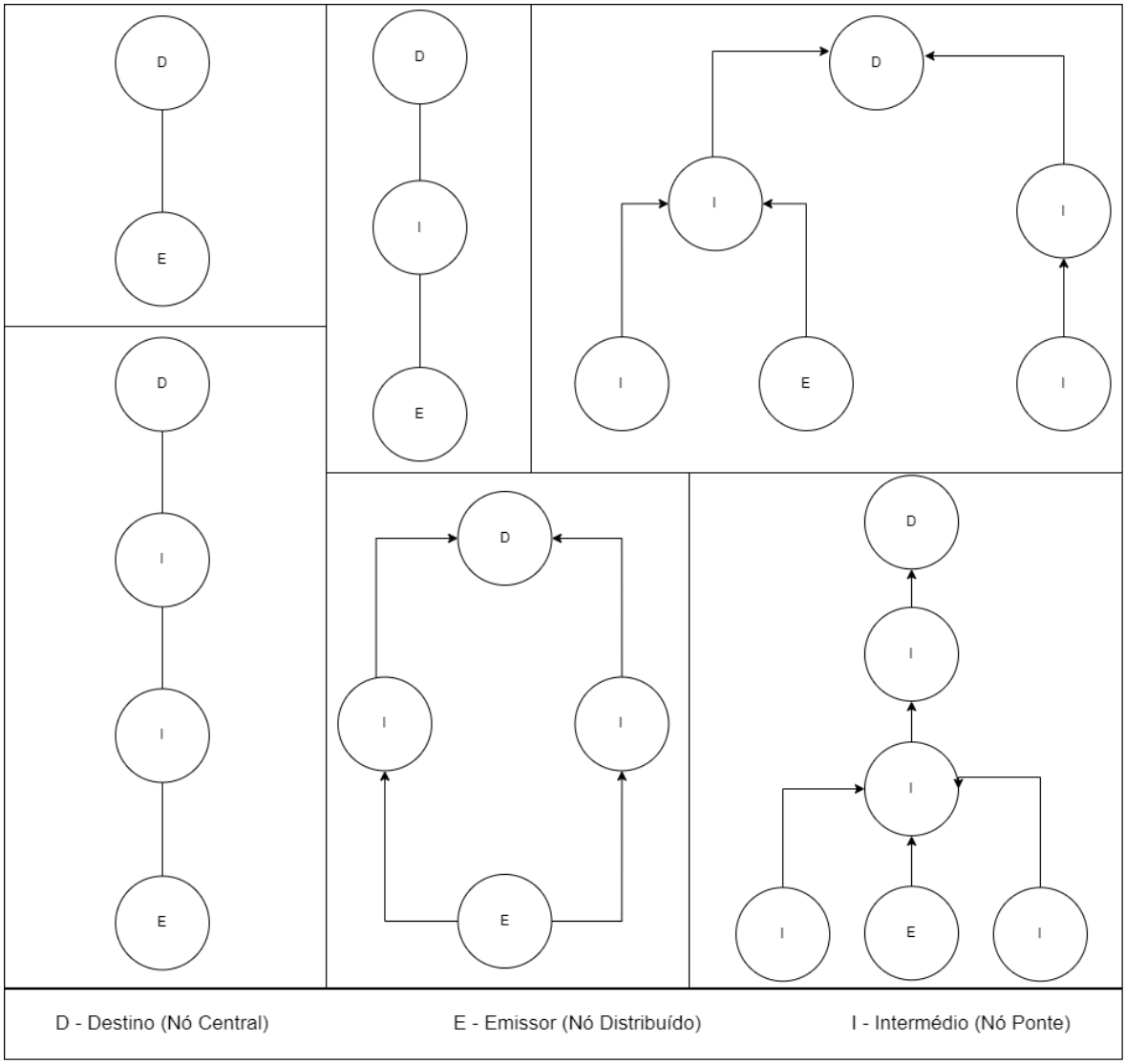

Figura 27 - Cenários de teste da rede

Ao observar os valores obtidos nos cenários 1, 2 e 3, podemos concluir que existe uma diferença nos tempos de envio, sendo esta de 0,16 segundos com a introdução de um nó intermédio (Figura 29) e de 0,25 segundos ao introduzir um segundo nó intermédio (Figura 30), em comparação com os tempos do nó diretamente conectado (Figura 28). No entanto, não podemos concluir definitivamente que a presença de nós intermédios melhora o tempo de envio, pois, ao analisar as tabelas com os tempos, verificamos um tempo de quase 6 segundos de envio nos dados do cenário 1. Se substituirmos esse tempo por um valor mais próximo dos obtidos, a média dos tempos apresenta uma diferença mínima em relação aos restantes.

Apesar de os resultados mostrarem que não há um grande impacto no tempo de latência, a adição de nós intermédios contribuiu para a melhoria da rede. Em pomares

de grandes dimensões, esses nós servem como nós de retransmissão, encurtando a distância entre os nós distribuídos e o nó central que resulta numa melhor qualidade do sinal e adiciona redundância aos caminhos.

Figura 28 - Gráfico dos tempos do cenário 1

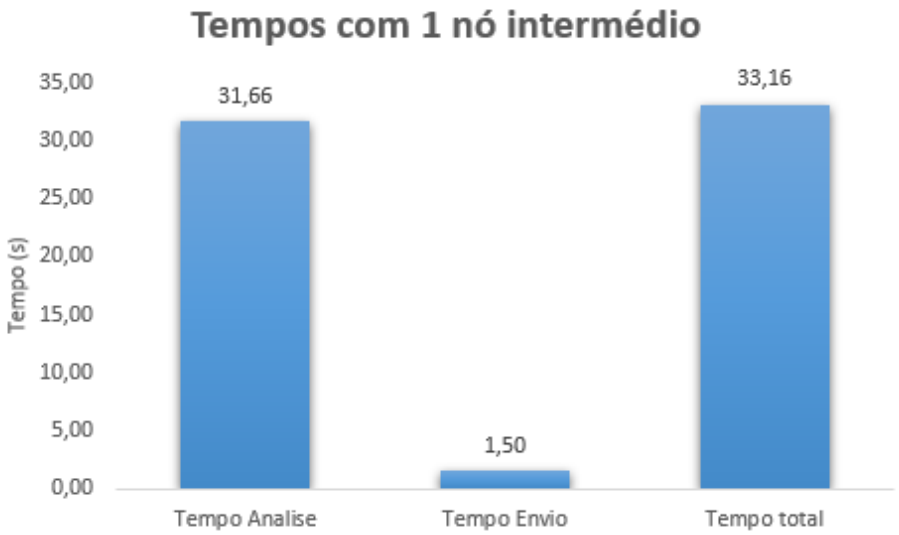

Figura 29 - Gráfico dos tempos do cenário 2

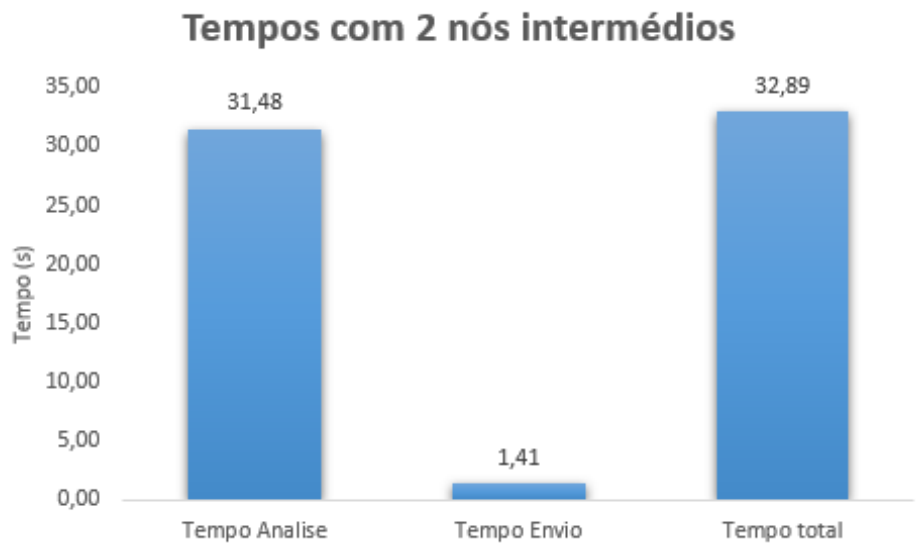

Figura 30 - Gráfico dos tempos do cenário 3

Ao tentar implementar o cenário 4, deparamo-nos com um comportamento inesperado da biblioteca *painlessMesh*, que não nos permitiu criar o cenário conforme planeado. Na tentativa de colocar dois nós à mesma distância, ambos ligados ao destino e ao emissor, verificamos que a rede considerava sempre um dos dois intermédios como "filho" do outro, resultando num cenário idêntico ao cenário 3.

Devido a essa limitação, não conseguimos testar os cenários 4, 5 e 6 conforme pretendíamos. Contudo, para avaliar a recuperação da rede e realizar o teste de latência sob condições de sobrecarga, decidimos realizar dois testes alternativos no cenário ilustrado na Figura 31, onde temos um nó identificado como I/E devido à sua dupla funcionalidade.

No primeiro teste, utilizaremos este nó como intermédio para avaliar a recuperação da rede. Durante uma transmissão, o nó será removido para verificar se a rede consegue se recuperar, qual o impacto no tempo de envio e se a mensagem chega ao destino ou é perdida. Em seguida, o nó atuará como emissor, permitindo o envio simultâneo de várias mensagens de ambos os emissores. Isso resultará na sobrecarga do nó ponte associado a um dos emissores, que receberá as mensagens de ambos os emissores.

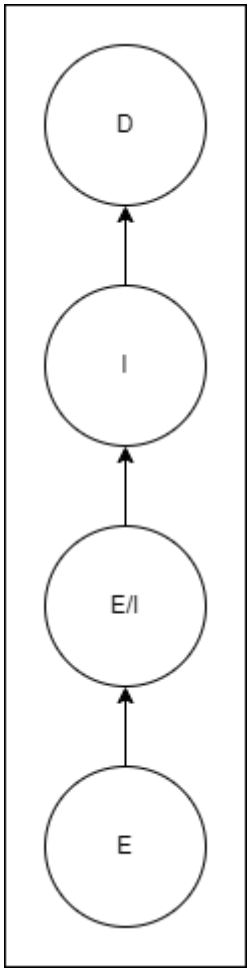

Figura 31 - Cenário alternativo para testes

A topologia de rede representada na Figura 32 ilustra o cenário descrito na Figura 31, onde o nó raiz está diretamente ligado a um nó intermédio, que, por sua vez, se conecta a um nó híbrido. Este nó híbrido desempenha simultaneamente as funções de emissor e intermédio, mas, neste caso, opera como emissor e está ligado a um nó exclusivamente emissor. Conforme mostrado na Figura 33, três mensagens consecutivas foram emitidas através deste nó emissor, utilizando o *script* previamente modificado. Após a receção da primeira mensagem, a ligação com o nó intermédio foi interrompida para forçar a rede a reorganizar-se, permitindo a análise do tempo necessário para esta reorganização, bem como a verificação da receção das mensagens seguintes.

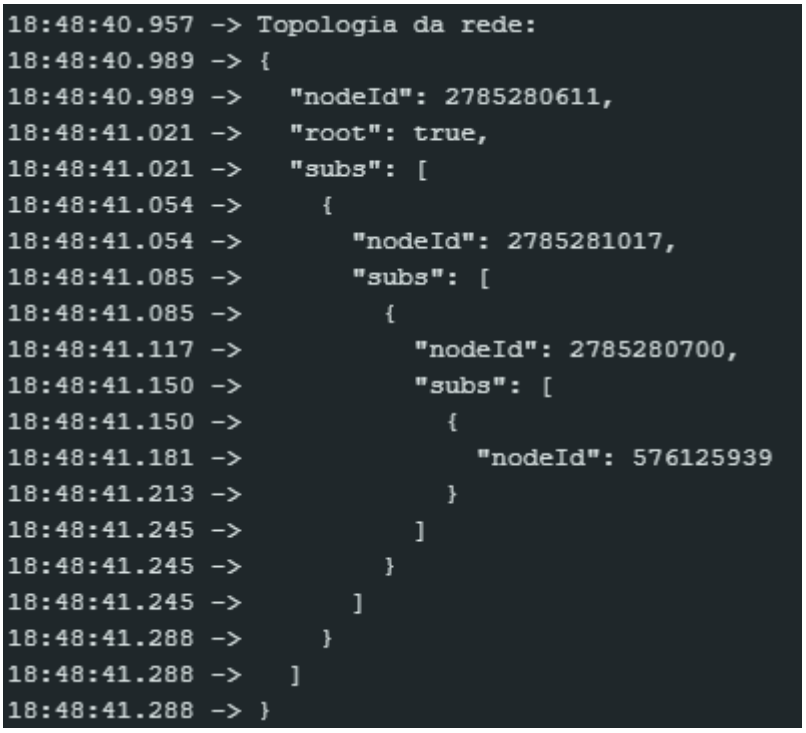

Figura 32 – Topologia da rede inicial

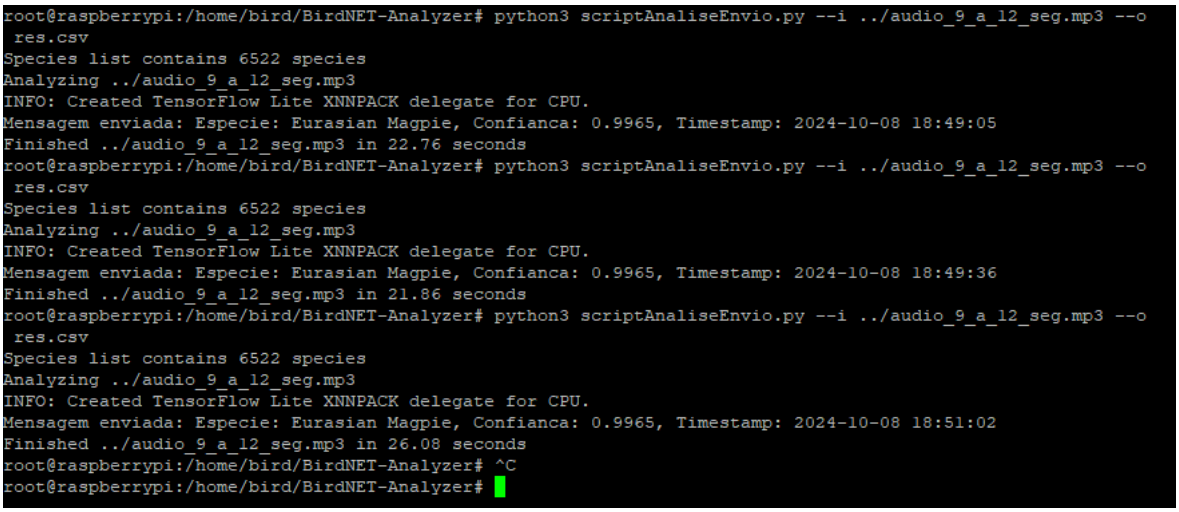

Figura 33 – Envio das mensagens consecutivas

Na Figura 34, pode-se verificar que após a receção da primeira mensagem, a rede inicialmente detetou a perda do nó intermédio e mostrou uma topologia que continha apenas com o nó raiz e, posteriormente, reconfigurou-se completamente para incluir os restantes nós. O tempo de reorganização total, que corresponde à segunda atualização da topologia, foi de aproximadamente 1 minuto e 26 segundos, calculado a partir da diferença entre o momento da receção da primeira mensagem (18:49:18:585) e o instante em que a nova topologia, incluindo todos os nós, foi estabelecida (18:50:44:871).

Durante este processo, apenas a terceira mensagem foi entregue com sucesso, indicando que a segunda mensagem foi perdida devido à falha momentânea da rede. Para mitigar este tipo de perda, seria necessário implementar mecanismos que assegurem a retransmissão de mensagens não entregues ou que evitem a sua perda em caso de falhas na rede.

```
18:49:18.585 -> IdNo: 576125939, Especie: Eurasian Magpie, Confianca: 0.9965, Timestamp: 2024-10-08 18:49:05
18:49:45.507 -> Topologia da rede:
18:49:45.540 -> {
18:49:45.540 ->   "nodeId": 2785280611,
18:49:45.540 ->   "root": true
18:49:45.582 -> }
18:50:44.871 -> Topologia da rede:
18:50:44.902 -> {
18:50:44.902 ->   "nodeId": 2785280611,
18:50:44.934 ->   "root": true,
18:50:44.967 ->   "subs": [
18:50:44.967 ->     {
18:50:44.967 ->       "nodeId": 576125939,
18:50:44.998 ->       "subs": [
18:50:45.030 ->         {
18:50:45.030 ->           "nodeId": 2785280700
18:50:45.062 ->         }
18:50:45.062 ->       ]
18:50:45.098 ->     }
18:50:45.098 ->   ]
18:50:45.098 -> }
18:51:04.865 -> IdNo: 576125939, Especie: Eurasian Magpie, Confianca: 0.9965, Timestamp: 2024-10-08 18:51:05
```

Figura 34 – Receção das mensagens e atualização da topologia

Após a verificação dos tempos de latência da rede no seu estado normal, com um único nó a enviar mensagens, torna-se necessário avaliar o comportamento da rede quando vários nós transmitem simultaneamente, como ocorreria em um pomar de maior extensão. Inicialmente, foi realizado um teste onde um emissor disparou 30 mensagens e os tempos correspondentes foram registados como base de comparação. Em seguida, as mesmas 30 mensagens foram enviadas quase ao mesmo tempo por dois emissores, com o objetivo de criar um cenário de stress na rede, especialmente nos nós responsáveis pela ligação à raiz.

A análise dos tempos obtidos revelou um aumento ligeiro de 0,15 segundos na média do tempo de envio (conforme apresentado na Figura 35). Isso indica que, mesmo com apenas dois nós a enviar mensagens através de nós intermédios, já é possível notar uma diferença na *performance* da rede. No gráfico da Figura 36, essa diferença é evidenciada de forma mais clara, através da comparação dos valores mínimos e máximos. Embora o tempo máximo para o envio das mensagens tenha aumentado apenas em 0,8 segundos, representando um acréscimo de 4%, o tempo mínimo para o envio apresentou um aumento de 0,26 segundos, o que equivale a um significativo

incremento de 21%. Este aumento é particularmente relevante, correspondendo a quase um quarto de tempo a mais no envio, e salienta a necessidade de implementar protocolos que melhorem a eficiência do envio de mensagens.

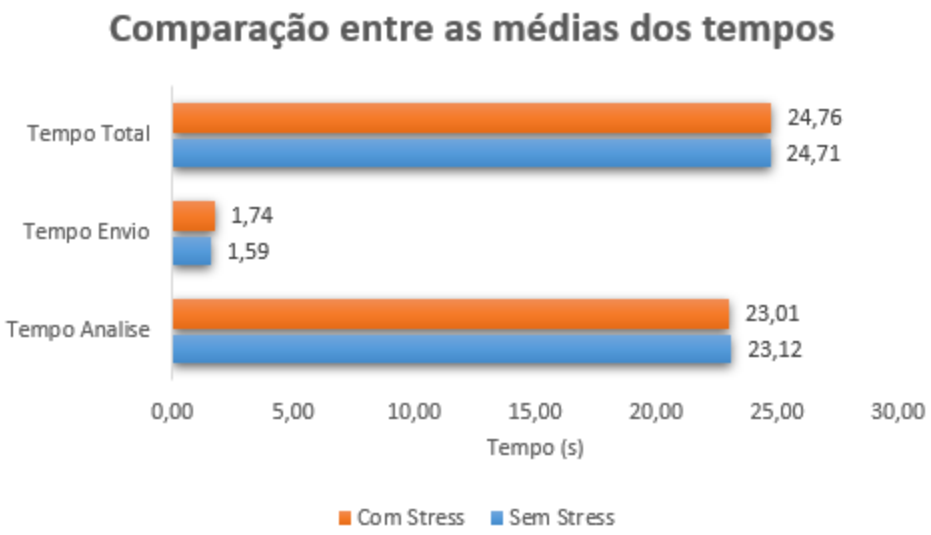

Figura 35 - Gráfico da comparação entre as médias dos tempos

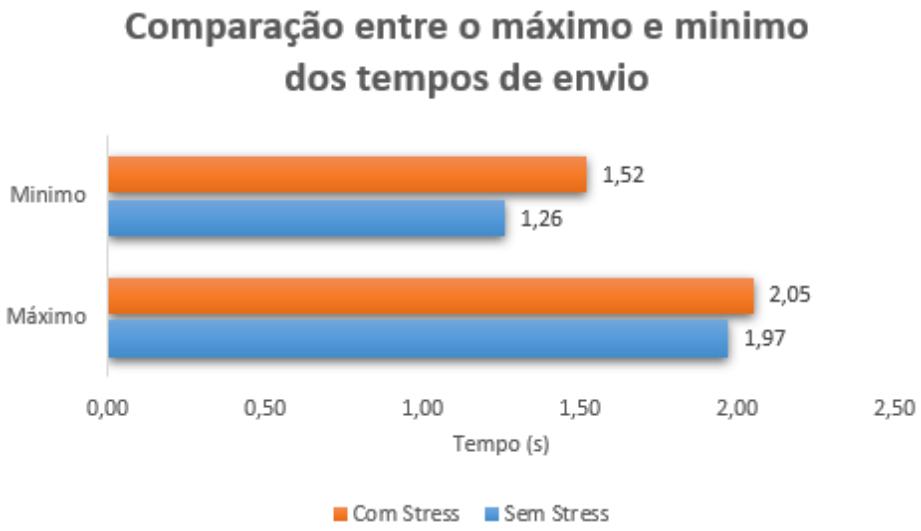

Figura 36 - Gráfico da comparação entre o tempo de envio mínimo e máximo

Em conclusão, apesar de os testes realizados serem simples e de uma maior quantidade de testes com parâmetros diferenciados ser preferível, já é possível concluir que, embora a rede esteja funcional e não apresente tempos de latência extraordinariamente elevados, há espaço significativo para melhorias. A utilização de protocolos destinados a otimizar a comunicação é essencial. É especialmente importante focar na melhoria do tempo de recuperação da rede, implementando protocolos como *Dynamic Source Routing* ou *Ad hoc On-Demand Distance Vector*,

conforme mencionados anteriormente, assim como na forma como as mensagens são entregues. A inclusão de métodos como *acknowledgments* ou uma fila de mensagens poderia prevenir a perda de mensagens, como ocorreu no teste de recuperação da rede, onde uma mensagem foi perdida devido a uma falha temporária. Essa situação, num cenário real, poderia resultar na incapacidade de mitigar ataques de pássaros.

Adicionalmente, seria crucial testar a rede numa escala maior para observar se os tempos de latência aumentam exponencialmente com o crescimento do número de nós e de mensagens na rede. Nesse contexto, a implementação de protocolos avançados de *routing*, como *Network Coding* ou *Opportunistic Routing*, tornar-se-ia necessária para garantir uma comunicação eficiente e robusta.

6 Conclusão

As redes *mesh* são uma solução eficiente para garantir a comunicação entre dispositivos em áreas de grande extensão, particularmente em ambientes rurais, onde a conectividade é frequentemente limitada. A sua principal característica é a interconexão distribuída dos nós, permitindo a transmissão de dados por múltiplos caminhos, o que garante uma maior resistência à falha de nós individuais. No entanto, ao longo do nosso estudo, identificámos diversos problemas associados à implementação de redes *mesh*, como a perda de mensagens, a latência elevada, a corrupção de dados e a necessidade de garantir uma entrega rápida e fiável dos dados. Para mitigar estes problemas, foram investigados diversos protocolos e técnicas, cada um com os seus próprios benefícios e desvantagens. Cada um destes protocolos oferece soluções para melhorar o desempenho da rede em diferentes aspetos, no entanto, determinar qual protocolo seria o mais adequado para a rede *mesh* requer necessariamente testes em condições reais, onde o desempenho de cada protocolo pode ser avaliado corretamente.

Durante o processo de implementação e tentativa de realizar os testes com a Abordagem 1, enfrentámos alguns desafios técnicos que comprometeram o funcionamento ideal da rede. Apesar de várias tentativas para mitigar esses problemas, foi necessário reconsiderar a estratégia adotada. Assim, avançámos para o desenvolvimento da Abordagem 2, ajustando as soluções para contornar as limitações encontradas. Esta nova abordagem permitiu-nos continuar com os testes e avaliações, adequando melhor a solução às exigências de uma rede mesh no contexto rural. Dessa forma, conseguimos prosseguir com a análise do desempenho da rede, focando-nos na escolha dos protocolos mais adequados.

Foram realizados testes para avaliar o desempenho da rede, começando com um cenário onde um único nó emissor enviou mensagens e, de seguida, onde dois emissores enviaram as mesmas mensagens simultaneamente. Os resultados mostraram um aumento significativo de 21 % no tempo mínimo de envio com apenas dois nós o que evidencia a necessidade de otimização da rede através do uso de protocolos.

7 Referências

[1] N. F. Maxemchuk, "Regular Mesh Topologies in Local and Metropolitan Area Networks," *AT&T Technical Journal*, vol. 64, no. 7, pp. 1659–1685, Sep. 1985, doi: 10.1002/j.1538-7305.1985.tb00030.x.

[2] J. W. R. K. Yaling Yang, "Designing routing metrics for mesh networks," Dec. 2005.

[3] Ren Yueqing and Xu Lixin, "A study on topological characteristics of wireless sensor network based on complex network," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, IEEE, Oct. 2010, pp. V15-486-V15-489. doi: 10.1109/ICCASM.2010.5622543.

[4] A. Esmailpour, N. Nasser, and T. Taleb, "Topological-based architectures for wireless mesh networks," *IEEE Wirel Commun*, vol. 18, no. 1, pp. 74–81, Feb. 2011, doi: 10.1109/MWC.2011.5714028.

[5] H. Yu, D. Wu, and P. Mohapatra, "Experimental Anatomy of Packet Losses in Wireless Mesh Networks," in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, IEEE, Jun. 2009, pp. 1–9. doi: 10.1109/SAHCN.2009.5168933.

[6] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and Real-Time Communication in Industrial Wireless Mesh Networks," in *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, IEEE, Apr. 2011, pp. 3–12. doi: 10.1109/RTAS.2011.9.

[7] G. Egeland and P. E. Engelstad, "A Model for the Loss of Hello-Messages in a Wireless Mesh Network," in *2010 IEEE International Conference on Communications*, IEEE, May 2010, pp. 1–6. doi: 10.1109/ICC.2010.5502278.

[8] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini, "Optimal joint routing and link scheduling for real-time traffic in TDMA Wireless Mesh Networks," *Computer Networks*, vol. 57, no. 11, pp. 2301–2312, Aug. 2013, doi: 10.1016/j.comnet.2012.11.021.

[9] O. T. Valle, A. V. Milack, C. Montez, P. Portugal, and F. Vasques, "Experimental evaluation of multiple retransmission schemes in IEEE 802.15.4 wireless sensor

networks," in *2012 9th IEEE International Workshop on Factory Communication Systems*, IEEE, May 2012, pp. 201–210. doi: 10.1109/WFCS.2012.6242568.

[10] J. Singh and J. Singh, "A Comparative Study of Error Detection and Correction Coding Techniques," in *2012 Second International Conference on Advanced Computing & Communication Technologies*, IEEE, Jan. 2012, pp. 187–189. doi: 10.1109/ACCT.2012.2.

[11] U. S. Priyanka Shrivastava, "Error Detection and Correction Using Reed Solomon Codes," 2013.

[12] K. D. Irianto, G. T. Nguyen, H. Salah, and F. H. P. Fitzek, "Partial packet in wireless networks: A review of error recovery approaches," Jan. 17, 2020, *Institution of Engineering and Technology*. doi: 10.1049/iet-com.2019.0550.

[13] K. Jamieson and H. Balakrishnan, *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Digital Library, 2013.

[14] C. Fragouli, J.-Y. Le Boudec, and J. ¨ Org Widmer, "Network Coding: An Instant Primer." [Online]. Available: www.networkcoding.info

[15] C. Fragouli and E. Soljanin, "Network coding applications," 2007. doi: 10.1561/1300000013.

[16] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Foundations and Trends in Networking*, vol. 2, no. 1, pp. 1–133, 2007, doi: 10.1561/1300000003.

[17] S. Kafaie, Y. P. Chen, O. A. Dobre, and M. H. Ahmed, "Network Coding Implementation Details: A Guidance Document."

[18] R. Jain Download, "Network Coding for Wireless Applications: A review Network Coding for Wireless Applications." [Online]. Available: http://www.cse.wustl.edu/~jain/cse574-16/ftp/netcode/index.htmlhttp://www.cse.wustl.edu/~jain/cse574-16/ftp/netcode/index.html

[19] S. A. Alabady, "An efficient forward error correction code for wireless sensor networks," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 10, no. 2, 2021, doi: 10.11591/ijict.v10i2.pp104-115.

[20] D. Liu, X. Wang, J. Lu, and Y. Gao, "A practical routing protocol based on WiFi for resources-constrained opportunistic networks," *Journal of Information and Computational Science*, vol. 12, no. 2, 2015, doi: 10.12733/jics20105259.

[21] S. A. G. B. H. Bhadauria, "Performance Analysis of Ad hoc On-Demand Distance Vector Protocol for MANET," 2013.

[22] T. Abbas, F. Qamar, M. N. Hindia, R. Hassan, I. Ahmed, and M. I. Aslam, "Performance Analysis of Ad Hoc on-Demand Distance Vector Routing Protocol for MANET," in *2020 IEEE Student Conference on Research and Development (SCOReD)*, IEEE, Sep. 2020, pp. 194–199. doi: 10.1109/SCOReD50371.2020.9250989.

[23] Gurmeet Kaur, "PERFORMANCE ANALYSIS OF AODV ROUTING PROTOCOL IN MANETS," 2012.

[24] G. S. T. A. A. A. Salem, "Performance Analysis of Dynamic Source Routing Protocol," Dec. 2017.

[25] A. M. R. Boppana, "Analysis of the Dynamic Source Routing Protocol for Ad Hoc Networks," 2005.

[26] K. K. S. Kant, "Performance Analysis Of Dynamic Source Routing Protocol In Wireless Mobile Ad Hoc Network".

[27] S. Ahmad, I. Awan, A. Waqqas, and B. Ahmad, "Performance Analysis of DSR & Extended DSR Protocols," in *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, IEEE, May 2008, pp. 191–196. doi: 10.1109/AMS.2008.72.

[28] G. Fairhurst and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)," Aug. 2002. doi: 10.17487/rfc3366.

[29] J. Morris, "On Another Go-Back-N ARQ Technique for High Error Rate Conditions," *IEEE Transactions on Communications*, vol. 26, no. 1, pp. 187–189, Jan. 1978, doi: 10.1109/TCOM.1978.1093954.

[30] E. Weldon, "An Improved Selective-Repeat ARQ Strategy," *IEEE Transactions on Communications*, vol. 30, no. 3, pp. 480–486, Mar. 1982, doi: 10.1109/TCOM.1982.1095497.

[31] H. Bruneel and M. Moeneclaey, “On the Throughput Performance of Some Continuous ARQ Strategies with Repeated Transmissions,” *IEEE Transactions on Communications*, vol. 34, no. 3, pp. 244–249, Mar. 1986, doi: 10.1109/TCOM.1986.1096518.

[32] “Performance Analysis of Basic Automatic Repeat Request Protocols and Hybrid Automatic Repeat Request Protocol,” in *4th International Conference on Communication Engineering and Computer Science (CIC-COCOS’2022)*, Cihan University, 2022, pp. 233–238. doi: 10.24086/cocos2022/paper.567.

[33] BirdGard, “Como Afugentar Aves das Plantações,” BlogBirdGard.

[34] Biodiversity4All, “Castelo Branco - Pega,” Biodiversity4All.

[35] eBird, “Explore.”

[36] eBird, “Common Magpie (Eurasian),” eBird.

[37] Cornell Lab of Ornithology, “Eurasian Magpie Audio,” Cornell Lab of Ornithology. Accessed: Aug. 12, 2024. [Online]. Available: https://cdn.download.ams.birds.cornell.edu/api/v2/asset/340316981.mp3

[38] L. Shenzhen WEMOS Electronics Co., “D1 Mini Pro,” https://www.wemos.cc/en/latest/d1/d1_mini_pro.html. Accessed: Aug. 12, 2024. [Online]. Available: https://www.wemos.cc/en/latest/d1/d1_mini_pro.html

[39] L. Santos, T. Costa, J. M. L. P. Caldeira, and V. N. G. J. Soares, “Performance Assessment of ESP8266 Wireless Mesh Networks,” *Information (Switzerland)*, vol. 13, no. 5, May 2022, doi: 10.3390/info13050210.

[40] G. Mag, “painlessMesh API Documentation,” https://github.com/gmag11/painlessMesh/wiki/api.

[41] B. Blanchon, “ArduinoJSON: Efficient and Elegant JSON Serialization in C++ for Embedded Systems,” https://www.arduino.cc/reference/en/libraries/arduinojson/.

[42] A. Arkhipenko, “TaskScheduler: Cooperative Multitasking for Arduino,” https://www.arduino.cc/reference/en/libraries/taskscheduler/.

[43] dvarrel, “ESPAsyncTCP: Asynchronous TCP Library for ESP8266,” https://www.arduino.cc/reference/en/libraries/espasynctcp/.

[44] K. Söderby, “ESP32: A Guide to Using ESP32 with Arduino Cloud,” https://docs.arduino.cc/arduino-cloud/guides/esp32/?queryID=935fdd6506525ed3c494e3b64b016338.

[45] C. Liechti, “PySerial: Python Serial Port Access Library,” https://pythonhosted.org/pyserial/.