DAVI RIBEIRO MILITANI

APRIMORAMENTO DE UM ALGORITMO DE ROTEAMENTO BASEADO EM APRENDIZADO POR REFORÇO: UM ESTUDO DE CASO USANDO VOIP

LAVRAS – MG

2021

DAVI RIBEIRO MILITANI

APRIMORAMENTO DE UM ALGORITMO DE ROTEAMENTO BASEADO EM APRENDIZADO POR REFORÇO: UM ESTUDO DE CASO USANDO VOIP

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores, para a obtenção do título de Mestre.

Prof. DSc. Demóstenes Zegarra Rodríguez
Orientador

Prof. DSc. Hermes Pimenta de Moraes Júnior
Coorientador
Profa. DSc. Renata Lopes Rosa
Coorientadora

LAVRAS – MG
2021

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo próprio autor.

Militani,Davi Ribeiro
Aprimoramento de um Algoritmo de Roteamento Baseado em Aprendizado por Reforço: Um estudo de caso usando VoIP / Davi Ribeiro Militani. – Lavras : UFLA, 2021.
72 p. :

Dissertação(mestrado acadêmico)–Universidade Federal de Lavras, 2021.
Orientador: Prof. DSc. Demóstenes Zegarra Rodríguez.
Bibliografia.

1. Algoritmos de Roteamento. 2. Roteamento Inteligente. 3. VoIP. I. Rodríguez, Demóstenes Zegarra. II. Júnior, Hermes Pimenta de Moraes. III. Rosa, Renata Lopes. IV. Título.

DAVI RIBEIRO MILITANI

APRIMORAMENTO DE UM ALGORITMO DE ROTEAMENTO BASEADO EM APRENDIZADO POR REFORÇO: UM ESTUDO DE CASO USANDO VOIP

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores, para a obtenção do título de Mestre.

APROVADA em 28 de Janeiro de 2021.

Prof. DSc. Pedro Henrique Juliano Nardelli	School of Energy Systems
Prof. DSc Luiz Henrique Andrade Correia	UFLA

Prof. DSc. Demóstenes Zegarra Rodríguez
Orientador

Prof. DSc. Hermes Pimenta de Moraes Júnior
Co-Orientador

Profa. DSc. Renata Lopes Rosa
Co-Orientadora

LAVRAS – MG
2021

Dedico este trabalho a minha esposa Ana Carolina, aos meus pais e principalmente a Deus

AGRADECIMENTOS

Primeiramente agradeço a Deus que me possibilitou concluir este trabalho. Agradeço a minha esposa pelo constante apoio e compreensão e por ter sido meu pilar nesta caminhada. Agradeço aos meu pais pelo constante apoio e incentivo. Agradeço ao meu orientar Prof. DSc. Demóstenes Zegarra Rodríguez pela constante dedicação em me ajudar e pelo conhecimento a mim repassado. Ao meu coorientador Prof. DSc. Hermes Pimenta de Moraes Júnior por todo seu apoio fundamental para este trabalho, assim como a Profa. DSc. Renata Lopes Rosa minha coorientadora que foi essencial para início deste trabalho. Agradeço ao Instituto Federal do Sul de Minas Gerais pelo o apoio e pelo afastamento concedido que possibilitou a realização deste trabalho.

Eu, a sabedoria, moro com a prudência, e tenho o conhecimento que vem do bom senso. (Provérbios 8:12)

RESUMO

A capacidade do canal de transmissão, a capacidade de processamento dos roteadores e os algoritmos de roteamento são alguns dos principais fatores que impactam diretamente o desempenho de uma rede de computadores. Enquanto os parâmetros de rede, como perda de pacote, taxa de transferência e atraso, afetam a qualidade da experiência dos usuários em diferentes serviços multimídia. Os algoritmos de roteamento são responsáveis por escolher a melhor rota entre um nó de origem e um destino. Porém, algoritmos de roteamento convencionais não consideram o histórico de dados da rede na tomada de decisão sobre por exemplo, *overhead* ou falhas recorrentes nos equipamentos. Portanto, espera-se que algoritmos de roteamento baseados em aprendizado de máquina que utilizam o histórico da rede para tomada de decisão apresentem algumas vantagens. No entanto, um algoritmo de roteamento baseado na técnica de aprendizado por reforço (*Reinforcement Learning*, RL) pode necessitar de cabeçalhos de mensagens de controle adicionais. Nesse contexto, esta pesquisa apresenta um protocolo de roteamento aprimorado baseado em RL, denominado e-RLRP, no qual o *overhead* de mensagens de controle é reduzido. Especificamente, um ajuste dinâmico no intervalo da mensagem *Hello* é implementado para compensar o *overhead* gerado pelo uso de RL. Diferentes cenários de rede ad-hoc são implementados nos quais os parâmetros de desempenho da rede, como perda de pacotes, atraso, taxa de transferência e *overhead* são obtidos. Além disso, um cenário de comunicação *Voice Over* IP (VoIP) é implementado, no qual o algoritmo E-model é usado para prever a qualidade da comunicação. Para comparação de desempenho, são usados os protocolos OLSR, BATMAN e RLRP. Resultados experimentais mostram que o e-RLRP reduz o *overhead* da rede em relação ao RLRP e supera na maioria dos casos testados todos esses protocolos, considerando os parâmetros de rede e a qualidade de uma comunicação VoIP.

Palavras-chave: Algoritmos de Roteamento, Aprendizagem de Máquina, Roteamento Inteligente, VoIP, QoE.

ABSTRACT

The channel capacity, the routers processing capability, and the routing algorithms are some of the main factors that directly impact on network performance. Network parameters such as packet loss, throughput, and delay affect the users' quality–of–experience in different multimedia services. Routing algorithms are responsible for choosing the best route between a source node to a destination. However, conventional routing algorithms do not consider the history of the network data when making about, for example, overhead or recurring equipment failures. Therefore, it is expected that routing algorithms based on machine learning that use the network history for decision making present some advantages. Nevertheless, in a routing algorithm based on reinforcement learning (RL) technique, additional control message headers could be required. In this context, this research presents an enhanced routing protocol based on RL, named e-RLRP, in which the control message overhead is reduced. Specifically, a dynamic adjustment in the Hello message interval is implemented to compensate for the overhead generated by the use of RL. Different ad-hoc network scenarios are implemented in which network performance parameters, such as packet loss, delay, throughput and overhead are obtained. In addition, a Voice over IP (VoIP) communication scenario is implemented, in which E-model algorithm is used to predict the communication quality. For performance comparison, the OLSR, BATMAN and RLRP protocols are used. Experimental results show that the e-RLRP reduces network overhead compared to RLRP, and overcomes in most cases all of these protocols, considering both network parameters and VoIP quality.

Keywords: Routing Algorithms, Machine Learning, Intelligent Routing, VoIP, QoE.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

É notório que nos últimos anos a Internet cumpre um papel muito importante nas atividades cotidianas dos usuários. O seu uso, que antes era meramente para envio de e-mails, navegações em páginas estáticas expandiu-se para o uso em aplicações avançadas como Voz sobre IP (VoIP), *streaming* de vídeo e de áudio, jogos online entre outras. O serviço VoIP tem se tornado um dos serviços de comunicação mais populares, devido ao custo se comparado com a telefonia convencional e principalmente devido ao alto nível de qualidade de voz alcançado nos últimos anos (RODRÍGUEZ; MÖLLER, 2019). Porém, essas aplicações demandam cada vez mais enlaces de redes com maior largura de banda e técnicas para melhorar a robustez da transmissão de dados de forma a garantir a Qualidade de Experiência (*Quality of Experience*, QoE) (SANCHEZ-IBORRA; CANO; GARCIA-HARO, 2014) do usuário final.

Em uma rede de computadores, denominamos de nó os dispositivos que se comunicam entre si, e para que um nó transmita desde um arquivo de texto até um vídeo em alta definição os dados são acondicionados em pacotes e transmitidos do nó fonte até o destino, passando por nós intermediários. Entre o nó fonte e o destino podem existir N caminhos pelos quais um pacote pode passar e o processo que determina qual caminho será tomado é denominado de roteamento.

O protocolo de roteamento, é formado por: (i) Um conjunto de regras que definem a forma e a ordem de como as mensagens de controle são trocadas entre os nós, e como é feita a descoberta da topologia da rede; (ii) Um algoritmo de roteamento que calcula o caminho ótimo entre a fonte e o destino (KUROSE; ROSS, 2010). Apesar de serem eficazes em escolher rotas os algoritmos de roteamento atuais são incapazes de aprender com as experiências anteriores referentes a problemas na rede (TANG et al., 2018). Portanto, eles não conseguem identificar padrões de tráfego ou de comportamento dos nós. Sendo assim, é possível que o caminho determinado pelo algoritmo como o de melhor dentre os existentes possa ter um nó que tende a ficar sobrecarregado no futuro ou que se desliga de forma recorrente. Neste caso, o desempenho da rede seria afetado por essa escolha caso uma das duas situações citadas ocorra com um dos nós do caminho.

As redes de computadores podem se distinguir pelo meio físico utilizado para comunicação entre os nós, podendo ser sem fio ou cabeada. As sem fio podem ser de arquitetura infraestrutura ou ad-hoc (KUROSE; ROSS, 2010). Em uma rede infraestruturada existe um ponto principal por onde toda a comunicação da rede passa, tal ponto é geralmente denominado de estação base. Diferentemente a rede ad-hoc não necessita deste ponto principal, desta forma, toda comunicação ocorre somente entre os próprios nós da rede.

Apesar de terem como vantagem uma maior tolerância a falhas por não possuírem um único ponto de falha, a implementação de redes ad-hoc requer algoritmos de roteamento mais robustos e que

saibam lidar com constante possibilidade de desconexão dos nós da rede. Garantir um desempenho confiável nesse tipo de rede é um grande desafio devido às suas características (WANG; Lin, 2015).

Os protocolos de roteamento convencionais em redes ad-hoc, como o *Optimized Link State Routing* (OLSR) são incapazes de aprender com histórico de eventos que ocorrem na rede (TANG et al., 2018), podendo assim, escolher uma rota que no passado apresentava problemas recorrentes. Por exemplo, vamos considerar um caminho P onde um dado nó N apresenta desligamentos recorrentes devido a falhas de dispositivo ou desligamentos programados para economizar energia (NAWROCKI et al., 2020). Se um protocolo convencional escolher este caminho P, pode ocorrer degradação da rede, como perda de pacotes (HUANG et al., 2016). Um protocolo de roteamento capaz de aprender com o histórico da rede pode evitar esse caminho, melhorando assim o desempenho da rede. Sendo assim, é considerável que os protocolos de roteamento usem estratégias que os façam aprender com as experiências anteriores para escolher os caminhos de roteamento ideais (DING et al., 2019).

Nas últimas décadas os algoritmos de Aprendizagem de Máquina passaram a ser usados em diversas aplicações (ROSA et al., 2018; GUIMARÃES et al., 2016; LASMAR et al., 2019; MEMETI et al., 2018; POPA et al., 2019; ROSA; Rodriguez; Bressan, 2013; DHOUIB et al., 2014). Assim, surge a possibilidade que esses algoritmos possam também ser aplicados em protocolos de controle de roteamento (BANGOTRA et al., 2020; HUNG, 2020; SHIN; LEE, 2020), especificamente a Aprendizagem por Reforço (*Reinforcement Learning*, RL) que está cada vez mais sendo usada para melhorar o tráfego nas redes (WU et al., 2020; LI; BOUKHATEM; MARTIN, 2015; YANG et al., 2019). Na RL, existe um agente que deve ser capaz de aprender a como se comportar em um ambiente dinâmico por meio de interações com o mesmo (SUTTON; BARTO, 1998). As interações ou ações, podem proporcionar resultados bons o que gera recompensas para o agente ou resultados ruins o que gera punições. O agente deve buscar maximizar as recompensas optando por ações que geram as melhores recompensas. Assim, a técnica RL pode ser aplicada a algoritmos de roteamento para redes ad-hoc onde um nó da rede, que no caso é um agente, deve escolher uma rota que pode ser boa ou ruim. Ao aprender a escolher as melhores rotas o algoritmo de roteamento proporciona um ganho de desempenho da rede e consequentemente proporciona melhoria nas aplicações e serviços como a comunicação VoIP (COUTINHO et al., 2015).

Existem trabalhos que propõem o uso de RL para protocolos de roteamento como em (PESHKIN; SAVOVA, 2002), onde os autores apresentam um modelo genérico baseado em RL para redes ad-hoc com foco em estratégias de roteamento. Alguns trabalhos usam RL para roteamento em redes veiculares urbanas (*Urban Vehicular Ad-hoc Networks*, VANETs) como (WU et al., 2020). Outros trabalhos enfocam redes de sensores sem fio e suas características (WANG; SHIN, 2019) ou sistemas robóticos não tripulados (JUNG; YIM; KO, 2017).Em (TANG et al., 2018) é proposto um controle inte-

ligente de tráfego por aprendizado profundo, cujo os resultados demonstraram um ganho de desempenho em relação ao tradicional protocolo de roteamento *Open Shortest Path First* (OSPF). Já em (DING et al., 2019), o autor usa *Deep Reinforcement Learning* para desenvolver um novo protocolo de propósito geral, e obteve resultados superiores em comparação com OSPF. No entanto, ambos os trabalhos não focam em redes ad-hoc e não comparam o algoritmo desenvolvido com protocolos específicos para redes ad-hoc. Já em (DUGAEV et al., 2018) é proposto um protocolo de roteamento que utiliza RL, o *Reinforcement Learning Routing Protocol* (RLRP) que pode ser aplicado a redes ad-hoc e utilizado em ambiente real.

Os protocolos de roteamento requerem o uso de mensagens de controle para seu funcionamento, elas são utilizadas para descoberta de rotas e disseminação de informações sobre topologia. Porém, as mensagens de controle geram *overhead* na rede diminuindo assim a sua capacidade, principalmente em situações onde o canal de transmissão pode sofrer interferência ou ficar saturado. Uma das principais mensagens de controle é a *Hello*, que é responsável pela disseminação de informações sobre a vizinhança de cada nó da rede.

Existem pesquisas com o objetivo de diminuir o *overhead* originada por mensagens de controle. Em (MAHMUD; CHO, 2019), os autores propõem um ajuste no intervalo de envio de mensagens *Hello* do protocolo *Ad-hoc On Demand Distance Vector* (AODV) para *Flying Ad-Hoc Networks* (FANETs), com foco na redução do consumo de energia de veículos aéreos não tripulados (*Unmanned Aerial Vehicles*, UAVs).

Em (GIRUKA; SINGHAL, 2005) os autores propõem três algoritmos para ajustar o tempo de envio de mensagens *Hello*. O primeiro algoritmo é chamado de *Hello* Reativo, em que as mensagens de *Hello* são enviadas apenas quando o nó deseja enviar algum pacote. Em outras palavras, a descoberta da vizinhança é feita apenas quando o nó deseja enviar um pacote.

O segundo método é denominado *Hello* Baseado em Eventos, o ajuste é feito com base nos eventos que ocorrem na rede. Nessa abordagem, primeiro um nó da rede envia mensagens *Hello* com a frequência padrão mas, se após um período predefinido esse nó não receber nenhuma mensagem *Hello* de um vizinho ou não precisar enviar pacotes, ele para de enviar mensagens *Hello*. No terceiro método chamado *Hello* Adaptativo cada nó na rede envia um *Hello* após se mover por uma distância definida.

1.1 Objetivo

Como explicando anteriormente na RL o agente recebe uma recompensa pela ação tomada. Em (DUGAEV et al., 2018) as recompensas são enviadas aos nós por meio de mensagens de controle usando um cabeçalho exclusivo para recompensas, o que gera um *overhead* devido ao uso de RL. Esse *overhead* adicional aumenta o *overhead* global do algoritmo e pode afetar o desempenho da rede.

Neste contexto, objetivo deste trabalho é implementar uma melhora no algoritmo RLRP. Especificamente, essa melhora consistem em um mecanismo capaz de ajustar a frequência do envio de mensagens *Hello*. O algoritmo aprimorado é denominado de *Enhanced Reinforcement Learning Routing Protocol* (e-RLRP).

Este ajuste proporcionará uma redução no *overhead* total da rede com intuito de compensar o *overhead* causada pelas mensagens de controle necessárias para o uso de RL no RLRP. Por sua vez essa redução proporcionará um ganho de desempenho da rede.

1.1.1 Objetivos específicos

- Desenvolver um protocolo de roteamento aprimorado baseado na técnica RL, denominado e-RLRP, capaz de aprender com o histórico de eventos da rede, evitando rotas ruins e que além disso, seja capaz de reduzir o número de mensagens de controle. O algoritmo de roteamento baseado em RL é desenvolvido de acordo com (DUGAEV et al., 2018).

- Implementar um algoritmo que compense o *overhead* inserido pelas mensagens relacionadas ao algoritmo RL no RLRP. Até onde sabemos um algoritmo de ajuste dinâmico do intervalo de tempo da mensagem *Hello* não foi tratado por outros protocolos de roteamento baseados em RL.

- Comparar o desempenho do método proposto com outros protocolos de roteamento amplamente utilizados como o *Better Approach To Mobile Ad-hoc Networking* (BATMAN) e o *Optimized Link State Routing* (OLSR) além disso, comparar também com protocolo RLRP. Para tal, foram implementadas diferentes topologias de rede. A comparação de desempenho considera os principais parâmetros de rede como taxa de transferência, taxa de perda de pacotes e atraso. Além disso, a qualidade perceptual da fala em um serviço de comunicação VoIP também é avaliada, no qual dois modos de operação do codec de fala AMR-WB (3GPP TS 26.171, 2018) são usados.

Para implementar o e-RLRP e posteriormente validar o seu desempenho de acordo com os objetivos acima descritos, foram utilizados diversos cenários de rede ad-hoc com diversas configurações. Assim, diferentes cenários são implementados, considerando diferentes topologias de rede, número de nós e diferentes números de fluxos de tráfego. Para simular falhas na rede, alguns nós se desconectam em instantes aleatórios durante cada simulação.

Nestes cenários, o tráfego de VoIP é simulado e usado como um estudo de caso. Os parâmetros de redes obtidos são usados para avaliar o impacto do algoritmo de roteamento na qualidade perceptual de uma comunicação VoIP de acordo com o algoritmo E-model descrito na recomendação G.107.1 do ITU-T (ITU-T Rec. P.862.1, 2003). É importante observar que o serviço VoIP é usado como um caso

de estudo específico, mas o algoritmo de roteamento proposto é para fins gerais, sendo independente do aplicativo de serviço.

Os resultados experimentais de avaliação de desempenho mostram que o e-RLPR superou, na maioria dos cenários de teste usados neste trabalho os demais protocolos de roteamento usados para fins de comparação. O e-RLRP oferece uma redução de *overhead* de até 18% em comparação com o RLRP. O estudo de caso demonstra que o e-RLRP pode fornecer uma melhoria na qualidade da comunicação VoIP de mais de 90% se comparado ao OLSR e de até 8% se comparado ao RLRP.

1.2 Justificativa

Considerando o impacto dos algoritmos de roteamento sobre o desempenho da rede e o atual cenário de demanda por redes cada vez mais robustas, o uso da RL em algoritmos de roteamento torna-se um tema promissor. Um algoritmo inteligente capaz de se adaptar ao histórico de comportamento dos nós da rede pode proporcionar um ganho de desempenho da mesma. Essa ganho seria mais evidente em redes ad-hoc cuja topologia tende a se alterar, com os nós desligando de forma recorrente para economia de energia, por exemplo. Apesar de promissor, o uso de RL pode gerar *overhead* como citado. Esse *overhead* pode degradar a rede, principalmente em uma rede ad-hoc onde a largura de banda pode ser limitada e o canal de comunicação está muito mais suscetível a interferências. Sendo assim, espera-se que o uso de RL em protocolo de roteamento, concomitante com algoritmo que visa reduzir o *overhead*, proporcionará um ganho de desempenho da rede.

De acordo com (CETIC.BR, 2017) o uso de telefonia VoIP vem crescendo constantemente. Segundo (VOIP-INFO.ORG, 2019) em 2015 o VoIP representou aproximadamente 52% da participação no mercado de ligações internacionais. Considerando o alto número de usuários do serviço VoIP, o estudo do impacto de um algoritmo de roteamento inteligente em comunicações VoIP e o desenvolvimento de ajustes que busquem melhorias contribui de forma consistente para a melhoria da qualidade de experiência dos usuários nesse tipo de ligação.

Além disso, a solução apresentada neste trabalho é uma alternativa a algumas das atuais soluções aplicadas por empresas para melhoria do desempenho da rede que tem um considerável custo financeiro como: o investimento em equipamentos com melhor capacidade de processamento, maior capacidade de transmissão, entre outros. Pois, a solução do e-RLRP é aplicada a sistemas Linux, portanto poderia ser instalada em roteadores já em uso com *firmware* baseados em Linux como o OpenWRT (FAINELLI, 2008), proporcionando assim, um ganho de desempenho sem necessidade de troca de equipamentos.

1.3 Organização do trabalho

O presente trabalho está dividido da seguinte maneira. No Capítulo 2 é apresentando todo o referencial teórico onde são abordados temas relacionados ao presente trabalho, dentre eles: redes ad-hoc, algoritmos de roteamento, aprendizado de máquina e QoE. No Capítulo 3 são apresentando alguns trabalhos relacionados a este projeto. No Capítulo 4 é abordada a metodologia para o desenvolvimento do projeto, detalhando os métodos, as ferramentas utilizadas, os cenários elaborados e os testes para avaliação de desempenho. No Capítulo 6 é apresentando o algoritmo desenvolvido neste trabalho. No Capítulo 7 são tratados os resultados obtidos. E no Capítulo 7 encontra-se a Conclusão.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentadas as tecnologias e os conceitos abordados nesse trabalho. O capítulo está divido nos seguintes tópicos: Aprendizagem de Máquina, Algoritmos de Roteamento em Redes ad-hoc, Voz sobre IP e Qualidade de experiência.

2.1 Aprendizado de Máquina

Diante da impossibilidade dos atuais algoritmos aprenderem com o histórico de congestionamento de uma rede, surge a necessidade de buscar uma solução que permita o mesmo se adaptar a padrões de tráfego. Nesse contexto a Aprendizagem de Máquina (AM) se apresentada como um campo a ser explorado. Segundo (BISHOP, 2006) a AM é um campo de pesquisa que tem fundamento na Inteligência Artificial e que estuda o processo de aprendizagem. Segundo Mitchell (1997) a AM estuda métodos computacionais de forma a obter conhecimento e meios para organizar o conhecimento já existente. Segundo (MITCHELL, 1997) ela utiliza técnicas que possibilitam extrair conceitos a partir de uma amostra de dados. Duas vertentes da AM são abordadas a seguir: a Aprendizagem Supervisionada e a Aprendizagem por Reforço.

2.1.1 Aprendizado Supervisionado

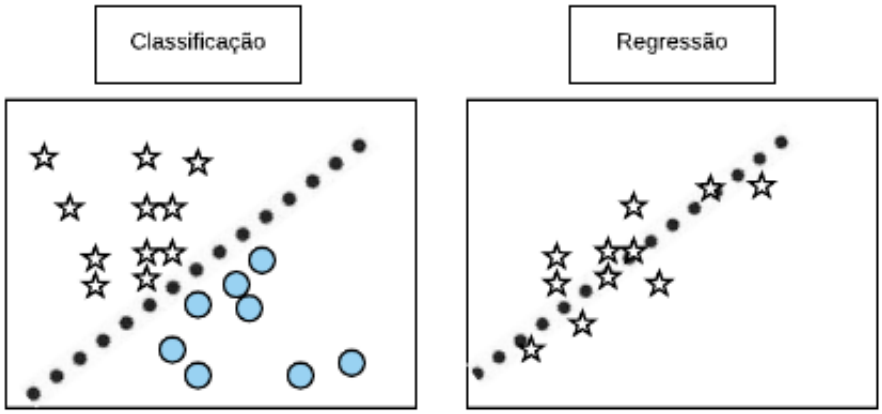
Segundo (THEODORIDIS, 2008) a Aprendizagem Supervisionada pode ser usada quando já se tem conhecimento do conjunto de dados, ou seja, o algoritmo é treinado por um conjunto de dados onde cada conjunto de entrada já possui uma saída definida.

No processo de aprendizagem supervisionada cada exemplo de treinamento é um conjunto de atributos geralmente descrito em forma de vetor, que serve como dados de entrada e estão associados a uma saída. A partir do treinamento, o algoritmo produz uma função de inferência e passa a ser capaz de produzir uma saída adequada a partir de uma entrada.

A aprendizagem supervisionada pode ser aplicada em problemas de classificação e regressão. No problema de classificação o algoritmo de aprendizagem supervisionada trabalha no mapeamento de um conjunto de entrada em categorias distintas. Por exemplo, dado as características de alguns tipos de animais como os répteis, mamíferos e anfíbios. Em um problema de classificação o algoritmo após ser treinado recebe como entrada as características de uma animal e o classifica como réptil, anfíbio ou mamífero. Já no problema de regressão o algoritmo busca mapear o conjunto de entrada para uma função contínua, como por exemplo definir a idade de um ser humano. Nesse caso, o algoritmo depois treinado recebe uma entrada com algumas características como por exemplo: a cor dos cabelos, traços do rosto

entre outras e busca definir uma idade para este ser humano. A Figura 2.1 ilustra os dois problemas de aprendizagem supervisionada.

Figura 2.1 – Problemas de Classificação e Regressão

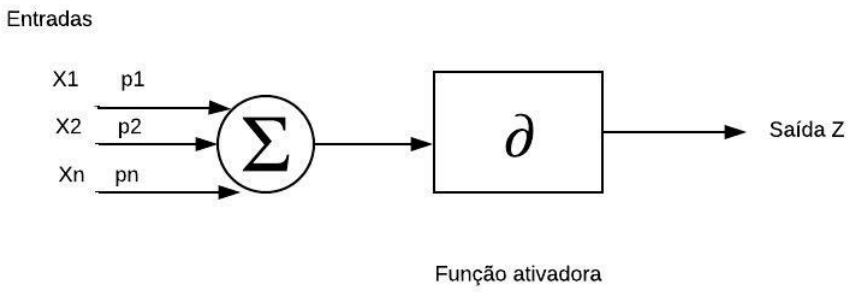


Fonte: Do autor (2021)

2.1.1.1 Redes Neurais

A capacidade do ser humano de aprender e realizar atividades complexas é possível devido a uma rede com mais de 100 bilhões de neurônios que forma o cérebro. Graças aos estudo que iniciaram desde os anos 40 em busca de um modelo computacional que funcione como os neurônios do cérebro foi possível a a criação de um modelo de neurônio artificial. Este modelo proposto está ilustrado de forma simplificada pela Figura 2.2.

Figura 2.2 – Neurônio Artificial



Fonte: Adaptado de (HAYKIN, 2007)

O neurônio é conjunto de N entradas ($X1, X2..Xn$) com seus respectivos pesos ($p1, p2..pn$) acrescido de um acumulador de sinais de entrada $\sum$ e uma função de ativação δ que limita o intervalo de amplitude do sinal de saída (z) a um valor fixo.

A junção de vários neurônios forma uma Rede Neural Artificial (RNA) que é um tipo de aprendizagem de máquina. A RNA tem por objetivo proceder de maneira similar as rede neuronais existentes em animais. A forma como as conexões entre os neurônios artificiais se comporta é simulado por meio de seus pesos. Um sinal enviado de um neurônio ao outro causa um efeito no receptor. Este efeito é determinado pela multiplicação do peso da conexão no neurônio receptor pelo valor do sinal recebido, podendo o peso ser positivo ou negativo simulando uma conexão excitatória ou inibitória respectivamente.

Existem vários tipos de RNA e elas se diferenciam por sua arquitetura e pela maneira como os seus pesos são associados. Segundo (HAYKIN, 2007) a arquitetura de uma RNA pode ser definida por: (I) Sua topologia; (II) Pelo número de camadas; (III) Pelo número de nós em cada camada; (IIII) Pelo tipo de conexão entre os nós. A estrutura de um RNA limita para qual tipo de problema ela pode ser utilizada.

2.1.1.2 Deep Learning

A união de vários neurônios forma as redes neurais artificiais. A *Deep Learning* (DL) é um avanço das redes neurais, formada por pilhas de camadas de neurônios onde a informação passa através de cada camada, sendo a primeira camada denominada de camada de entrada e a última de saída e as demais denominadas camadas ocultas. Cada camada gera uma saída que alimenta a outra camada influenciando assim o resultado final. Formada por várias camadas não lineares a DL é capaz de implementar funções extremamente complexas (LECUN; BENGIO; HINTON, 2015).

De forma ampla é possível dizer que diferentemente dos outros métodos de aprendizagem de máquina denominados rasos que utilizam um única função (camada) que recebe um conjunto de parâmetros e gera um resultado desejado. A DL é uma composição de camadas, ou seja, uma composição de funções conforme a Equação (2.1).

$$f(x) = f_C(...(f_2(f_1(x_1))...) \tag{2.1}$$

Sendo c o número de camadas, temos que cada função f_c recebe um entrada X_c portanto, a função f_1 recebe como entrada um vetor de dados X_1 e gera como saída o vetor X_{1+1} para a função f_{1+1}.

Cada função além de ter o seu vetor de dados de entrada para treinamento possui também parâmetros ou pesos que são ajustados durante o treinamento, sendo assim a Equação (2.1) deve ser reescrita

considerando estes parâmetros p. A Equação (2.2) descreve a composição de funções com as entradas e parâmetros de treinamento.

$$f(x) = f_C(...(f_2(f_1(x_1, p_1), p_2)...p_C) \tag{2.2}$$

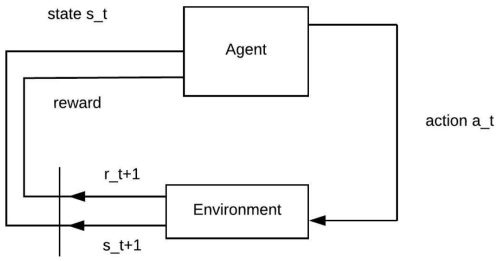
A DL funciona ajustando os parâmetros p a partir do conjunto de dados de entrada, sendo assim, cada representação é a combinação de outras anteriores mais simples (GOODFELLOW; BENGIO; COURVILLE, 2016). De forma geral, podemos dizer que cada função processa uma entrada e gera uma representação para próxima função. Se existir dados suficientes para o treinamento, ou seja, um número correto de camadas e parâmetros será possível então através das várias transformações dos dados de entrada separar as diversidades dos dados, podendo assim desembaralhar as informações.

2.1.2 Aprendizado por Reforço

O Aprendizado por Reforço (*Reinforcement Learning*, RL) é uma técnica de aprendizado de máquina onde existe um agente que interage com o ambiente por meio de ações e recebe recompensas pelas ações realizadas. O problema RL pode ser resumido da seguinte forma, um agente interage com um ambiente a fim de maximizar a recompensa acumulada ao longo do tempo (SUTTON; BARTO, 1998).

A generalização do processo de interação do RL (SUTTON; BARTO, 1998) é mostrada na Figura 2.3, onde o Agente interage com o Ambiente através de uma ação a_t. Essa interação leva a um novo estado s_{t+1} e gera uma recompensa para o Agente.

Figura 2.3 – Generalização do Processo de Interação do RL



Fonte: Adaptado de (SUTTON; BARTO, 1998)

As recompensas são enviadas para o Agente e definem se ação tomada foi boa ou ruim, que no último caso é na verdade um punição. Por meio das recompensas, o Agente estima a ação realizada, e esse conhecimento é então utilizado pelo Agente para adaptar as decisões futuras. As decisões futuras então, são controladas pelo valor de estimativa denominado Q. Em geral, podemos dizer que o valor de estimativa diz quão "bom"será optar por uma determinada ação.

A formulação do processo de otimização RL pode ser representada como um Processo de Decisão de Markov (MDP) (SUTTON; BARTO, 1998), introduzindo 4 conjuntos S, A, P, R, onde, S é um conjunto de possíveis estados do agente; A representa o conjunto de ações possíveis que um agente pode realizar; P é definido como um conjunto de probabilidades em que um agente no estado s avança para um estado s' ao optar por uma ação pertencente ao conjunto A. E finalmente a função recompensa R, que gera o reforço que o agente recebe pela escolha da ação A.

De acordo com o MDP, as probabilidades de transição de s para s' após realizar a ação a ($P_{ss'}{}^{a}$) podem ser descritas da seguinte maneira:

$$P_{\mathrm{ss'}}{}^{a} = Pr\left\{s_{\mathrm{t+1}} = s' \mid s_{\mathrm{t}} = s, a_{\mathrm{t}} = a\right\} \tag{2.3}$$

Os valores de recompensa de estimativa (E) para a ação a ($R_{ss'}{}^{a}$) do estado s ao s', são definidos a seguir:

$$R_{\mathrm{ss'}}{}^{a} = E\left\{r_{\mathrm{t+1}} \mid s_{\mathrm{t}} = s, a_{\mathrm{t}} = a, s_{\mathrm{t+1\,=\,s'}}\right\} \tag{2.4}$$

Os conjuntos S e P podem ser definidos como um conjunto de valores de estimativa Q, que é dependente do valor de recompensa obtido de um ambiente, e também do momento atual t, quando a ação correspondente foi realizada. A função de valores de estimativa é apresentada da seguinte forma:

$$Q_{k+1} = Q_k + \alpha * [r_{k+1} - Q_k] \tag{2.5}$$

onde Q_{k+1} representa o novo valor da estimativa; Q_k é o valor da estimativa atual; r_{k+1} define o valor da recompensa por uma ação realizada na etapa atual; α representa o parâmetro de tamanho do passo; e k é o número da etapa atual. Dessa forma, é possível calcular o valor de estimativa para um dada ação.

Um dos problemas do RL é definir se o Agente deve aproveitar as ações atuais que geram até o momento as maiores recompensas ou explorar novas ações na possibilidade obter recompensas ainda melhores. Para maximizar as recompensas recebidas, o Agente então deve equilibrar a necessidade de explorar novas ações ou aproveitar as atuais.

No RL existem alguns métodos para decidir entre explorar novas ações ou aproveitar as atuais. Os métodos mais comuns para seleção de ação são os métodos *greedy*, e-greedy e o Softmax (KOULOURIOTIS; XANTHOPOULOS, 2008). A seleção *greedy* opta pela ação com o valor máximo de estimativa o tempo todo. A seleção e-greedy opta pela ação com a estimativa máxima quase todas as vezes, entretanto, às vezes explora novas ações ao acaso. O método Softmax (ASADI; LITTMAN, 2017) fornece uma mudança dinâmica das probabilidades de seleção das ações. Essa mudança nas probabilidades de

seleção das ações ocorre de acordo com uma função de probabilidade predefinida, como a distribuição de Gibbs-Boltzmann (SUTTON; BARTO, 1998).

2.2 Algoritmos de Roteamento em Redes Ad-hoc

A internet é a maior rede de computadores do mundo e se tornou um dos maiores meios de comunicação existente. Ela se encontra presente nas empresas, nas residências, nos carros, nos celulares entre tantos outros lugares e dispositivos.

Uma rede de computadores é formada por diversos dispositivos que se comunicam entre si, que são denominamos de nó. Os nós estão interligados e para que um dado nó se comunique com um outro muitas vezes se faz necessário que a informação trafegue através nós intermediários. Porém, podem existir *N* caminhos diferentes, sendo necessário a escolha de um dentre os existentes. O processo de escolha do caminho denomina-se roteamento.

Segundo (KUROSE; ROSS, 2010) para que a escolha do caminho seja feita é necessário um protocolo de roteamento que é formado por: (i) Um conjunto de regras que definem a forma e a ordem de como as mensagens de controle são trocadas entre os nós e como é feita a descoberta da topologia da rede; (ii) Um algoritmo de roteamento que calcula o caminho entre a fonte e o destino (KUROSE; ROSS, 2010).

A finalidade de um algoritmo de roteamento é a escolha do 'bom' caminho entre um nó fonte e o nó de destino. Normalmente um 'bom' caminho é aquele que tem o 'menor custo' (KUROSE; ROSS, 2010). Existem diversos algoritmos de roteamento, mas todos tem como objetivo encontra o caminho de menor custo de acordo com um métrica definida.

Todos os nós de uma rede de computadores possuem a sua tabela de roteamento, que de forma geral pode ser definida como uma tabela que possui informações sobre a topologia da rede. Essa tabela é construída e atualizada através da troca de mensagens entre os nós da rede. Segundo (KUROSE; ROSS, 2010) os dois tipos principais de protocolos de roteamento são *Link State* e o *Distance Vector*. Os dois se diferem na forma com disseminam a informação sobre mudanças na topologia da rede. No *Link State* quando um nó percebe uma mudança na topologia ele atualiza sua tabela e envia essa informação para os nós da rede. Diferentemente o *Distance Vector* ao tomar ciência de uma mudança na topologia atualiza sua tabela e a informação é disseminada apenas para os nós vizinhos, que por sua vez atualizam suas tabelas e reenviam a informação para os seus vizinhos. Esse processo acontecem de forma sucessiva até que todos os nós tenham a informação.

Os protocolos de roteamento podem ser divididos em três subclasses:

- Protocolos reativos: (DHURANDHER; OBAIDAT; Gupta, 2010) que trocam informações de topologia sob demanda. Nesse tipo de protocolo, a troca de informações sobre a topologia ocorre apenas quando um nó deseja enviar uma mensagem. No protocolo reativo, a redundância na transmissão de mensagens de controle é menor em relação a outros tipos de protocolos;

- Protocolos proativos: (MBARUSHIMANA; SHAHRABI, 2007) que atualizam continuamente as informações da rota enviando mensagens de controle. Nesse tipo de protocolo, a troca de informações entre os nós sobre a topologia da rede ocorre antes mesmo do nó enviar qualquer pacote. Os protocolos proativos geralmente fornecem maior flexibilidade na seleção de rotas em comparação com os reativos. No entanto, ele produz um número maior de mensagens de controle, o que aumenta o *overhead*;

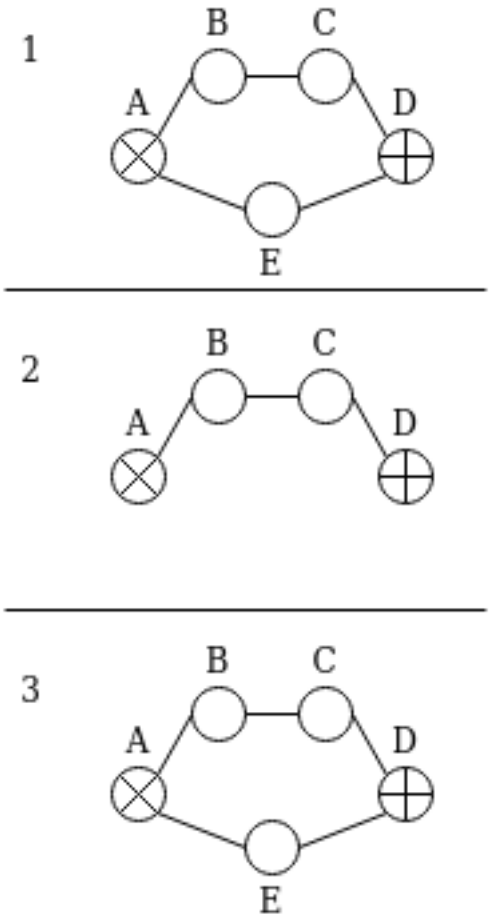
- Protocolos híbridos: (RAMASUBRAMANIAN; HAAS; SIRER, 2003) que combinam características do proativo e do reativo. Em protocolos híbridos, algumas rotas são criadas de forma pró-ativa e posteriormente o protocolo funciona de forma reativa.

Em protocolos de roteamento a definição de caminho de menor custo depende do tipo de métrica utilizada. Existem várias métricas, uma delas é a número de saltos. Na número de saltos define-se que todos os nós tem peso igual portanto, neste caso o caminho de menor custo é aquele que tem um número menor de nós, ou saltos. Um problema dessa métrica é a possibilidade do algoritmo optar sempre por uma rota que apesar de possuir menor número de nós esteja congestionada ou que possuem nós que apresentem falhas ou se desliguem de forma recorrente.

A Figura 2.4 ilustra uma situação em que considerar apenas o número de saltos na escolha de uma rota degrada o desempenho da rede. Na figura um dado nó A deseja enviar um pacote para um dado nó D. Utilizando-se a métrica número de saltos a rota escolhida pelo algoritmo é o que passa pelo nó E, no momento 1 então os pacotes são trafegados pela rota AED. No momento 2 o nó E se desliga e ocorre uma perda de pacotes, o algoritmo então converge para rota ABCD e envia pacotes por ela. Porém, no momento 3 o nó E se religa e após o algoritmo reconhecer a rota AED, novamente então, ele passa a optar por ela por ter menor número de saltos. Se o nó E se desligar de forma recorrente os passos citados anteriormente se repetiram, gerando assim um grande perda de pacotes, o que degradaria desempenho da rede.

A outra métrica é a *Expected Transmission Count* (ETX) (COUTO et al., 2003) que avalia o número de transmissões necessárias para entregar um pacote entre um nó fonte e um nó destino. De forma geral as rotas que necessitam um menor número de transmissões são escolhidas. O peso de um enlace entre um dado nó X e um no Y que estão diretamente ligados é o inverso da probabilidade (P) de

Figura 2.4 – Cenário de Degradação Devido uma Desconexão



Fonte: Do autor (2021)

sucesso de transmissão de um pacote entre X e Y. A equação a seguir descreve o cálculo do peso desse enlace.

$$ETX_{xy} = \frac{1}{P_{xy}} \tag{2.6}$$

O cálculo da probabilidade (P) é feito da seguinte forma:

$$P = \frac{E}{S} \tag{2.7}$$

onde E é o número de pacotes enviados e S é o número de sucessos obtidos no envio dos pacotes.

Em uma rede com três nós onde o nó X está ligado diretamente a Y que por sua vez está ligado diretamente a Z, o peso da rota entre X e Z é o somatório do peso de X para Y e Y para Z. Dessa forma, o cálculo do ETX primeiramente é feito entre nós vizinhos e cada nó tem o valor do ETX do seu vizinho. Para se calcular o custo de uma rota inteira soma-se o valor do ETX de todos os nós que pertencem a

rota. A melhor rota será aquele que soma dos ETX gere um menor valor. A métrica ETX considera além do tráfego na rede o número de saltos, uma vez que em um cenário que todos os nós possuem o mesmo valor do ETX a rota com um número menor de nós resultara um somatório de valor menor.

Como exposto anteriormente diversos problemas podem afetar o funcionamento de uma rota, por exemplo um nó pode apresentar falha de comunicação ou ficar sobrecarregado, e o algoritmo de roteamento deve ser capaz de lidar com as alterações na topologia sem interromper toda a comunicação da rede (TANENBAUM, 2003). Os algoritmos de roteamento convencionais são capazes de encontrar um novo caminho de menor custo em caso de falha de um dos nós do caminho antigo, porém eles são incapazes de aprender com padrões de congestionamento de enlaces e ou de comportamento dos nós. Os algoritmos que utilizam a métrica ETX por exemplo, são capaz de evitar caminhos congestionado mas, são incapazes de aprender com a tendencia de um dado nó de estar congestionado ou se desligar. Nesse contexto, um algoritmo capaz de aprender com o comportamento da uma rede, evitando em algumas ocasiões caminhos que tem tendência a estarem congestionado ou que possuem nós que se desligam de forma recorrente contribuiria significativamente para a melhoria do *throughput* da rede.

2.2.1 Better Approach to Mobile Ad-hoc Networking (BATMAN)

O BATMAN (SANCHEZ-IBORRA; CANO; GARCIA-HARO, 2014) é um protocolo de roteamento pró-ativo para rede ad-hoc. Ele utiliza uma abordagem diferenciada para compartilhar o conhecimento sobre os melhores caminhos. Basicamente, cada nó possui informações sobre qual vizinho de um salto tem a melhor rota para um determinado destino X, ou seja, qual vizinho deve ser escolhido quando se deseja enviar um pacote para o nó X.

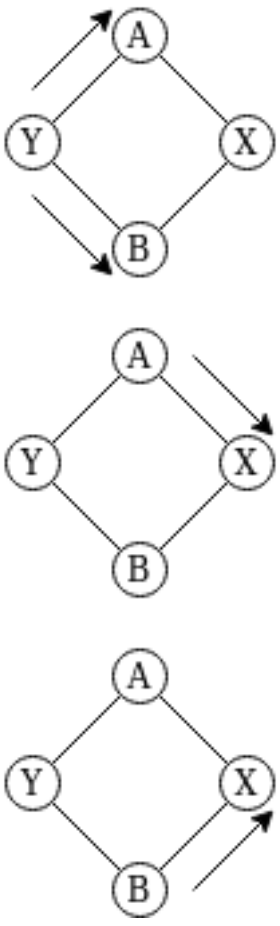
No BATMAN cada nó da rede envia uma mensagem denominada *OriGinator Messages* (OGMs), a todos os seus vizinhos para informar de sua existência. Os OGMs são pequenas mensagens que contêm o endereço do nó de origem, o endereço do nó que retransmitiu, um *Time To Live* (TTL) e um número de sequência para registrar a rota já percorrida pelo pacote. Quando um nó da rede recebe uma mensagem OGM ele atualiza sua tabela de roteamento, diminui o TTL e aumenta o campo de sequência. Depois disso, ele reencaminha a mensagem para seus vizinhos, esse procedimento é repetido até que todos os nós da rede recebam a mensagem.

O BATMAN usa a troca de mensagens OGMs para influenciar na escolha das rotas, basicamente isso acontece da seguinte forma: Quando um nó X na rede recebe a mesma OGM de um emissor Y por dois caminhos diferentes ele descarta a última mensagem e considera apenas a primeira mensagem. A ideia é que o OGM que chegou primeiro provavelmente fez o melhor trajeto.

O nó X então registra qual vizinho de um salto emitiu o OGM que chegou primeiro. Este vizinho é definido como o melhor caminho para uma rota possível para o emissor Y. Quando os OGMs passam por rotas ruins geralmente são perdidos ou demoram mais para chegar, portanto, o nó somente considerará OGMs de rotas boas. Dessa forma, o BATMAN define qual é a melhor rota entre o nó X e o emissor Y.

A Figura 2.6 ilustra como funciona a escolha de rota com o mecanismo da OGM. O nó Y envia sua OGM para seus vizinhos A e B. Os vizinhos A e B retransmitem a mensagem para seu vizinho, o nó X. Suponha-se que nó B esteja sobrecarregado e devido a isto sua OGM demore a ser enviada e por causa desse atraso a OGM retransmitida pelo nó A seja recebida primeiro pelo nó X. O nó X considerará então o vizinho A como melhor rota para o nó Y.

Figura 2.5 – Envio de OGMs



Fonte: Do autor (2021)

Outro mecanismo importante do BATMAN é o sistema de inundação seletiva que funciona da seguinte maneira: Quando um nó recebe um OGM além de retransmitir o OGM recebido para os vizinhos, ele também responde ao nó de origem com outra mensagem OGM. Porém, ele não envia a mensagem

em brodcast, ele primeiro consulta em sua tabela qual vizinho tem a melhor rota para o nó fonte e envia apenas para este vizinho. Portanto, as mensagens são enviadas de forma seletiva, o que diminui o *overhead* das mensagens de controle.

2.2.2 Optimized Link State Routing (OLSR)

O OLSR (CLAUSEN et al., 2003) é um protocolo que foi desenvolvido para redes ad-hoc. Ele é pró-ativo, ou seja, as mensagens de controle são trocadas de forma constante independente da ocorrência de tráfego de dados. Além disto, ele é um protocolo *Link State* portanto as informação de mudança na topologia da rede são enviadas para todos os nós da rede e não somente para os vizinhos.

No OLSR cada nó possui uma tabela de roteamento com as informações sobre o nó de destino, próximo salto, quantidade de saltos e endereço da interface local. A construção da tabela de roteamento é feita com o algoritmo de Dijkstra e o OLSR utiliza como métrica o número de saltos.

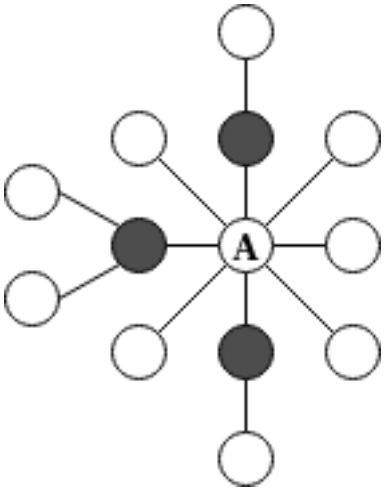
Em rede ad-hoc a topologia da rede tende a a mudar, alguns nós podem se locomover ou se desligar. Portanto é necessário uma constate troca de informações entre os nós. O OLSR troca informações sobre o estado dos enlaces em intervalos de tempos definidos por meio das mensagens *Topology Control* (TC) e *Hello*. A mensagem TC tem como objetivo difundir informações sobre a topologia da rede. Já mensagem *Hello* tem como objetivo detectar a presença dos nós vizinhos, divulgar informações sobre o estado dos links e sinalizar os *MultiPoint Relays* (MPRs).

O MPRs é um mecanismo do OLSR responsável por reduzir o número de mensagens de controle reduntantes como a TC e a *Hello*. O MPRs é uma das principais vantagens do OLSR pois contribui para reduzir o *overhead* de mensagens controle da rede. O conceito desse mecanismo, é selecionar dentre os nós da rede quais podem retransmitir as mensagens de controle. Quando um nó da rede envia mensagens de controle, somente os vizinhos do conjunto MPRs podem retransmitir, reduzindo assim o número de mensagens redundantes na rede, o que consequentemente pode melhorar o *throughput* da rede. Segundo (CLAUSEN et al., 2003) quanto menor for o conjunto de MPRs de um nó menor será o tráfego de controle.

A escolha dos nós MPRs ocorre de acordo com a seguinte regra: (i) Os nós MPRs de um dado nó *A* é um subconjunto do conjunto de vizinhos de um salto do nó *A*. (ii) Os nós MPRs estão a um salto dos vizinhos de dois saltos do nó *A*. Para melhor entendimento a Figura 2.6 ilustra o conjunto de nós MPRs do nó A.

Na figura 2.6 os preenchidos de preto são os nos MPRs. Podemos verificar que eles estão a um salto do nó A, satisfazendo assim a primeira regra. Além disto, eles estão a um salto dos nós que estão a dois saltos do nó A, satisfazendo assim a segunda regra.

Figura 2.6 – MPRs



Fonte: Do autor (2021)

2.2.3 Reinforcement Learning Routing Protocol (RLRP)

O RLRP é um protocolo de roteamento reativo para redes multi-hop ad-hoc. Em geral, o objetivo do RLRP é tomar uma decisão sobre o envio de pacotes com base em valores de estimativa. Esses valores são atualizados dinamicamente por meio do mecanismo de recompensas usado pela RL. O RLRP funciona em sistemas Linux com a pilha TCP/IP fornecendo roteamento para quaisquer pacotes de dados com endereçamento IPv4 ou IPv6 (DUGAEV et al., 2018). O processo de roteamento começa após inicializar o *daemon* de roteamento e é executado em uma interface virtual criada.

O RLRP, como qualquer outro protocolo convencional para redes ad-hoc multi-hop, é baseado em dois modos operacionais. A primeira é a descoberta de caminho, que ocorre quando um nó precisa enviar um pacote e não tem informações de roteamento para um destino. O segundo é o encaminhamento de pacotes, que é quando um protocolo decide qual rota é a melhor para enviar o pacote.

No primeiro modo, o RLRP usa a abordagem reativa. Assim, um nó de origem *A* envia uma mensagem de solicitação de rota (RREQ) para seus vizinhos diretos e os vizinhos, por sua vez, retransmitem esse RREQ para seus vizinhos. Desta forma, a mensagem RREQ é encaminhada para todos os nós da rede até que o contador de tempo de vida da transmissão (TTL) seja alcançado ou até que um nó que já tenha enviado este RREQ receba a mensagem novamente. Todos os nós da rede que participaram da retransmissão do RREQ obtêm informações de rota para o nó de origem e atualizam suas tabelas de roteamento com essas informações. O nó de destino *B* que recebe a RREQ envia uma mensagem de resposta de rota (RREP) que passa pelo mesmo processo de retransmissão. Os nós vizinhos de *B* e todos os nós participantes da retransmissão do RREP atualizam sua tabela de roteamento com informações de caminho para alcançar o nó *B*. Quando o nó *A* receber o RREP enviado pelo nó *B*, todos os nós da rede

já estarão cientes das rotas entre *A* e *B*. Assim, o processo de descoberta de caminho termina e o modo de encaminhamento de pacote pode ser iniciado.

Os protocolos de roteamento convencionais têm em sua tabela de roteamento um campo com informações de endereço de destino. Cada rota está associada a um custo que é calculado de acordo com uma métrica específica, dessa forma o caminho selecionado é o que tiver o menor custo. Diferentemente, o RLRP usa RL para decidir qual caminho é o melhor.

Conforme explicado na subseção 2.1.2 na aprendizagem por reforço existe um agente e conjunto de ações que o agente pode realizar. A ação de cada agente gera uma recompensa, e desta forma o agente cria um conjunto de estimativas para as ações, ou seja, um conjunto que informa o quanto "bom"será a recompensa para um determinada ação. Esse processo do RL pode ser usado na tarefa de roteamento. Para melhor associação, a Tabela 2.1 apresenta uma relação entre a aprendizagem por reforço e os protocolos de roteamento convencionais.

Tabela 2.1 – Relação entre RL e os Protocolos de Roteamento Convencionais

Tarefa do RL	**Tarefa de Roteamento**
Agente	Nó Fonte
Conjunto de ações	Conjunto de vizinhos a escolher
Conjunto de valores de Estimativa (Q)	Tabela de Roteamento
Ação do Agente	Enviar um pacote
Agente Recebe Recompensa	O nó recebe um mensagem ACK

Fonte: Do autor (2021)

A tabela anterior explica uma relação entre o mecanismo/tarefas de roteamento e os mecanismos de Aprendizado por Reforço. Através deste relacionamento é possível aplicar o RL à tarefa de roteamento. Assim, um nó *X* da rede que utiliza o protocolo RLRP pode ser considerado um Agente.

O Conjunto de Ações é o conjunto de nós da rede em que *X* pode enviar mensagens. O envio de um pacote pelo nó X para um determinado outro nó da rede é uma ação do Agente. E ao enviar este pacote, o nó X espera receber uma *Acknowledgment Message (ACK)*, caso isso aconteça significa que o pacote atingiu o nó determinado, ou seja, foi gerada uma recompensa por ter escolhido este nó para enviar a mensagem. Se o ACK não for recebido, significa que a mensagem foi perdida e a rota é ruim, então o nó X recebe uma punição pela ação escolhida.

E finalmente, a tabela de roteamento de um protocolo define a melhor rota para enviar um pacote a um determinado destino. Da mesma forma, um Conjunto de Estimativas define qual ação gera a melhor recompensa.

2.2.3.1 Propagação de Recompensa através de Acknowledgment Message (ACK)

O valor da recompensa está diretamente relacionado ao recebimento do ACK. Quando um nó deseja enviar um pacote a um determinado destino, ele seleciona um vizinho dos existentes e envia o pacote a esse vizinho. Em seguida, aguarda a mensagem ACK do nó correspondente, que contém metainformações sobre o pacote recebido, e o valor da recompensa pela ação de escolha deste vizinho.

Se o ACK não for recebido dentro de um tempo pré-definido, então o nó emissor define uma punição, ou seja, uma recompensa negativa para o nó vizinho para o qual o pacote foi encaminhado. Este valor negativo é definido no RLRP como -1. Se o ACK não estiver sendo recebido, provavelmente o nó vizinho ficou fora de alcance, seja por que vizinho pode estar enfrentando problemas de hardware, como queda de energia ou forte interferência com a transmissão sem fio, o nó estar sobrecarregado com tráfego de entrada ou ele se moveu para fora do alcance. Portanto, é consistente que esse vizinho seja evitado no futuro.

Se a mensagem ACK for recebida dentro do prazo, um valor de recompensa será fornecido na mensagem. Se o valor for alto significa que o vizinho tem um bom caminho até o destino, a probabilidade de escolher este vizinho no futuro aumentará. Se o valor for baixo significa que o vizinho escolhido não tem uma boa rota para aquele destino, seja porque está com problemas de hardware ou por haver muitos saltos até o nó destino ou a qualidade dos demais links é ruim. Nesse caso, o nó de origem diminuirá lentamente o valor de estimativa para este vizinho, o que provavelmente fará com que o nó escolha posteriormente outros vizinhos.

2.2.3.2 Geração de Recompensa

O mecanismo de ajuste do valor da recompensa deve ser flexível, ou seja, o ajuste não pode ser muito pequeno que não provoque mudanças ou muito grande a ponto de induzir mudança repentina devido a um evento específico. Por exemplo, se o valor da punição após a escolha de uma rota ruim for muito baixo, o valor estimado dessa rota irá diminuir lentamente e provavelmente esta rota ruim ainda pode ser escolhida por um longo tempo. Por outro lado, se o valor de punição for muito alto, uma rota pode não ser mais escolhida devido a apenas um evento de perda de pacote. Portanto, um equilíbrio deve ser encontrado entre recompensas / punições altas e baixas.

De acordo com (DUGAEV et al., 2018), o valor da recompensa é calculado da seguinte forma: Quando um nó X recebe um pacote do nó Y, um ACK é enviado com o valor da recompensa para Y. Para calcular o valor da recompensa, soma-se os valores de estimativa que cada vizinho do nó X possui em relação ao nó de destino Y, que são denominados como $Q_{dst_{ip}}$, o resultado então é dividido pelo número

correspondente de vizinhos (N). Assim, recompensa$_{\text{valor}}$ é a média dos valores Q dos vizinhos em relação ao nó Y. O recompensa$_{\text{value}}$ é calculado de acordo com a equação a seguir:

$$\text{recompensa}_{\text{valor}} = \sum Q_{\text{dst}_{\text{ip}}}/N \tag{2.8}$$

Ao receber o recompensa$_{\text{value}}$, o nó Y ajusta o valor de estimativa para o nó X. No entanto, se o ACK não for recebido, o nó Y define automaticamente o valor da recompensa para -1, ou seja, é gerada uma punição que impacta negativamente o valor de estimativa. O valor da estimativa é definido a seguir.

2.2.3.3 Cálculo do Valor de Estimativa

Um valor inicial deve ser definido para cada nó quando o protocolo é iniciado, o que geralmente é chamado de partida a frio. O RLRP define inicialmente todos os vizinhos com um valor de 0 quando um nó de origem não tem informações de rota para um nó de destino. O intervalo disponível de valores estimados é definido como: [0, 100]. Quando o protocolo inicia o processo de descoberta de rota, os valores estimados são definidos da seguinte forma:

$$Q_{\text{n}} = 100/N_{\text{hops}} \tag{2.9}$$

onde Q_{n} é o valor estimativa para o IP de destino; N_{hops} é o número de saltos nos quais as mensagens RREQ ou RREP passaram do nó origem para o destino.

Depois que o procedimento de descoberta de caminho termina, os nós da rede têm os valores iniciais estimados para as rotas. De acordo com o cálculo apresentado na Equação (2.9), o valor da estimativa é inicialmente definido com base no número de saltos entre a origem e o destino. Então, pode-se definir que o RLRP usa como abordagem inicial a métrica de número de saltos, na qual as rotas com o menor número de salto são escolhidas.

Porém, posteriormente os valores devem ser ajustados, pois a rota com o menor número de saltos nem sempre é a melhor. Pois, uma rota pode ter o menor número de saltos, mas apresentar um link sobrecarregado ou ter nós que apresentam mau funcionamento. O ajuste então é feito de acordo com o valor da recompensa recebida. O valor de estimativa Q como descrito em 2.1.2 é calculado da seguinte forma:

$$Q_{k+1} = Q_k + \alpha * [r_{k+1} - Q_k] \tag{2.10}$$

onde Q_{k+1} representa o novo valor de estimativa que será calculado para uma dada ação; Q_k é o valor estimativa atual para a mesma ação; r_{k+1} define o valor da recompensa obtida; α representa o parâmetro de tamanho do passo; e k é o número da etapa atual. Portanto, o valor estimado conforme descrito acima é impactado pelo valor da recompensa.

No e-RLRP a recompensa está associada à entrega bem-sucedida de pacotes. Então, em geral, a melhor recompensa local é gerada pela rota que tem a melhor taxa de sucesso na entrega de pacotes, e a recompensa de longo prazo está relacionada ao ganho de desempenho da rede global, quando o algoritmo encontra rotas com as maiores taxas de sucesso global. Conforme explicado na subseção 2.1.2, o algoritmo RL deve saber balancear entre a seleção de ações que obtêm no dado momento os valores de recompensa mais altos e explorar novas ações que podem gerar recompensas ainda melhores, com objetivo de assim alcançar as recompensas de longo prazo. Para esta tarefa de decisão, o e-RLRP usa o método Softmax (LUO et al., 2020) baseado na distribuição Gibbs/Boltzmann, definido pela formula a seguir:

$$P_{sa} = \frac{e^{\frac{Q_{sa}}{\tau}}}{\sum_{i=1}^{n} e^{\frac{Q_{si}}{\tau}}} \tag{2.11}$$

onde P_{sa} é a probabilidade de escolher uma ação a na etapa s; Q_{sa} é a estimativa da ação a na etapa s; Q_{si} é a estimativa para uma ação alternativa i na etapa s; e τ é o parâmetro de temperatura. O Softmax busca balancear entre a ação com maior recompensa a ou explorar um ação alternativa b. O parâmetro de temperatura ajusta a probabilidade de explorar novas ações ou aproveitar a ação que geral maior recompensa no momento.

2.3 Comunicação VoIP

Basicamente a VoIP transforma sinais analógicos em digitais possibilitando aos mesmos trafegar em uma rede IP. Os estudos sobre tráfego de voz em redes de pacotes se iniciaram na década de 80, o objetivo inicial era fazer uma integração da rede de dados com a rede de telefonia. Apesar ter iniciado nos anos 80 somente em 1998 foi publicado a primeira versão de conjunto de protocolos para padronizar o uso de voz e vídeo em redes IP. Porém, esta primeira versão era complexa demais o que encareceu a implantação e atrapalhou o projeto. Mas em 1999 foi desenvolvido o *Session Initiation Protocol* (SIP), um protocolo mais leve e menos complexo que foi responsável pelo crescimento do uso do VoIP (HANDLEY et al., 1999).

O serviço de telefonia convencional funciona através da comutação de circuitos diferentemente do serviço de telefonia VoIP que se baseia em comutação de pacotes e funciona sobre uma rede IP

(HARTPENCE, 2013). Na comutação de pacotes o meio físico de comunicação é compartilhado, desta forma vários dispositivos podem usar acesso o meio ao mesmo tempo, já a telefonia convencional utiliza a comutação de dados que ao estabelecer a comunicação entre dois dispositivos ocupa o meio de comunicação não podendo este ser usado por outros dispositivos enquanto a comunicação não for encerrada.

2.3.1 Arquitetura da telefonia IP

Para funcionamento da telefonia IP são necessários um protocolo de sinalização, responsável pelo processo de transmissão, e um protocolo de transporte, que tem como objetivo transmitir os sinais de voz e/ou vídeo e dados (HARTPENCE, 2013).

O protocolo de sinalização mais utilizado e que foi o responsável pelo crescimento do uso do VoIP é o SIP (HANDLEY et al., 1999), por não ser proprietário e por ser o mais suportado pelos fabricantes de tecnologia VoIP, além de ser mais fácil de usar do que os demais. Os outros dois importantes protocolos são o *Skinny* pertencentes a Cisco e o H.323 da ITU-T, sendo este dois juntamento com SIP os três mais utilizados (HARTPENCE, 2013).

O protocolo SIP foi desenvolvido em um único módulo e projetado para operar juntamente com as demais aplicações existentes na Internet. Quando dois dispositivos VoIP desejam se conectar existe a necessidade criar a comunicação e de estabelecer regras para funcionamento da mesma e quem é responsável por isto é o protocolo de sinalização. Segundo (HARTPENCE, 2013) no caso SIP especificamente além do próprio existe também o *Session Description Protocol* (Protocolo de Descrição de Sessão, SDP) que além da tarefa citada anteriormente também é responsável por estabelecer parâmetros da conexão multimídia.

Os dois principais protocolos da camada de transporte segundo (TANNENBAUM; WETHERALL, 2010) são o *Transmission Control Protocol*(TCP) e o *User Datagram Protocol* (UDP) sendo este último o mais utilizado em telefonia IP.

O protocolo UDP permite o envio de *datagrama* sem nenhum tipo de garantia, ou seja, ele é um protocolo não confiável, apesar disso ele é um protocolo adequado para aplicações que necessitam de fluxo de dados em tempo real como a VoIP. Já o protocolo TCP é confiável, ou seja, permiti a entrega de pacotes sem erro porém, o mesmo é um protocolo mais lento que o UDP.

Uma das principais preocupações no serviço VoIP é o custo associado ao meio de transmissão. Devido a este fato, em uma comunicação VoIP são utilizadas técnicas de compressão, que não causam perdas significativas na qualidade do sinal recebido (RODRÍGUEZ et al., 2019).

Os codecs de fala são responsáveis por essa compressão. Existem diferentes codecs de fala, um

dos mais adotados nas redes de comunicação atuais é o codec *Adaptive Multi-Rate Wideband* (AMR-WB) (SALAMI et al., 2002).

O AMR-WB é um codec de voz usado para comunicações de dispositivos móveis. É amplamente utilizado por operadoras de rede para fornecer conversas de alta qualidade. AMR-WB é baseado na predição linear gerada pelo código algébrico ACELP que usa uma técnica de quantização vetorial (KUMARI et al., 2017). O AMR-WB usa nove modos de operação, de 6,6 kbps a 23,85 kbps, e cada um tem uma resposta diferente para perdas de pacotes.

2.4 Qualidade de experiência

Atualmente o uso de aplicações avançadas como *streaming* de áudio e vídeo tem se tornado o principal tráfego da Internet. O uso por exemplo da webconferência e da tecnologia VoIP para comunicação dentro das empresas aumenta constantemente. Segundo (CETIC.BR, 2017) em 2017 no Brasil 33% das empresas que possuem internet utilizam a mesma para VoIP e ou vídeoconferência. Estas aplicações avançadas necessitam de uma rede robusta e de alta capacidade, tais redes demandam equipamentos com alta capacidade de processamento. A fim de garantir a Qualidade de Experiência (QoE) do usuário no uso destas aplicações os provedores de serviço tem investido cada vez mais em hardware melhores e em enlaces com maior capacidade física.

Segundo Setor de Normalização em Telecomunicações da União Internacional das Telecomunicações (ITU-T) a QoE é definida como o grau de satisfação ou insatisfação de uma aplicação ou serviço percebido pelo usuário final. Ela é expressa em sentimentos humanos e é influenciada por fatores humanos como gênero, idade, humor entre outros mas, também é influenciada por fatores dos sistemas.

2.4.1 Qualidade de Serviço (QoS)

Definida pela ITU-T E.800 (RECOMMENDATION, 2008) a Qualidade de Serviço (QoS) pode ser definida como as características necessárias ao serviço de comunicação de forma a satisfazer as demandas necessárias ao usuário do serviço (MÖLLER; RAAKE, 2014). Tais características são os parâmetros de rede como o *throughput*, a perda de pacotes, o *delay* entre outros.

Portanto de forma geral a QoS está diretamente ligada a parâmetros de rede da mesma forma que a QoE porém, segundo (MÖLLER; RAAKE, 2014) a QoE além de assegurar que os requisitos técnicos sejam satisfeito, ela se baseia na adoção do ponto de vista do usuário para julgar se as reais necessidades e expectativas são satisfeitas.

2.4.2 QoE em chamadas VoIP

A QoE no uso da telefonia VoIP pode ser expressa através da Pontuação Média de Opinião (*Mean Opinion Score*, MOS) da ITU-T P.800 (ITU-T Rec. P.800, 1996) que é uma indicação numérica da qualidade percebida. O MOS pode ser quantificado de forma direta ou indireta, a forma direta é a captação de dados através da aplicação de questionário especificado pela ITU-T P.800 (ITU-T Rec. P.800, 1996) nos usuários finais que testam as ligações e definem notas que variam de 1 a 5 conforme a Tabela 2.2.

Tabela 2.2 – Pontuação do MOS

Mos	**Qualidade**	**Descrição**
5	Excelente	Um sinal de voz perfeito capturado em local silencioso
4	Bom	Chamada telefônica de longa distância (rede PSTN)
3	Razoável	Necessário algum esforço para ouvir
2	Pobre	Baixa qualidade na fala, difícil de ouvir
1	Ruim	Fala não clara

Fonte: Adaptado da recomendação ITU-T P.800 (ITU-T Rec. P.800, 1996)

A forma indireta é através da avaliação de parâmetros, especificamente em aplicações que fazem uso da redes de computadores a QoE é influenciada por parâmetros da rede como a largura de banda, o *delay* e o *jitter* que por sua vez dependem de vários fatores como: capacidade física do enlace, capacidade de processamento dos roteadores, os algoritmos de roteamento entre outros.

Existem técnicas que possibilitam calcular o valor do QoE e uma visão geral de algumas dela é abordado em (KUIPERS et al., 2010). Uma técnica proposta por (SHAIKH; FIEDLER; COLLANGE, 2010) relaciona a QoE e a largura de banda através do MOS. Descrita pela Equação (2.12) essa relação determina o valor do MOS através da largura de banda (B).

$$MOS = 1,15 + 1,50 * ln(B) \tag{2.12}$$

Em (KIM et al., 2008) foi proposto um relação da QoS com QoE que possibilita calcular a QoE através de alguns parâmetros de rede dentre eles o *delay*, jitter, perda de pacotes e a largura de banda. Uma métrica de avaliação que é determinada pela taxa de codificação e pelas características da rede como a perda de pacotes é proposto em (GARCIA; RAAKE, 2010). Uma outra abordagem é o E-Model, definido pela ITU-T G.107 (ITU, 2003) foi implantado com finalidade de calcular o MOS em função de métricas de rede como a taxa de perdas de pacote e o atraso da rede, sendo necessário avaliar somente o lado receptor da conversa, ele será melhor abordado na próxima subseção.

2.4.3 Wide Band E-Model

O algoritmo WB E-Model (ITU-T Rec. P.862.1, 2003) é um método paramétrico que prevê a qualidade da conversa usando diferentes fatores de comprometimento relacionados ao ambiente acústico, rede e codec de voz. O R_{WB} é a classificação de qualidade global que é obtida usando todos os fatores de imparidade. Esse valor é expresso em uma escala de qualidade de 0 a 129. Quanto maior o valor melhor a qualidade. A pontuação R_{WB} é determinada por:

$$R_{\mathrm{WB}} = R_{0,\mathrm{WB}} - I_{\mathrm{s,WB}} - I_{\mathrm{d,WB}} - I_{\mathrm{e\text{-}eff,WB}} + A \tag{2.13}$$

onde $R_{0,\,\mathrm{WB}}$ representa a relação sinal-ruído básica (SNR), e para redes WB o valor padronizado é 129; $I_{\mathrm{s,\,WB}}$ representa a combinação de todas as degradações que ocorrem simultaneamente com o sinal de voz, para sinais WB o valor adotado deste fator é 0; $I_{\mathrm{d,\,WB}}$ representa as degradações causadas pelo atraso; $I_{\mathrm{e\text{-}eff,\,WB}}$ é a degradação da qualidade devido ao equipamento; A representa um fator de vantagem mas, no WB E-model não é considerado, portanto seu valor é 0. Para objeto de estudo deste trabalho o foco deve ser nos valores de $I_{\mathrm{e\text{-}eff,\,WB}}$ e $I_{\mathrm{d,\,WB}}$.

$I_{\mathrm{e\text{-}eff}}$ é determinado da seguinte maneira:

$$I_{\mathrm{e,eff,WB}} = I_{\mathrm{e,WB}} + (95 - I_{e,WB}).\frac{Ppl}{Ppl + Bpl_{\mathrm{WB}}} \tag{2.14}$$

onde, $I_{\mathrm{e,WB}}$ é o fator de comprometimento do equipamento quando a perda pacote é zero, relacionado apenas ao comprometimento do codec; Ppl é a probabildiade de perda de pacote; e Bpl_{WB} é o fator de robustez de perda de pacote para um codec específico em redes WB.

No anexo IV da recomendação G.113 do ITU-T (ITU-T Rec. G.113 - Appendix IV, 2009), os valores de Ie e Bpl para AMR-WB codec são definidos. A Tabela 2.3 apresenta a taxa de transferência de cada modo de operação AMR-WB e seus valores I_e e Bpl existentes. Observe que alguns valores de Bpl não são definidos (ND) em alguns casos.

Tabela 2.3 – Modos de Operação AMR-WB e seus Respectivos Valores de I_e e Bpl

Modos de Operação AMR-WB	Taxa (kbps)	Ie	Bpl
0	6,60	41	ND
1	8,85	26	ND
2	12,65	13	4.3
3	14,25	10	ND
4	15,85	7	ND
5	18,25	5	ND
6	19,85	3	ND
7	23,05	1	ND
8	23,85	8	4.9

Fonte: Do autor (2021)

Neste trabalho, será considerado os modos de operação 2 e 8, pois possuem o parâmetro Bpl já definido, conforme pode ser observado na Tabela 2.3. Assim, o valor $I_{\text{e,eff,WB}}$ pode ser calculado.

O $I_{\text{d.WB}}$ é calculado usando a seguinte relação:

$$I_{\text{d, WB}} = I_{\text{dte, WB}} + I_{\text{dle, WB}} + I_{\text{dd, WB}} \tag{2.15}$$

onde $I_{\text{dte, WB}}$ é uma estimativa para as degradações devido ao eco do locutor, $I_{\text{dle, WB}}$ representa as deficiências devido ao eco do ouvinte, e $I_{\text{dd, WB}}$ representa o prejuízo causado por um atraso absoluto T_a na rede. Neste estudo, $I_{\text{dte, WB}}$ e $I_{\text{dle, WB}}$ não são considerados porque estão relacionados a problemas de eco e acústicos nas extremidades da comunicação que está fora de âmbito desta pesquisa.

O $I_{\text{dd, WB}}$ é definido por:

$$I_{\text{dd,WB}} = \begin{cases} 1, & T_a < 100ms \\ 25[(1+X^6)^{(\frac{1}{6})} - 3(1+(\frac{X}{3})^6)^{\frac{1}{6}} + 2], & T_a > 100ms \end{cases} \tag{2.16}$$

onde

$$X = \frac{log(\frac{T_a}{100})}{log2} \tag{2.17}$$

Os valores das variáveis Ppl e T_a são parâmetros de rede, portanto, o R_{WB} pode ser calculado usando (2.13).

3 TRABALHOS RELACIONADOS

Nos últimos anos foram propostos alguns trabalhos que aplicam *Machine Learning* em protocolos de roteamento. Alguns apresentam apenas um modelo genérico como em (PESHKIN; SAVOVA, 2002), onde o autor criar um modelo genérico baseado em RL para redes ad-hoc com foco em estratégias de roteamento. Outro trabalhos apresentam um solução real, porém com uma abordagem específica como em (WU et al., 2020) onde o autor aborda a aprendizagem por reforço em roteamento em redes de veículos *Vehicular Ad-hoc Networks (VANETs)*, nesse trabalho são considerados fatores intrínseco a VANETs para o desenvolvimento da solução. Outros trabalhos focam em redes de sensores sem fio e suas características como (WANG; SHIN, 2019) ou robótica não tripulada (JUNG; YIM; KO, 2017).

Em (TANG et al., 2018) propõe-se um controle de tráfego inteligente por meio de aprendizagem profunda e não RL, especificamente as redes neurais convolucionais como método de controle de tráfego inteligente. O algoritmo proposto se divide basicamente em duas partes. Na primeira parte denominada Inicial o algoritmo calcula todos os possíveis caminhos para cada nó da rede. As informações sobre os caminhos e sobre as métricas (número de saltos, distância) são gravadas em uma matriz bidimensional. A segunda parte denominada Execução é divida em 3 etapas descritas a seguir:

1. A primeira etapa é chamada de partida a frio, onde o algoritmo roda sem o auxilio da RNA e opta pelo caminho com número menor de saltos para enviar o primeiro pacote. Após a primeira execução uma coleta de dados é feita para alimentar a rede neural pela primeira vez.

2. Na segunda etapa denominada de fase de atualização o algoritmo coleta os dados sobre a atual situação da rede, essa coleta ocorre em um intervalo de tempo h sendo h>1. De acordo com análise sobre o tráfego de pacotes um limite superior é definido e todo caminho que estiver acima do limite será considerado congestionado. Posteriormente a coleta é feito o julgamento do caminho através da rede neural, que considera o caminho válido no caso um bom caminho ou inválido que é o caso de um caminho congestionado.

3. A terceira etapa é responsável pela alimentação e treinamento da rede neural enquanto o algoritmo é executado, esta etapa é responsável pelo ajuste dos pesos da RNA. As etapas 2 e 3 ocorrem de forma periódica e continua enquanto a rede estiver funcionando.

Nos testes o algoritmo foi comparado com o protocolo OSPF padrão e apresentou uma redução na taxa de perda de pacotes.

No trabalho (DING et al., 2019) o autor usa *Deep Reinforcement Learning (DRL)* para desenvolver um novo protocolo de propósito geral. A DRL implementa arquiteturas de aprendizado profundo

(redes neurais profundas) com algoritmos de aprendizado de reforço. No trabalho existe um controlador centralizado para gerenciar os roteadores no envio de pacotes, baseando no aprendizado de reforço de um agente único, que no caso é o controlador. Ele é responsável por escolher as tarefas em uma lista e aplicar elas.

O algoritmo proposto procurar evitar caminhos congestionados e demasiadamente longos, para tal a recompensa está ligada ao sucesso na entrega do pacote e ao tamanho do caminho. A recompensa é definida da seguinte maneira:

1. r_c se a rede está congestionada
2. r_e se a ação é inválida
3. 0 se o pacote chegar no destino
4. -1 demais casos

onde, r_c é definido toda vez que o caminho escolhido está congestionado, ou seja, quando é feito a escolha de uma ação ruim, é uma constante com valor negativo inferior a -1. r_e também é uma constante de valor negativo inferior a -1 e é definida como recompensa sempre que o controlador escolhe uma ação inválida, que não pode ser executada. O valor 0 é definido toda que vez que o pacote chega ao nó destino. E por último o valor -1 é definido quando um pacote trafega com sucesso entre dois nós da rede, exceto entre qualquer nó e o destino. A ideia é que quanto mais saltos tiverem entre o nó destino e o fonte mais vezes terá a punição -1, o que diminuirá a estimativa. Sendo assim rotas maiores tem estimativas menores.

Os resultados obtidos são satisfatórios em comparação também ao OSPF. Porém, a dependência de um controlador central tem desvantagens como: um único ponto de falha.

Analisando os trabalhos citados anteriormente foram encontradas algumas lacunas que potencializam novos trabalhos na área. Primeiramente foi possível constatar que nos trabalhos não existe uma solução de uso geral que utiliza RL, foca em redes ad-hoc e que faça comparação com alguns dos protocolos atuais para redes ad-hoc.

Outro ponto a se destacar, é que nos trabalho descritos anteriormente para avaliar o desempenho foi utilizado simulações de um cenário de rede sem mobilidade dos nós, ou seja, todos os nós ficam fixos e ativos durante toda a simulação. Porém, em um cenário real muitas vezes ocorre que os nós da rede desligam, param de se comunicar por algum problema ou se deslocam de posição no caso de rede sem fio. Portanto, uma outra lacuna encontrada é explorar cenários onde os nós da rede não estejam fixos.

O ajuste na frequência de envio de mensagens *Hello* pode reduzir o *overhead* da rede e consequentemente melhorar o desempenho da mesma.

Em (MAHMUD; CHO, 2019), os autores propõem um ajuste no intervalo de envio de mensagens *Hello* do protocolo AODV para *Flying Ad-hoc Networks* (FANETs), com foco na redução do consumo de energia de veículos aéreos não tripulados (*Unmanned Aerial Vehicles*, UAVs). O denominado *Energy Efficient Hello* (EE-Hello) envia mensagens de *Hello* depois que os UAVs percorrem uma distância específica. Essa distância é definida com base no alcance de transmissão da UAVs, na velocidade de movimentação, no número de UAVs da rede e no espaço aéreo permitido. O valor do parâmetro *Hello Interval* (T_{H}) que define o tempo de envio de mensagens *Hello* é calculado da seguinte maneira:

$$T_H = \gamma \times \frac{1}{v_{avg(n)}} \tag{3.1}$$

onde γ é a distância que o UAV percorrer para então enviar uma mensagem *Hello* e $v_{\mathrm{avg(n)}}$ é a velocidade média atual do UAV.

Já o valor de γ é calculado através da taxa de transferência atual da UAV e do parâmetro α que é definido pela missão atual da UAV.

Nos protocolos de roteamento se define um tempo de envio de mensagens *Hello* e um tempo de espera em que se estima receber uma mensagem *Hello* dos vizinhos, caso esse tempo seja excedido o vizinho é então removido da lista de vizinhos. O algoritmo ajusta o valor de T_{H} sempre que se excede esse tempo de espera ou quando se recebe um *Hello* e a taxa atual de densidade de UAVs é alta. Apesar de apresentar bons resultados como redução de overhead em relação ao AODV padrão e principalmente proporcionar uma redução no consumo de energia em até 25% o trabalho considera características das FANETs para o ajuste do *Hello Interval*.

Em (GIRUKA; SINGHAL, 2005) os autores propõem três algoritmos para ajustar o tempo de envio de mensagens *Hello*. O primeiro algoritmo é chamado de *Hello* Reativo, em que as mensagens de *Hello* são enviadas apenas quando o nó deseja enviar algum pacote. Dessa forma, quando um dado nó da rede desejar enviar um pacote ele primeiro envia uma mensagem *Hello* e aguarda por um tempo determinando pela constante RESPWAITTIME, se não receber resposta dos vizinhos ela repete o processo mais duas vezes. Ao receber resposta dos seus vizinhos o nó então atualiza as informações sobre a sua vizinhança. A proposta reduz o *overhead* pois o número de mensagens *Hello* é reduzido. Porém, as informações sobre o estado da vizinhança ficam cada vez mais defasadas a medida que o nó demora para enviar algum pacote e isto pode afetar negativamente o desempenho da rede.

O segundo método é denominado *Hello* Baseado em Eventos, nele o ajuste é feito com base nos eventos que ocorrem na rede. O nó da rede a princípio envia mensagens *Hello* com uma frequência

padrão, mas se após um período de tempo que é predefinido esse nó não receber mensagens *Hello* ou não precisar enviar pacotes ele então para de enviar mensagens *Hello*. Se receber algum *Hello* posteriormente o nó processa a informação recebida e então recomeça a enviar mensagens *Hello*. Esse hiato no envio de mensagens *Hello* proporciona redução do *overhead* na rede. Mas, por outro lado as informações podem ficar muito defasadas em determinadas situações. Por exemplo, se todos os nós da rede se afastarem e somente voltarem a se aproximar após o período de tempo predefinido em que o algoritmo suspende o envio de mensagens *Hello* nenhum nó enviará mensagens *Hello*. Dessa forma, as informações de topologia de todos os nós da rede estariam desatualizada até que um nó decida enviar um pacote, o que poderia acarretar na degradação de desempenho da rede.

No terceiro método chamado *Hello* Adaptativo cada nó na rede envia um *Hello* após se mover por uma distância definida. A frequência do envio de mensagens *Hello* é então ajustada de acordo com a velocidade em que o nós se movimenta. Para evitar que o valor cresça de forma indefinida foram definidos limites para frequência do envio de mensagens *Hello*. Também é definido uma frequência mínima de envio de mensagens quando o nó não se mover. Nessa solução quando os nós estiverem parados o *overhead* da rede será reduzido porém, a solução não considera situações em que o nós não se movem mas se desligam ou apresentam falhas. Além disso, é necessário que o nó tenha conhecimento de sua posição.

Em (THAKKAR; SHETE, 2015) foi proposto o *Adaptive Hello messaging based AODV* (AH AODV) que faz ajuste dinâmico na frequência envio de mensagens *Hello* do protocolo AODV. Neste trabalho a frequência é ajustada de acordo com os eventos que acontecem na rede. São considerados eventos o envio de qualquer mensagem de controle e o envio de pacotes entre os nós da rede. Existe um buffer que armazena os eventos ocorridos, o tempo de duração de cada evento é registrado. A cada 6 segundos o nó calcula o *Event Interval*(EI) que é a diferença entre o tempo de evento armazenado no *buffer* e o o tempo de evento obtido. O *Hello Interval* é então ajustado da seguinte maneira:

Tabela 3.1 – Ajuste do Hello Interval no Algoritmo AH AODV

Valor do EI(segundos)	**Valor do Hello Interval(segundos)**
EI < 1	Hello Interval = 1
1 < EI <= 3	Hello Interval = 2
EI > 3	Hello Interval = 3

Fonte: Do autor (2021)

Dessa forma, o *Hello Interval* é ajustado de acordo com os eventos ocorridos na rede. O AH AODV obteve bons resultados se comparado ao AODV padrão. Porém, não analisar a mobilidade da rede pode definir uma baixa frequência de envio de mensages *Hello* quando se tem uma alta mobilidade gerando assim um degradação de desempenho da mesma. Além disto, o trabalho não descreve o

tamanho desse *buffer* que pode impactar negativamente em redes onde os nós tem pouca capacidade de armazenamento como por exemplo em redes de sensores sem fio.

4 METODOLOGIA

Neste capítulo é apresentada a metodologia deste trabalho. A descrição dos cenários e parâmetros de emulação utilizados para avaliar o e-RLRP com os demais protocolos.

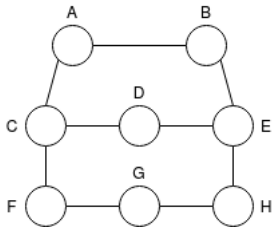
4.1 Avaliação do Algoritmo Proposto

Nesta seção, primeiramente, são descritas as quatro topologias de rede utilizadas nas simulações. Posteriormente, dois cenários de simulação são explicados. Finalmente, as taxas de transmissão nos cenários são explicadas e o ambiente de simulação é descrito.

4.1.1 Topologia de Rede

Neste trabalho, quatro topologias de rede foram criadas para simular uma rede de nó sem fio, que são denominadas T1, T2, T3 e T4. Os nomes dos nós foram distribuídos de forma a melhorar a compreensão dos cenários que serão descritos posteriormente. Na Topologia T1, existem 8 nós dispostos e interligados conforme ilustrado na Figura 4.1.

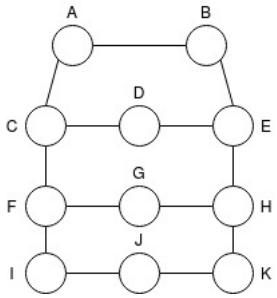
Figura 4.1 – Topologia T1



Fonte: Do autor (2021)

A topologia T2 é uma extensão da T1 com a adição de 3 nós, totalizando assim 11 nós conforme ilustrado na Figura 4.2.

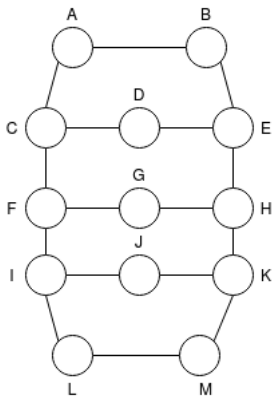
Figura 4.2 – Topologia T2



Fonte: Do autor (2021)

A T3 está ilustrada na Figura 4.3. Ela também é uma extensão da T1, mas como 5 nós extras, totalizando assim 13 nós.

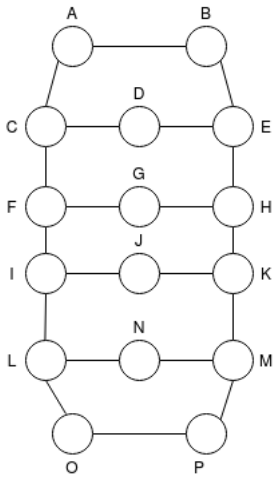
Figura 4.3 – Topologia T3



Fonte: Do autor (2021)

Por último a T4 está ilustrada na Figura 4.4. Assim como as demais também é um extensão da T1 com 8 nós extras totalizando um rede com 16 nós.

Figura 4.4 – Topologia T4



Fonte: Do autor (2021)

4.1.2 Cenário de Emulação

Para testar as funcionalidades do e-RLRP, foram desenvolvidos dois cenários distintos em que existem rotas que degradam o desempenho da rede. Para tanto, alguns nós da rede foram configurados para se desconectarem de forma recorrente em instantes aleatórios, simulando falhas de nós e mobilidade na rede.

No primeiro cenário, apenas a topologia T1 é usada. Um fluxo é definido com o nó C sendo a origem e E sendo o destino, o nó D será programado para desligar 5 vezes simulando assim mobilidade na rede. Este nó faz parte da rota mais curta entre a origem e o destino do tráfego para T1. Para uma melhor associação posterior, o primeiro cenário é denominado Programado (P). Assim, o cenário P é uma prova de conceito para testar o RL no e-RLRP, no qual se espera um desempenho melhor que o de outro protocolo.

Em princípio, a rota com o menor número de saltos é o melhor caminho e é aquela que deve ser escolhida inicialmente por todos os protocolos. Porém, neste cenário a escolha deste caminho causará degradação na rede, visto que existe um nó que se desconecta recorrentemente causando perda de pacotes. Como o e-RLRP pode aprender com a rede, ele deve ser capaz de evitar o caminho que contém nós que se desconectam de forma recorrentes.

No segundo cenário denominado *Random* (R) também um fluxo é definido com o nó C sendo a origem e E sendo o destino. Os nós A, D e G da topologia T1; nós A, D, G e J de T2; nós A, D, G, J e L de T3; e os nós A, D, G, J, N e O de T4 são desabilitados aleatoriamente em diferentes instantes, a fim de simular desconexões aleatórias. Além disso, são definidas 3 configurações para as desconexões. Na primeira configuração 3 eventos de desconexão são sorteados entre os nós mencionados acima para cada topologia. Na segunda configuração 5 e na terceira configuração 7 eventos de desconexão. O motivo da escolha apenas desses nós é garantir que cada rota tenha apenas um nó que falhe, garantindo assim a mesma probabilidade de sortear uma desconexão para cada rota. Esses nós são desconectados aleatoriamente em cada simulação. Os instantes em que cada nó cai durante a simulação são definidos aleatoriamente, então, o algoritmo de roteamento não sabe qual nó está inativo para evitar aquele caminho. Essas características são diferentes dos cenários de rede em que as desconexões dos nós são controladas e um escalonador de rede pode ser implementado. O objetivo deste cenário é testar o e-RLRP em um cenário aleatório quando a degradação da rede aumenta.

Além disso, duas configurações diferentes do cenário R são definidas para as topologias T3 e T4 onde o número do fluxo é maior que um. Uma configuração de rede com 3 fluxos é definida, nela o primeiro fluxo é do nó C para o E, o segundo do nó F para o B e o terceiro do nó I para o H. Na segunda configuração da rede são 4 fluxos, onde um fluxo adicional de o nó M para o K é adicionado aos três fluxos mencionados anteriormente. O objetivo desses dois cenários é investigar o impacto do *overhead* adicional da rede devido às mensagens de controle RL.

Em ambos os cenários, a capacidade do e-RLRP de evitar rotas que degradam a rede por meio do uso de RL é testada. E principalmente também é avaliada a capacidade do e-RLRP de reduzir o *overhead*

da rede em cenários de mobilidade. Proporcionando assim um maior *throughput* e reduzindo a taxa de perda de pacotes.

A Tabela 4.1 mostra quais nós foram configurados para desligar, simulando assim uma desconexão nas topologias T1, T2, T3 e T4 para todos os cenários. No Cenário Programado usa-se apenas o T1 porque é um cenário para prova de conceito.

Tabela 4.1 – Conjunto de Nós Definidos para se Desligarem

	T1	**T2**	**T3**	**T4**
Cenário P	D	-	-	-
Cenário R	ADG	ADGJ	ADGJL	ADGJN

Fonte: Do autor (2021)

4.1.3 Taxa de transmissão do Codec AMR-WB

Este trabalho também visa testar o impacto dos protocolos de roteamento mencionados anteriormente em um serviço de comunicação real, para tal, utiliza-se como estudo de caso o cenário de comunicação VoIP. Assim, um tráfego do nó C para E é simulado com diferentes taxas de bits definidas de acordo com o codec AMR-WB.

O sinal de voz transmitido em uma rede IP é compactado por um codec de voz e para ser transmitido na rede a carga útil é empacotada. São inseridos o *Real Time Protocol* (RTP), os cabeçalhos UDP e IP. As taxas de bits apresentadas na Tabela 2.3 referem-se apenas à carga útil, então, é necessário adicionar o número de bits referente ao RTP (12 bytes), UDP (8 bytes) e IP (20 bytes) para obter a taxa de transmissão. Por exemplo, AMR-WB-Modo 2 (12,65 kbps) contém 253 bits que são enviados a cada 20 ms, então, se os 320 bits de cabeçalhos são adicionados, um total de 573 bits são enviados neste mesmo período de tempo, o que representa um taxa de transmissão de 28,65 kbps.

A Tabela 4.2 mostra as taxas de transmissão usadas nos cenários de teste.

Tabela 4.2 – Bit-rate após Adição dos Cabeçalhos RTP, UDP e IP

AMR-WB Bit-rate (kpbs)	**Bit-rate considerando os cabeçalhos RTP/UDP/IP (kpbs)**
12.65	28.65
23.85	39.85

Fonte: Do autor (2021)

Fonte: Do autor (2021)

4.1.4 Ambiente de Emulação

Para testar e analisar o desempenho dos quatro protocolos mencionados anteriormente, utilizamos o emulador de rede *Common Open Research Emulator* (CORE) (AHRENHOLZ et al., 2008). Desenvolvido pela divisão de Pesquisa e Tecnologia da Boeing, CORE é um emulador de código aberto

e em tempo real. O CORE foi escolhido porque permite o uso de protocolos e aplicativos de roteamento do mundo real usando a virtualização do sistema Linux. O código e-RLRP deve ser executado em uma plataforma Linux. Cada nó no emulador é uma máquina virtual com interface de rede e recursos compartilhados com a máquina host. Os protocolos de roteamento e-RLRP, RLRP, BATMAN e OLSR são instalados para serem usados pelos nós da rede.

As métricas de desempenho da rede obtidas nos testes foram *throughput*, Probabilidade de Perda de Pacotes (*Probability of Packet Loss*, Ppl), *Round Trip Time* (RTT) e o *overhead* gerado por mensagens de controle. Os valores de *throughput* e Ppl são calculados usando a ferramenta Iperf (TIRUMALA et al., 2005), que é capaz de gerar fluxos de tráfego UDP com taxas definidas. Para calcular o RTT, o fluxo UDP é substituído por um fluxo ICMP gerado pelo comando PING nativo do Linux. O próprio comando PING retorna o valor RTT. O *overhead* é medido usando a ferramenta WireShark (LAMPING; WARNICKE, 2004). Para o calculo do *overhead* as mensagens de controle são contabilizadas e todo tráfego da rede também. Por fim, o valor de *overhead* é confirmado pelo resultado da quantidade total de bytes trafegados na rede menos a quantidade de bytes trafegados pelos fluxos UDP definidos.

Além das ferramentas mencionadas, o shell script nativo do Linux é usado para desligar nós de forma programada ou aleatória.

Finalmente, a qualidade da fala de uma comunicação VoIP é avaliada. Para tanto, os parâmetros da rede como Ppl e *delay* são utilizados como entradas do algoritmo E-model para estimar a qualidade da comunicação.

5 O ALGORITMO PROPOSTO

Neste capítulo é apresentado o algoritmo desenvolvido que propõe um mecanismo de ajuste dinâmico no frequência de envio de mensagens *Hello* para um protocolo de roteamento baseado em RL.

Para enviar um pacote, o nó precisa saber quais nós vizinhos estão conectados diretamente. Portanto, um procedimento de descoberta de vizinhança é necessário. No protocolo RLRP (DUGAEV et al., 2018) descrito na subseção 2.2.3, esse procedimento ocorre por meio da transmissão de mensagens denominadas *Hello*.

Por padrão, as mensagens *Hello* são enviadas a cada 2 segundos, portanto, as informações sobre os vizinhos são atualizadas no mesmo período de tempo. Esse parâmetro de intervalo de atualização é denominado *Broadcast Interval* (BI). O RLRP tem 10 tipos de cabeçalhos, destacamos dois deles que são: o cabeçalho de recompensa denominado *Reward Header* e o *Hello Header*. A estrutura dos campos de dados do *Reward Header* e do *Hello Header* são descritos na Tabela 5.1 e na Tabela 5.2, respectivamente.

Tabela 5.1 – Formato do Cabeçalho de Recompensa

Campo	Tamanho (bits)	Descrição
TYPE	4	Identificador do Cabeçalho
ID	20	Identificador do serviço de Mensagem
NEG_REWARD_FLAG	1	Flag de Recompensa
REWARD_VALUE	7	Valor de Recompensa
MSG_HASH	32	Identificador do Pacote

Fonte: Adaptado de (DUGAEV et al., 2018)

O campo TYPE descreve qual é o cabeçalho, o campo ID é o identificador exclusivo do serviço de mensagem. O campo Neg Reward Flag é um sinalizador de teste que verifica se a recompensa é negativa ou positiva, o Reward Value é o valor da recompensa e por fim, o Msg Hash é o identificador do pacote enviado a qual a recompensa pertence. O cabeçalho de Recompensa possui tamanho de 8 bytes.

Tabela 5.2 – Formato do Cabeçalho Hello

Field Name	Size (bits)	Description
TYPE	4	Identificador do Cabeçalho
IPV4_COUNT	1	Número de endereços IPv4
IPV6_COUNT	2	Número de endereços IPv6
TX_COUNT	24	Número de retransmissões
GW_MODE	1	Configuração de Modo GateWay
IPV4_ADDRESS	32	Endereço IPv4 (se existir)
IPV6_ADDRESS_1	128	Endereço IPv6 (se existir)
IPV6_ADDRESS_2	128	Endereço IPv6 (se existir
IPV6_ADDRESS_3	128	Endereço IPv6 (se existir)

Fonte: Adaptado de (DUGAEV et al., 2018)

O tamanho do *Hello Header* varia de 4 a 56 bytes, esta variação depende do endereço do nó que pode ser IPV4 ou IPV6. O campo Type define qual é o cabeçalho, o campo IPv4 Count define o número de endereços IPv4 atribuídos, limitado a um. O campo IPv6 Count é o número de endereços IPv6 atribuídos, limitado a três. Tx Count é o número de retransmissões da mensagem, o GW Modo define que um nó é um Gateway na rede, os campos IPv4 e IPv6 Address definem o endereço do nó.

5.1 e-RLRP

Como pode ser observado na Tabela 5.1, o cabeçalho de Recompensa utilizado no RLRP tem 8 bytes de comprimento e gera um *overhead* adicional, que corresponde ao uso da técnica RL. Neste contexto, este trabalho implementou um algoritmo para reduzir o *overhead* gerado pela mensagem *Hello*, especificamente para reduzir a frequência de envio de mensagens *Hello* de forma a compensar o *overhead* adicional gerado pelo cabeçalho de Recompensa.

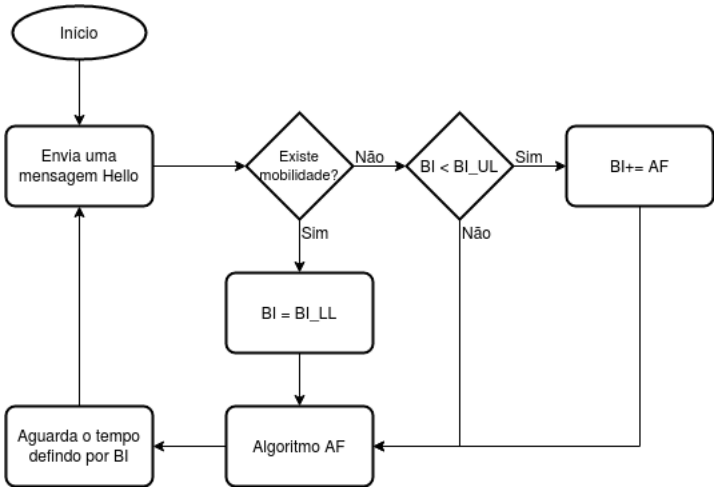
É claro que ao aumentar o intervalo de tempo para envio de mensagens *Hello*, definido pelo parâmetro BI, diminuirá a frequência de envio de mensagens, e consequentemente, o *overhead* também será reduzido. No entanto, um valor alto também pode atrasar o tempo de atualização das informações sobre a vizinhança e o roteamento pode ser prejudicado devido a informações desatualizadas.

Assim, o algoritmo proposto implementado no e-RLRP é capaz de ajustar dinamicamente a frequência de envio de mensagens *Hello*. Este ajuste no parâmetro BI é feito de acordo com a mobilidade presente na rede. Se a rede for estática, ou seja, nenhum vizinho entrar ou sair, não é necessário enviar mensagens *Hello* com alta frequência. Caso contrário, se a rede apresentar alta mobilidade, é necessário enviar mensagens com maior frequência para que as informações sobre as alterações seja disseminada rapidamente.

Uma representação geral do algoritmo proposto é apresentada na Figura 5.1.

O envio de mensagens *Hello* começa junto com o daemon do e-RLRP, em seguida o algoritmo verifica a mobilidade da rede. Para tanto, o e-RLRP utiliza duas funções já existentes no RLRP, uma com nome de *Update Neighbors File* responsável por atualizar a lista de vizinhos toda vez que um *Hello* de um novo nó for recebido. E outra função denominada *Check Expired Neighbors* que verifica se uma mensagem *Hello* foi recebida de um vizinhos já conhecido a cada 7 segundos. Se um vizinho está 7 segundos ou mais sem enviar um *Hello*, ele é removido da lista porque provavelmente está fora de alcance . Este intervalo de tempo foi definido experimentalmente em (DUGAEV et al., 2018). Caso seja detectado um novo vizinho ou perdido um existente, será considerado pelo e-RLRP que existe mobilidade na rede.

Figura 5.1 – Mecanismo de Ajuste Dinâmico do Algoritmo Proposto



Fonte: Do autor (2021)

No e-RLRP proposto, quando ocorrer mobilidade, o valor do BI será reduzido a um limite inferior denominado BI *Lower Limit* (BI_{LL}), após isto o algoritmo aguarda o intervalo de tempo definido pelo novo valor de BI, envia uma mensagem *Hello* e reinicia o processo. Caso não ocorra um evento de mobilidade, o parâmetro BI será incrementado com o parâmetro *Adjustment Factor* (AF). Esse incremento é feito respeitando o limite superior denominado BI *Upper Limit* (BI_{UL}). Da mesma forma após o incremento o novo tempo definido por BI será aguardado, após isto uma mensagem *Hello* será enviada e o processo será reiniciado. Dessa forma a frequência de envio de mensagens *Hello* é ajustada de acordo com a mobilidade da rede.

5.1.1 Definição dos Limites Superior e Inferior para o parâmetro BI

Quanto mais alto o valor do parâmetro BI menor é a frequência de envio de mensagens *Hello* e consequentemente menor é o *overhead*. Porém, é necessário definir um limite para que o valor não cresça indefinidamente.

O BI_{UL} não pode ser maior ou igual a 7 segundos devido à função *Check Expired Neighbors*. Caso contrário, os nós da rede serão eliminados quando o *timeout* da função for atingido. Considerando que o valor BI_{UL} deve ser menor que 7 segundos e também considerando uma margem devido a latência da rede, o valor 6 segundos é definido para garantir que os vizinhos não sejam erroneamente removidos.

Para definir BI_{LL} três valores de BI inferiores ao padrão 2 segundos são testados no cenário denominado Programado que está descrito na subseção 4.1.2. O *overhead* é calculado considerando os nós de origem e destino. Os valores de BI 0,5, 1,0 e 1,5 são testados. O valor o padrão também é testado

no mesmo cenário e o *overhead* obtido para ele foi de 1,42 MB. A Tabela 5.3 mostra os resultados de *overhead* para cada valor de BI.

Tabela 5.3 – Overhead Obtido para os Valores de BI

Valores de BI (s)	Overhead (MB)	Incremento do Overhead (%)
1.5	1.45	2.12% de ganho em relação ao BI 2,0 s
1.0	1.48	4.22% de ganho em relação ao BI 2,0 s
0.5	1.60	12.67% de ganho em relação ao BI 2,0 s

Fonte: Do autor (2021)

A Tabela 5.3 mostra o aumento de *overhead* dos valores testados em relação ao valor padrão utilizado no RLRP. O valor de BI de 1,5 teve um ganho de 2,12%, o valor de 1,0 apresentou um aumento de 4,22%. O valor de 0,5 obteve o maior aumento, um ganho de aproximadamente 12,67%, considerando que o valor 0,5 teve um grande aumento no *overhead* em relação aos anteriores, o valor de 0,5 é então descartado. Portanto, optou-se pelo valor testado intermediário, BI_{LL} então é definido como 1,0.

5.1.2 Fator de Ajuste

Para o incremento de BI é utilizado como explicado anteriormente um fator de ajuste denominado *Adjustment Factor* (AF). O objetivo do e-RLRP é reduzir o *overhead* mas, sem degradar o desempenho do algoritmo. Assim, após a detecção de um cenário de alta mobilidade, o incremento do parâmetro BI deve ser mais lento para garantir que o limite superior do BI seja atingido lentamente, pois existe a probabilidade de que a ocorrência de mobilidade se repita. Já em um cenário em que ocorre um episódio isolado de mobilidade o ideal é que a subida seja um pouco mais rápida. Portanto, o valor de AF também deve ser ajustado dinamicamente de acordo com a mobilidade da rede. É importante notar que nos testes iniciais, utilizamos valores fixos para a frequência das mensagens *Hello*, e os resultados demonstraram que os métodos dinâmicos permitem obter melhores resultados em termos dos parâmetros de desempenho da rede utilizados neste trabalho

Para garantir que nenhuma mudança repentina ocorra no AF, uma escala de dez posições é definida, na qual o limite superior é denominado *Adjustment Factor Upper Limit* (AF_{ul}) e o limite inferior é denominado *Adjustment Factor Lower Limit* AF_{ll}.

Também no cenário Programado descrito na subseção 4.1.2, o tempo de convergência (CT) do e-RLRP, que é definido como o tempo decorrido entre a quebra de uma rota até que o algoritmo consiga convergir para encontrar uma nova rota, também foi avaliado. Os resultados do teste experimental demonstraram que a média do CT é de 20,6 segundos.

O valor AF não pode ser alto de forma que o parâmetro BI seja ajustado para um valor que alcance BI_{UL} antes do tempo de convergência. Então, para calcular AF_{ul} a Progressão Aritmética (PA) é

aplicada, sendo a diferença entre os termos consecutivos ou razão da PA igual a AF_{ul}, o termo A_1 da PA é BI_{LL} e o termo A_n é BI_{UL}, por fim a soma dos termos não deve ser maior que CT.

Para garantir que o valor não seja alcançado antes de 20,6 segundos, o valor é arredondado para 21 segundos. Aplicando a fórmula da soma de um PA obtém-se o valor de AF_{ul}.

$$\text{CT} \leq \frac{(\text{BI}_{\text{UL}} + \text{BI}_{\text{LL}}) \times n}{2} \tag{5.1}$$

Aplicando o resultado da Equação (5.1) na fórmula do termo geral da PA temos:

$$\text{BI}_{\text{UL}}=\text{BI}_{\text{LL}} + (n-1) \times \text{AF}_{\text{ul}} \tag{5.2}$$

O valor obtido para AF_{ul} é 1, então, o valor máximo de AF deve ser 1. Como dito anteriormente, foi definida uma escala de 10 posições. Então, o valor de AF_{ll} é 0,1 e cada posição dessa escala é aumentada em 0,1.

Sempre que ocorre mobilidade na rede, o AF é diminuído na escala. Já o aumento ocorrerá quando houver tendência de diminuição da mobilidade durante um período de tempo definido.

Este período de tempo denominado *Time of Check* (TC) é definido pela média entre o valor de CT e o tempo gasto pelo algoritmo para incrementar BI a partir de BI_{LL} até atingir o valor BI_{UL} com ajuste AF_{ll}. Para calcular TC primeiramente é aplicada a fórmula do termo geral da PA. O AF_{ll} é a razão da PA, BI_{UL} e BI_{LL} são os termos An e A1 respectivamente.

$$BI_{\text{UL}} = BI_{\text{LL}} + (n-1) * AF_{\text{ll}} \tag{5.3}$$

Aplicando o resultado da Equação (5.3) na equação da soma dos termos de uma PA e calculando a média:

$$TC = \frac{CT + \frac{(BI_{\text{LL}}+BI_{\text{UL}})*n}{2}}{2} \tag{5.4}$$

O valor obtido de TC é 99,7. Desta forma, após 99,7 segundos se houver tendência de redução da mobilidade o AF será incrementado. Para verificar se existe tendência na redução da mobilidade um contador de mobilidade denominado M_{counter} é usado para contar quantos eventos de mobilidade ocorrem no período de tempo TC. Ele será incrementado sempre que um novo vizinho for encontrado ou quando um vizinho for perdido.

$$M_{\text{counter}} = NovoVizinho_{\text{contador}} + VizinhoPerdido_{\text{contador}} \tag{5.5}$$

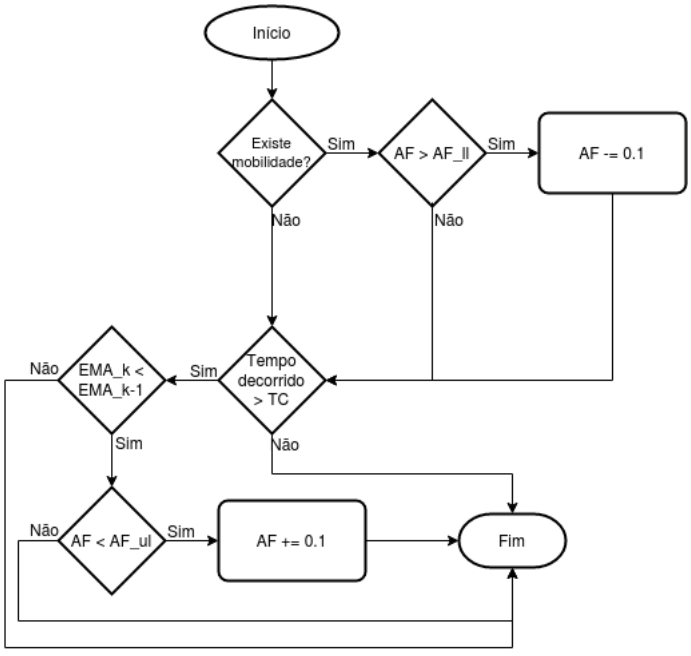
Pertencendo a uma família de abordagens estatísticas usadas para analisar dados de séries temporais na área de finanças e análise técnica (AWHEDA; SCHWARTZ, 2013), a Média Móvel Exponencial (*Exponential Moving Average*, EMA) pode ser usada para estimar valores (AWHEDA; SCHWARTZ, 2013; BURKOV; CHAIB-DRAA, 2009; TESAURO, 2004). Neste trabalho ela é usada para calcular se a ocorrência de mobilidade tende a aumentar ou diminuir de acordo com a Equação (5.6). Ela é aplicada em uma série de 10 valores M_{counter}.

$$EMA_{\text{k}} = (M_{\text{counter}} - EMA_{\text{k-1}})(\frac{2}{(N+1)}) + EMA_{\text{k-1}} \tag{5.6}$$

Se $EMA_{\text{k}} < EMA_{\text{k-1}}$ existe uma tendência na redução no número de eventos de mobilidade, então o valor de AF será decrementado na escala. O período N de 10 valores foi escolhido justamente por ser o número de vezes que AF deve ser aumentado até chegar a AF_{ul}.

O mecanismo de ajuste de AF é ilustrado na Figura 5.2.

Figura 5.2 – Mecanismo de Ajuste do AF



Fonte: Do autor (2021)

6 RESULTADOS

Para avaliar o desempenho do e-RLRP em relação aos protocolos BATMAN, OLSR e RLRP diferentes cenários de rede foram simulados. Cada cenário simulação é executado 50 vezes e o valor médio para cada cenário é calculado. A simulação de cada cenário tem duração de 600 segundos.

Nos cenários de teste foram considerados os modos de operação 2 e 8 do codec AMR-WB. Assim, as taxas de bits de transmissão consideradas foram as apresentadas na Tabela 4.2.

Primeiramente, é testado um cenário ideal onde não ocorre desconexão entre os nós. O objetivo é avaliar a redução do *overhead* de mensagens de controles proporcionado pelo e-RLRP em relação ao RLRP.

A Tabela 6.1 mostra o *overhead* no cenário de rede sem desconexão, esses resultados representam a média do *overhead* dos nós considerando os modos AMR-WB 2 e 8.

Tabela 6.1 – Overhead (kbps) Obtida no Cenário Sem Desconexão Considerando os Modos 2 e 8 do AMR-WB

Protocolo de Roteamento	**Modo 8** (kbps)	**Modo 2** (kbps)
e-RLRP	**2,29**	**2,24**
RLRP	2,71	2,65
BATMAN	6,60	6,30
OLSR	2,63	2,58

Fonte: Do autor (2021)

Como esperado, os resultados obtidos no cenário ideal sem desconexão demonstram que o e-RLRP obteve um *overhead* aproximadamente 16% menor que o RLRP. Esse resultado se deve ao fato de que o e-RLRP em um cenário sem eventos de desconexão, ou seja, sem mobilidade na rede mantém a frequência de envio de mensagens menor que o RLRP.

Em um ambiente de rede ad-hoc real, os nós podem se mover ou podem falhar, degradando o desempenho da rede. Portanto, os protocolos e-RLRP, RLRP, BATMAN e OLSR são testados em cenário onde ocorre mobilidade. Os resultados de *throughput* e Ppl, no chamado cenário P, são ilustrados na Tabela 6.2 e na Tabela 6.3, respectivamente.

Tabela 6.2 – Throughput (kbps) Obtido no Cenário P Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8 T1 (kbps)	**AMR-WB Modo-2 T1** (kbps)
e-RLRP	**39,80**	**28,60**
RLRP	39,79	28,60
BATMAN	39,28	28,25
OLSR	36,59	26,31

Fonte: Do autor (2021)

Tabela 6.3 – Ppl (%) Obtido no Cenário P Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8 T1 (%)	**AMR-WB Modo-2 T1** (%)
e-RLRP	**0,04**	**0,03**
RLRP	0,05	0,05
BATMAN	1,3	1,23
OLSR	8,08	8,02

Fonte: Do autor (2021)

Os resultados apresentados na Tabela 6.2 e Tabela 6.3 demonstram que e-RLRP e RLRP têm um desempenho melhor do que BATMAN e o OSLR.

O e-RLRP e o RLRP possuem um valor Ppl próximo a zero, pois evitam a rota que contém o nó B que apresenta desconexões recorrentes. O valor não chega a zero porque ao iniciar o roteamento, ambos os protocolos escolhem a rota do nó B que possui o menor número de saltos mas, após sucessivas desconexões do nó B, ambos os protocolos não consideram mais o uso desta rota.

Diferentemente, o protocolo OLSR escolhe a rota que contém o nó B, por ser o caminho com menor número de saltos. Quando o nó B cai, ocorre a perda de pacotes até que o OLSR convirja para uma rota alternativa. Quando o nó B se reconectar, o protocolo provavelmente retornará à rota anterior. Isso ocorre várias vezes causando um Ppl mais alto.

Apesar do protocolo BATMAN ter obtido um Ppl maior que o e-RLRP e RLRP, ele apresentou desempenho melhor que o OLSR. Isso se deve ao mecanismo de mensagens OGM.

Os resultados de *overhead* para o cenário P considerando os modos AMR-WB 2 e 8 são mostrados na Tabela 6.4.

Tabela 6.4 – Overhead (kbps) Obtida no Cenário P Considerandos os dois Modos AMR-WB

	Modo 8 (kbps)	**Modo 2** (kbps)
e-RLRP	**2,80**	**2,75**
RLRP	2,90	2,84
BATMAN	6,90	6,50
OLSR	2,88	2,80

Fonte: Do autor (2021)

Os resultados de *overhead* apresentados na Tabela 6.4 mostram que o e-RLRP reduziu o *overhead* em relação ao RLRP em aproximadamente 7%, Isso ocorre porque o e-RLRP reduziu a frequência de envio de mensagens *Hello* nos momentos em que a rede se manteve estática. Já nos momentos em que ocorreu eventos de mobilidade na rede o e-RLRP aumentou a frequência de forma a não degradar o desempenho da rede. Também podemos verificar que o e-RLRP obteve resultados melhores que os protocolos BATMAN e OLSR. A abordagem pró-ativa do OLSR e a troca constante de mensagens OGM do BATMAN proporcionou aos mesmos um *overhead* maior.

Da mesma forma, os mesmos parâmetros de desempenho da rede são avaliados no cenário R. O *throughput* e o Ppl quando são sorteadas 3 desconexões são apresentados na Tabela 6.5 e na Tabela 6.6, respectivamente.

Tabela 6.5 – Throughput (kbps) Obtido no Cenário R com Três Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
e-RLRP	**39,40**	**39,50**	**39,54**	**39,56**	**28,31**	**28,39**	**28,41**	**28,41**
RLRP	39,06	39,46	39,51	39,51	28,11	28,37	28,42	28,38
BATMAN	39,05	39,20	39,24	39,30	28,06	28,17	28,20	28,22
OLSR	37,57	38,5	39,09	39,14	27,06	27,74	28,11	28,12

Fonte: Do autor (2021)

Tabela 6.6 – Ppl (%) Obtido no Cenário R com Três Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)
e-RLRP	**1,00**	**0,73**	**0,64**	**0,58**	**1,01**	**0,72**	**0,65**	**0,63**
RLRP	1,85	0,85	0,72	0,71	1,72	0,81	0,75	0,76
BATMAN	1,86	1,49	1,39	1,25	1,79	1,51	1,35	1,32
OLSR	5,60	3,25	1,78	1,65	5,40	3,02	1,71	1.67

Fonte: Do autor (2021)

Como pode ser observado na Tabela 6.5 e Tabela 6.6, o OLSR teve o pior desempenho considerando Ppl e *throughput*, o que é explicado pelo uso de RL em e-RLRP e RLRP, e pelo mecanismo de mensagem OGM do BATMAN. O e-RLRP alcançou resultados de *throughput* semelhantes aos outros protocolos, mas em alguns cenários, o Ppl teve uma redução significativa com e-RLRP.

O *overhead* quando os nós são desconectados 3 vezes é apresentada na Tabela 6.7. Os resultados apresentados na Tabela 6.7 demonstram que o *overhead* do e-RLRP é menor que o RLRP, em alguns cenários essa redução é próxima a 18%.

Tabela 6.7 – Overhead (kbps) Obtida no Cenário R com Três Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
e-RLRP	**2,71**	**2,83**	**2,95**	**2,99**	**2,68**	**2,77**	**2,89**	**2,92**
RLRP	2,79	2,93	3,45	3,89	2,71	2,95	3,03	3,77
BATMAN	6,20	7,80	8,03	8,45	6,05	7,70	7,97	8,23
OLSR	2,74	2,94	2,99	3,27	2,70	2,86	2,95	3,18

Fonte: Do autor (2021)

O *throughput*, o Ppl e o *overhead* quando os nós são desligados 5 vezes são apresentados na Tabela 6.8, Tabela 6.9 e Tabela 6.10, respectivamente.

Tabela 6.8 – Throughpu (kbps) Obtido no Cenário R com Cinco Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1 (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)	**T1** (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)
e-RLRP	**38,98**	**39,11**	**39,28**	39,28	**28,06**	**28,20**	**28,23**	**28,21**
RLRP	38,97	39,01	39,16	39,23	28,01	28,05	28,16	28,19
BATMAN	38,48	38,20	38,88	38,95	27,68	27,47	27,87	27,99
OLSR	38,03	38,38	38,60	38,64	27,32	27,59	27,73	27,74

Fonte: Do autor (2021)

Tabela 6.9 – Ppl (%) Obtido no Cenário R com Cinco Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1 (%)	**T2** (%)	**T3** (%)	**T4** (%)	**T1** (%)	**T2** (%)	**T3** (%)	**T4** (%)
e-RLRP	**2,10**	**1,71**	**1,29**	**1,30**	**1,90**	**1,67**	**1,28**	**1,25**
RLRP	**2,10**	1,98	1,60	1,54	2,05	1,91	1,54	1,50
BATMAN	3,30	4,01	2,30	2,10	3,20	3,91	2,50	2,30
OLSR	4,44	3,56	3,01	2,95	4,47	3,52	3,04	2,98

Fonte: Do autor (2021)

Tabela 6.10 – Overhead (kbps) Obtida no Cenário R com Cinco Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1 (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)	**T1** (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)
e-RLRP	**2,90**	**2,98**	**3,01**	**3,03**	**2,86**	**2,94**	**2,98**	**3,02**
RLRP	2,94	3,14	3,88	3,95	2,92	3,09	3,78	3,90
BATMAN	11,05	11,20	11,40	11,48	11,01	11,00	11,30	11,38
OLSR	3,16	3,70	4,67	4,72	3,12	3,75	4,52	4,61

Fonte: Do autor (2021)

De acordo com os resultados obtidos em um cenário de 5 desconexões o e-RLRP e o algoritmo RLRP tiveram um desempenho melhor que o BATMAN e o OLSR. Além disso, o e-RLRP obteve redução do *overhead* e valores de Ppl menores em relação ao RLRP.

Da mesma forma, o *throughput*, o Ppl e o *overhead* quando os nós são desconectados 7 vezes, são apresentados na Tabela 6.11, Tabela 6.12 e Tabela 6.13, respectivamente.

Tabela 6.11 – Throughput (kbps) Obtido no Cenário R com Sete Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1 (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)	**T1** (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)
e-RLRP	**38,98**	**39,15**	**39,13**	39,19	**28,01**	**28,15**	**28,14**	**28,16**
RLRP	38,97	38,95	38,98	**39,01**	28,00	27,99	28,06	28,19
BATMAN	37,19	37,63	37,47	37,49	26,73	27,05	27,00	27,03
OLSR	36,77	37,98	38,52	38,55	26,40	27,30	27,71	27,73

Fonte: Do autor (2021)

Tabela 6.12 – Ppl (%) Obtido no Cenário R com Sete Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1 (%)	**T2** (%)	**T3** (%)	**T4** (%)	**T1** (%)	**T2** (%)	**T3** (%)	**T4** (%)
e-RLRP	2.10	**1.61**	**1.59**	1.55	**2.01**	**1.57**	**1.60**	**1.54**
RLRP	**2.09**	2.13	2.10	**1.45**	2.05	2.12	1.90	1.83
BATMAN	6.55	5.45	5.85	5.62	6.56	5.39	5.60	5.49
OLSR	7.60	4.56	3.20	3.10	7.70	4.54	3.10	3.01

Fonte: Do autor (2021)

Tabela 6.13 – Overhead (kbps) Obtida no Cenário R com Sete Desconexões Considerando os Modos 2 e 8 do AMR-WB

	AMR-WB Modo-8				**AMR-WB Modo-2**			
	T1 (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)	**T1** (kbps)	**T2** (kbps)	**T3** (kbps)	**T4** (kbps)
e-RLRP	**2,98**	**3,41**	**4,84**	**4,99**	**2,94**	**3,25**	**4,10**	**4,96**
RLRP	3,10	3,64	5,29	5,45	3,01	3,39	5,36	5,25
BATMAN	11,12	11,15	12,10	12,35	11,21	11,13	12,01	12,49
OLSR	3,26	3,79	5,10	5,17	3,20	3,85	4,99	5,01

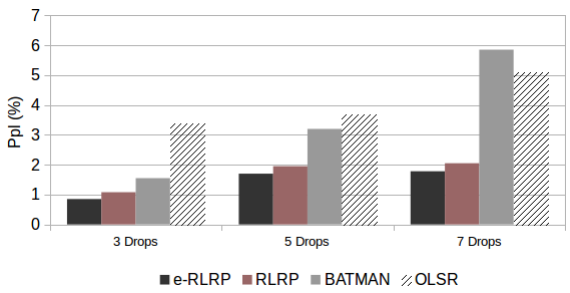
Fonte: Do autor (2021)

De acordo com os resultados apresentados, no cenário em que ocorrem 7 desconexões o e-RLRP obtém melhor desempenho em todos os casos em relação ao BATMAN e OLSR, e também apresenta um desempenho melhor que o RLRP na maioria dos cenários.

Em geral, o ganho de desempenho nos cenários R foi menor do que no cenário P. Esse comportamento ocorre porque, no cenário P as desconexões são recorrentes em apenas uma rota, o que facilita o processo de aprendizagem do e-RLRP.

A Figura 6.1 demonstra a melhoria de desempenho do e-RLRP em relação aos outros protocolos. Os valores Ppl obtidos são a média dos resultados obtidos nas quatro topologias e ambos os modos de taxas AMR-WB usados nos testes.

Figura 6.1 – Desempenho do e-RLRP em Termos de Ppl Considerando Diferente Números de Desconexões



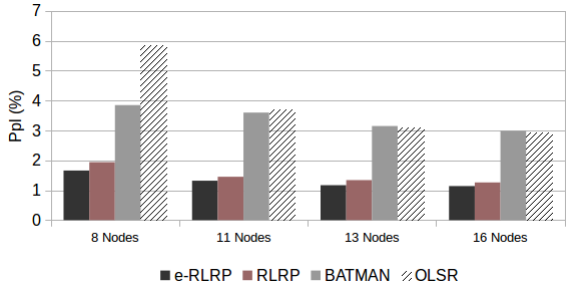
Fonte: Do autor (2021)

A partir dos resultados podemos concluir que o desempenho do e-RLRP em relação aos outros três protocolos aumenta à medida que aumenta o número de desconexões. Ao aumentar o número de des-

conexões, o desempenho de todos os algoritmos se degrada, porém, no e-RLRP e RLRP essa degradação é menor.

A Figura 6.2 mostra a relação entre o desempenho do e-RLRP e o número de nós na rede. É importante notar que quanto maior o número de nós na rede, maior o processamento necessário pelo algoritmo RL para determinar os valores de recompensa. Apesar do aumento do processamento de RL, o desempenho obtido pelo e-RLRP, em termos de Ppl, é superior em relação aos demais protocolos de roteamento.

Figura 6.2 – Desempenho do e-RLRP em Termos de Ppl Considerando Diferente Números de Nós



Fonte: Do autor (2021)

O *throughput*, Ppl e o *overhead* para o cenário com três fluxos considerando o modo 8 do AMR-WB são apresentados na Tabela 6.14, na Tabela 6.15 e na Tabela 6.16 respectivamente.

Tabela 6.14 – Throughput (kbps) Obtido no Cenário R Com Três Fluxos Considerando o AMR-WB Modo 8

	Modo 8					
	T3			**T4**		
	3 Desconexões CE/FB/IH (kbps)	**5 Desconexões** CE/FB/IH (kbps)	**7 Desconexões** CE/FB/IH (kbps)	**3 Desconexões** CE/FB/IH (kbps)	**5 Desconexões** CE/FB/IH (kbps)	**7 Desconexões** CE/FB/IH (kbps)
e-RLRP	**38,50/38,38/**38,20	**38,16/**38,11/**38,33**	**37,81/37,60/38,10**	**38,49/38,40/38,27**	**38,15/38,17/38,32**	**37,91/37,77/38,06**
RLRP	38,33/38,19/38,18	38,10/**38,14/**38,28	37,77/37,59/37,50	38,40/38,32/38,25	38,14/38,16/38,29	37,87/37,75/37,83
BATMAN	38,17/38,18/**38,37**	37,60/37,71/37,72	36,89/36,90/37,77	38,18/38,22/38,26	37,71/37,83/37,79	36,83/37,02/36,95
OLSR	37,93/38,11/38,18	37,32/37,84/38,01	37,46/37,59/37,44	38,04/38,22/38,26	37,48/37,96/38,06	37,52/37,66/37,56

Fonte: Do autor (2021)

Tabela 6.15 – Ppl (%) Obtido no Cenário R Com Três Fluxos Considerando o AMR-WB Modo 8

	Modo 8					
	T3			**T4**		
	3 Desconexões CE/FB/IH (%)	**5 Desconexões** CE/FB/IH (%)	**7 Desconexões** CE/FB/IH (%)	**3 Desconexões** CE/FB/IH (%)	**5 Desconexões** CE/FB/IH (%)	**7 Desconexões** CE/FB/IH (%)
e-RLRP	**0,69/0,63/**1,03	1,40/**1,10/0,82**	**2,05/2,63/1,99**	**0,51/0,51/0,85**	**1,17/1,10/0,72**	**1,80/2,20/1,40**
RLRP	0,71/1,06/1,09	**1,31/**1,19/0,85	2,15/2,66/2,86	0,52/0,74/0,91	1,20/1,15/0,80	1,90/**2,20/**2,00
BATMAN	1,41/1,18/**1,01**	2,70/2,40/2,28	5,40/4,60/5,10	1,10/0,98/0,88	2,30/2,00/2,10	4,60/4,10/4,20
OLSR	1,75/1,27/1,1	3,41/1,96/1,52	2,95/2,68/3,01	1,45/0,98/0,89	2,90/1,67/1,40	2,80/2,45/2,70

Fonte: Do autor (2021)

Da mesma forma, o *throughput*, o Ppl e o *overhead* para três fluxos considerando AMR-WB Modo 2, são apresentados na Tabela 6.17, Tabela 6.18 e Tabela 6.19 respectivamente.

Tabela 6.16 – Overhead (kbps) Obtida no Cenário R Com Três Fluxos Considerando o AMR-WB Modo 8

	Modo 8					
	T3			**T4**		
	3 Desconexões (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)	**3 Desconexões** (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)
e-RLRP	**2,78**	**3,33**	**5,92**	**2,81**	**3,35**	**5,17**
RLRP	4,78	4,72	4,95	4,90	4,82	5,41
BATMAN	9,90	12,20	15,45	10,20	11,98	14,85
OLSR	3,92	5,10	6,80	4,10	5,33	5,21

Fonte: Do autor (2021)

Tabela 6.17 – Throughput (kbps) Obtido no Cenário R Com Três Fluxos Considerando o AMR-WB Modo 2

	Modo 2					
	T3			**T4**		
	3 Desconexões CE/FB/IH (kbps)	**5 Desconexões** CE/FB/IH (kbps)	**7 Desconexões** CE/FB/IH (kbps)	**3 Desconexões** CE/FB/IH (kbps)	**5 Desconexões** CE/FB/IH (kbps)	**7 Desconexões** CE/FB/IH (kbps)
e-RLRP	**27,84/27,87/27,77**	**27,86**/27,65/**27,82**	**27,48/ 27,31/27,67**	**27,85/27,86/27,76**	**27,65/27,71/27,84**	**27,50/27,35/27,69**
RLRP	27,83/27,71/27,73	27,80/**27,70**/27,66	27,46/27,30/27,23	27,84/ 27,76/27,75	27,65/27,74/27,82	27,48/27,34/27,20
BATMAN	27,69/27,74/27,76	27,60/27,63/27,66	26,69/26,97/26,87	27,70/ 27,75/27,75	27,63/27,68/27,71	26,87/26,91/26,80
OLSR	27,57/27,66/**27,77**	27,06/27,49/27,61	27,23/27,29/27,20	27,61/27,65/27,71	27,24/27,48/27,65	27,21/27,24/27,23

Fonte: Do autor (2021)

Tabela 6.18 – Ppl (%) Obtido no Cenário R Com Três Fluxos Considerando o AMR-WB Modo 2

	Modo 2					
	T3			**T4**		
	3 Desconexões CE/FB/IH (%)	**5 Desconexões** CE/FB/IH (%)	**7 Desconexões** CE/FB/IH (%)	**3 Desconexões** CE/FB/IH (%)	**5 Desconexões** CE/FB/IH (%)	**7 Desconexões** CE/FB/IH (%)
e-RLRP	**0,64/0,56/0,99**	**1,14**/1,27/**0,71**	**1,95/2,58/1,25**	**0,62/0,54/0,85**	**1,32/1,01/0,59**	**1,86/2,39/1,19**
RLRP	0,68/1,03/1,11	1,20/**1,15**/0,78	2,02/2,59/2,84	0,64/0,95/0,98	1,32/1,02/0,72	1,95/2,42/2,69
BATMAN	1,19/1,01/1,03	1,70/1,40/1,28	4,50/4,10/4,30	1,15/ 0,98/0,96	1,42/1,22/1,10	4,10/3,98/4,35
OLSR	1,62/1,28/0,90	3,45/1,90/1,47	2,83/2,60/2,94	1,49/1,23/0,86	2,67/1,90/1,32	2,71/2,62/2,83

Fonte: Do autor (2021)

Tabela 6.19 – Overhead (kbps) Obtida no Cenário R Com Três Fluxos Considerando o AMR-WB Modo 2

	Modo 2					
	T3			**T4**		
	3 Desconexões (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)	**3 Desconexões** (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)
e-RLRP	**2,62**	**3,09**	**6,89**	**3,02**	**2,97**	**5,14**
RLRP	4,16	4,76	4,92	4,21	4,80	5,62
BATMAN	9,70	11,45	14,26	9,23	11,52	13,52
OLSR	3,86	4,95	5,76	3,49	4,88	5,77

Fonte: Do autor (2021)

Analisando o cenário com 3 fluxos pode-se verificar que o e-RLRP supera na maioria dos casos os demais protocolos considerando Ppl e Troughput. Em relação ao *overhead* o e-RLRP alcançou os melhores resultados em todos os cenários.

O *throughput*, o Ppl e o *overhead* para o quatro fluxos, considerando AMR-WB Modo 8 são apresentados na Tabela 6.20, Tabela 6.21 e Tabela 6.22 respectivamente.

Da mesma forma, o *throughput*, o Ppl e o *overhead* para o cenário quatro fluxos considerando AMR-WB Modo 2 são apresentados na Tabela 6.23, Tabela 6.24 e Tabela 6.25 respectivamente.

Verifica-se nos resultados dos cenários de 4 fluxos que o e-RLRP supera outros protocolos na maioria dos casos em termos de *throughput* e Ppl. Em relação ao *overhead* o e-RLRP obteve os melhores

Tabela 6.20 – Throughput (kbps) Obtido no Cenário R Com Quatro Fluxos Considerando o AMR-WB Modo 8

	Mod0 8					
	T3			**T4**		
	3 Desconexões CE/FB/IH/MK (kbps)	**5 Desconexões** CE/FB/IH/MK (kbps)	**7 Desconexões** CE/FB/IH/MK (kbps)	**3 Desconexões** CE/FB/IH/MK (kbps)	**5 Desconexões** CE/FB/IH/MK (kbps)	**7 Desconexões** CE/FB/IH/MK (kbps)
e-RLRP	**38,40/38,23/38,29**/39,8	**38,23**/37,98/**38,26/39,8**	**38,40/37,93/38,26**/39,8	**38,41/38,29/38,29**/39,8	**38,28**/38,09/**38,21**/39,8	**38,04/37,87**/37,86/39,8
RLRP	38,37/**38,23**/38,27/39,01	38,14/**38,16**/38,20/39,6	37,89/37,79/38,18/39,8	38,39/38,28/38,29/39,8	38,15/**38,07**/38,18/39,8	38,02/37,82/**37,87**/39,8
BATMAN	37,99/38,06/38,15/39,7	37,60/37,49/37,53/39,8	36,52/36,71/36,79/39,8	38,04/38,10/38,22/39,8	37,71/37,54/37,51/39,8	36,72/36,92/36,81/39,8
OLSR	38,22/38,18/38,21/39,8	37,40/37,61/38,07/39,8	37,45/37,71/37,71/39,8	38,25/38,22/38,25/39,8	37,46/37,59/38,10/39,8	37,46/37,60/37,62/39,8

Fonte: Do autor (2021)

Tabela 6.21 – Ppl (%) Obtido no Cenário R Com Quatro Fluxos Considerando o AMR-WB Modo 8

	Modo 8					
	T3			**T4**		
	3 Desconexões CE/FB/IH/MK (%)	**5 Desconexões** CE/FB/IH/MK (%)	**7 Desconexões** CE/FB/IH/MK (%)	**3 Desconexões** CE/FB/IH/MK (%)	**5 Desconexões** CE/FB/IH/MK (%)	**7 Desconexões** CE/FB/IH/MK (%)
e-RLRP	**0,52**/0,98/**0,92**/0	**0,96**/1,70/**1,14**/0	**1,79/2,30/1,85**/0	**0,49**/0,85/ **0,81**/0	**0,83**/1,40/**1,01**/0	**1,45/1,89/1,81**/0
RLRP	0,61/**0,95**/0,96/0	1,22/**1,25**/1,15/0	1,83/2,31/1,93/0	0,55/**0,84/0,81**/0	1,17/**1,35**/1,10/0	1,50/1,96/1,89/0
BATMAN	1,59/1,41/1,16/0	2,58/2,89/2,78/0	5,40/4,90/4,70/0	1,46/ 1,31/0,98/0	2,30/2,76/2,65/0	4,78/4,30/4,65/0
OLSR	0,98/1,10/1,02/0	3,11/2,57/1,38/0	3,08/2,68/2,64/0	0,92/0,98/0,91/0	2,95/2,63/1,29/0	2,95/2,59/2,43/0

Fonte: Do autor (2021)

resultados em todos os cenários testados. Além disso, vale a pena mencionar que o *overhead* em geral aumenta quando há mais fluxos, entretanto, especificamente o *overhead* gerado pela mensagem de con-

Tabela 6.22 – Overhead (kbps) Obtida no Cenário R Com Quatro Fluxos Considerando o AMR-WB Modo 8

	Modo 8					
	T3			**T4**		
	3 Desconexões (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)	**3 Desconexões** (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)
e-RLRP	**3,15**	**4,19**	**4,20**	**3,25**	**4,37**	**5,60**
RLRP	3,45	5,45	4,92	3,49	5,6	5,70
BATMAN	9,22	10,20	13,03	9,3	10,10	13,02
OLSR	3,22	4,72	5,30	3,33	4,68	5,90

Fonte: Do autor (2021)

Tabela 6.23 – Throughput (kbps) Obtido no Cenário R Com Quatro Fluxos Considerando o AMR-WB Modo 2

	Modo 2					
	T3			**T4**		
	3 Desconexões CE/FB/IH (kbps)	**5 Desconexões** CE/FB/IH (kbps)	**7 Desconexões** CE/FB/IH (kbps)	**3 Desconexões** CE/FB/IH (kbps)	**5 Desconexões** CE/FB/IH (kbps)	**7 Desconexões** CE/FB/IH (kbps)
e-RLRP	**27,58/27,48/27,50**/28,6	**27,51**/27,29/**27,43**/28,6	**27,29/27,15**/27,28/28,6	**27,88**/27,72/**27,79**/28,6	**27,81**/27,60/**27,74**/28,6	27,61/**27,41/27,65**/28,6
RLRP	27,57/**27,48**/27,49/28,6	27,35/**27,42**/27,41/28,6	27,20/27,13/**27,40**/28,6	27,85/**27,77**/27,78/28,6	27,80/**27,62**/27,72/28,6	**27,68**/27,37/27,62/28,5
BATMAN	27,32/27,31/27,39/28,6	26,99/26,94/26,91/28,5	26,27/26,52/26,60/28,6	27,61/27,64/27,73/28,6	27,35/27,31/27,33/28,6	26,62/26,87/26,93/28,6
OLSR	27,29/27,38/27,49/28,6	26,78/27,21/27,33/28,6	26,95/27,02/26,92/28,5	27,61/27,64/27,73/28,6	27,35/27,31/27,33/28,6	26,62/26,87/26,93/28,6

Fonte: Do autor (2021)

Tabela 6.24 – Ppl (%) Obtido no Cenário R Com Quatro Fluxos Considerando o AMR-WB Modo 2

	Modo 2					
	T3			**T4**		
	3 Desconexões CE/FB/IH (%)	**5 Desconexões** CE/FB/IH (%)	**7 Desconexões** CE/FB/IH (%)	**3 Desconexões** CE/FB/IH (%)	**5 Desconexões** CE/FB/IH (%)	**7 Desconexões** CE/FB/IH (%)
e-RLRP	**0,58/0,90/0,85**/0	**0,87**/1,60/**1,11**/0	**1,62/2,10**/1,65/0	**0,52/0,86/0,83**/0	**0,75/1,40/1,01**/0	1,47/**2,20/1,32**/0
RLRP	0,59/0,91/0,89/0	1,38/**1,16**/1,19/0	1,93/2,20/**1,63**/0	0,62/0,84/ 0,84/0	**0,75**/1,41/1,05/0	**1,23**/2,32/1,43/0
BATMAN	1,52/1,55/1,26/0	2,70/2,86/2,98/0	5,30/4,40/4,10/0	1,49/1,35/1,03/0	2,41/2,56/2,49/0	5,01/4,1/3,89/0
OLSR	1,62/1,28/0,90/0	3,45/1,90/1,47/0	2,83/2,60/2,94/0	1,49/1,35/1,03/0	2,41/2,56/2,49/0	5,01/4,1/3,89/0

Fonte: Do autor (2021)

Tabela 6.25 – Overhead (kbps) Obtida no Cenário R Com Quatro Fluxos Considerando o AMR-WB Modo 2

	Modo 2					
	T3			**T4**		
	3 Desconexões (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)	**3 Desconexões** (kbps)	**5 Desconexões** (kbps)	**7 Desconexões** (kbps)
e-RLRP	3,06	**4,26**	**4,13**	**3,28**	**4,10**	4,46
RLRP	3,88	5,25	4,61	3,72	4,55	4,88
BATMAN	8,26	9,04	11,65	8,25	9,16	11,33
OLSR	2,96	4,66	5,30	3,95	4,68	5,11

Fonte: Do autor (2021)

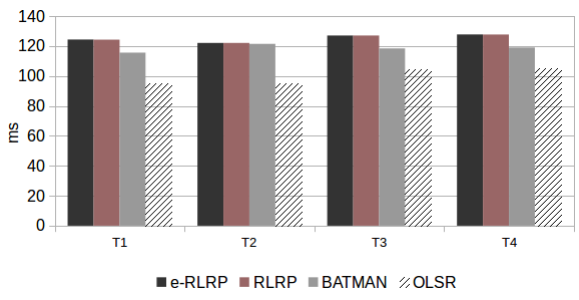
trole *Hello* não é tão impactado. Isso ocorre porque as mensagens *Hello* são trocadas independentemente do número de fluxos na rede.

Nos resultados apresentados nas Tabelas 6.21 e 6.24 em relação ao Ppl, podemos observar que um dos fluxos de tráfego (par MK) atingiu um Ppl quase igual a O pois existe uma rota direta entre os dois pares de nós e nenhuma queda ocorreu neste caminho. Os fluxos extras foram adicionados para sobrecarregar a rede.

Analisando os resultados dos cenários com mais de um fluxo de tráfego, especificamente três e quatro fluxos é possível observar que o e-RLRP supera os demais protocolos de roteamento na maioria dos cenários de rede testados em termos de Ppl e *throughput*. Em relação ao *overhead* podemos observar que o e-RLRP continua superando os demais protocolos. Os resultados experimentais confirmaram que o e-RLRP obteve um *overhead* menor que o RLRP na maioria dos cenários, mesmo quando o número de fluxos de tráfego, o número de rotas ou desconexões de nó foram aumentados. Assim, demonstraram que a função de ajuste proposta funcionou adequadamente na tarefa de redução do *overhead*.

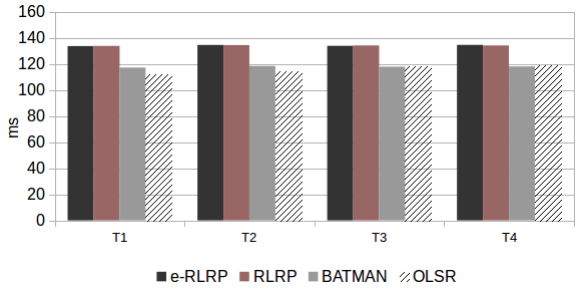
Adicionalmente, os valores dos parâmetros de RTT obtidos no cenário P são apresentados na Figura 6.3, e a Figura 6.4 mostra a média dos valores RTT dos cenários R também com um único fluxo. Esses resultados representam os valores médios dos dois modos AMR-WB pois não houve diferença significativa entre eles.

Figura 6.3 – RTT Obtido no Cenário P (Progamado)



Fonte: Do autor (2021)

Figura 6.4 – RTT Obtido no Cenário R (Randomico)

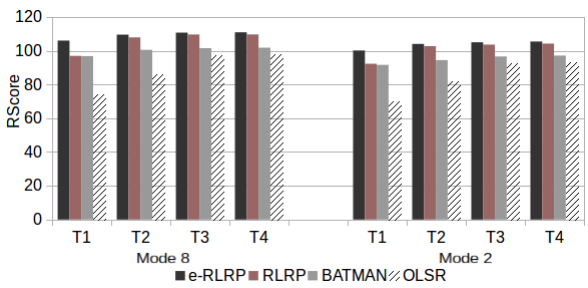


Fonte: Do autor (2021)

Analisando os resultados apresentados na Figura 6.3 e Figura 6.4, observa-se que o e-RLRP e RLRP apresentaram os maiores valores de RTT. Isso pode ser justificado porque eles são implementados no espaço do usuário no Linux usando um interpretador Python dinâmico. Segundo o (DUGAEV et al., 2018) este tipo de implementação gera uma grande perda de performance principalmente devido ao grande número de operações de I/O que causam atrasos no processo de envio de pacotes. Vale ressaltar que esta é uma limitação gerada pela linguagem em que foi implementado e não pelo código/projeto. Assim, a implementação desses dois protocolos teve uma restrição nesse sentido que se refletiu nos valores de RTT obtidos nos testes experimentais. Vale resaltar que de acordo com (2.15) e (2.16) atrasos na rede têm um impacto negativo nas previsões de qualidade de voz.

Por fim, a qualidade da comunicação de fala foi avaliada usando (2.13), (2.14) e (2.15) considerando os valores de Ppl e RTT obtidos nos cenários de teste que consideram um único fluxo de tráfego com as topologias T1, T2, T3 e T4 utilizados neste trabalho. A Figura 6.5 apresenta as pontuações de R_{WB} para o cenário R com três desconexões, a Figura 6.6 apresenta as pontuações de R_{WB} para o cinco desconexões e a Figura 6.7 o cenário com sete desconexões.

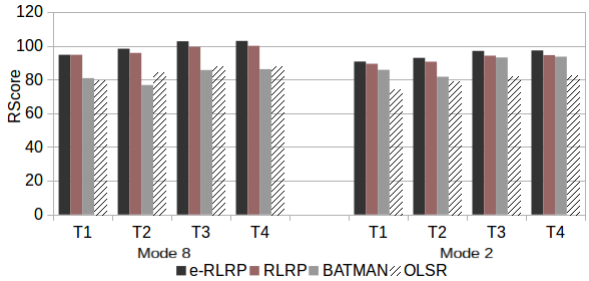
Figura 6.5 – R_{WB} Score no Cenário com 3 Desconexões



Fonte: Do autor (2021)

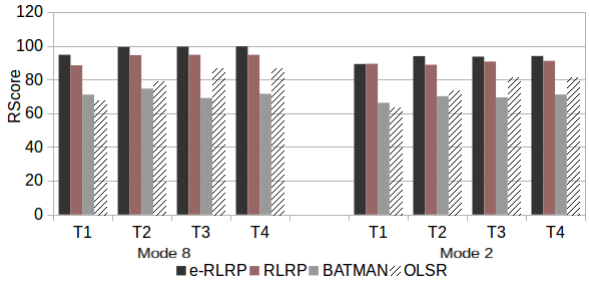
A figura 6.8 ilustra o R_{WB} para o cenário P.

Figura 6.6 – R_{WB} Score no Cenário com 5 Desconexões



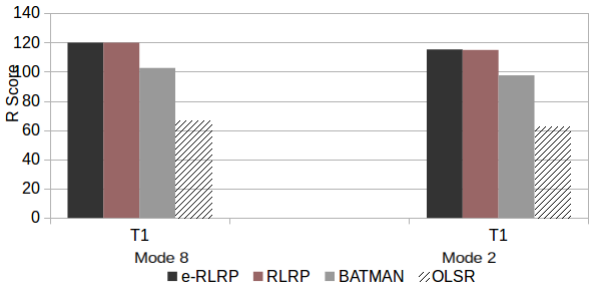
Fonte: Do autor (2021)

Figura 6.7 – R_{WB} Score no Cenário com 7 Desconexões



Fonte: Do autor (2021)

Figura 6.8 – R_{WB} Score no Cenário P



Fonte: Do autor (2021)

Como pode ser observado da Figura 6.5 e na Figura 6.8 o uso do e-RLRP promove um ganho de R_{WB} score em relação aos obtidos pelos protocolos RLRP, BATMAN e OLSR. Em alguns casos o ganho em relação ao OLSR é de mais de 90%. Em relação ao BATMAN, em alguns casos o ganho é de aproximadamente 33%. Já em relação ao RLRP o ganho se aproxima de 8%.

Portanto, a RL em protocolos de roteamento melhora a QoE do usuário em um serviço de comunicação por voz. O e-RLRP não só reduz o *overhead*, mas também proporciona um impacto positivo na qualidade da comunicação VoIP, principalmente porque o Ppl é reduzido.

7 CONCLUSÃO

Neste trabalho, os resultados experimentais demonstram que um protocolo de roteamento baseado em RL supera os protocolos tradicionais, como BATMAN e OLSR, especificamente em parâmetros de Probabilidade de Perda de Pacotes (Probability of Packet Loss, Ppl) e *throughput*. Esses resultados de desempenho de rede comprovam a relevância dos protocolos de roteamento baseados em RL para o desempenho de redes ad-hoc. No entanto, a técnica RL neste caso gera um *overhead* maior. Assim, o algoritmo de ajuste proposto e desenvolvido foi capaz de reduzir o *overhead* da rede em termos de redução do número de mensagens de controle. O ajuste dinâmico na frequência de envio de mensagens *Hello* proporcionou uma redução de até 18% do *overhead*. Esse ganho aumenta a carga útil da rede proporcionando melhor desempenho da mesma. O desempenho global do método proposto foi otimizado utilizando diferentes valores de configuração e parâmetros, cuja configuração final foi definida experimentalmente. Em termos de *throughput* e Ppl na maioria dos cenários de teste utilizados neste trabalho o e-RLRP obteve melhor desempenho. Portanto, é demonstrado que a solução proposta reduz o *overhead* e também melhora as condições da rede. É importante notar que em nossos testes experimentais foram utilizadas diferentes topologias e configurações de rede, incluindo diferentes números de nós, número de desconexões e também diferentes números de fluxos de tráfego.

Além disso, resultados experimentais mostram o impacto dos parâmetros de desempenho da rede na QoE do usuário nos serviços de comunicação VoIP. O e-RLRP obteve melhores valores de R_{WB} por ter valores de Ppl menores apesar de ter os maiores valores de RTT. O R_{WB} é calculado de acordo com (2.15) e (2.16) definidos no algoritmo do E-model WB. Nesse caso, observa-se que o Ppl tem maior impacto na qualidade da fala do que o RTT, para os valores obtidos nos cenários de simulação considerados nesta pesquisa. Os resultados indicam uma melhoria de qualidade de mais de 90% se comparado ao OLSR e de até 8% se comparado ao RLRP. Portanto, pode-se concluir que os protocolos de roteamento baseados em RL têm um impacto positivo significativo na QoE do usuário em serviços de comunicação em tempo real.

Como conclusão geral, esta pesquisa destaca a utilidade da incorporação de algoritmos de aprendizado de máquina em protocolos de roteamento, especialmente para redes ad-hoc que podem apresentar recorrentes desconexões nos nós. Os protocolos de roteamento baseados em RL podem ajudar a melhorar as condições da rede e como consequência diferentes aplicativos de comunicação podem ser impactados positivamente. Neste trabalho apenas o serviço VoIP é avaliado, mas em trabalhos futuros também será avaliado o serviço de videocomunicação.

Reduzir o *overhead* da rede em protocolos convencionais é uma abordagem importante, porque fornece melhorias de desempenho. Essa abordagem é ainda mais relevante quando se é utilizado novas

técnicas de roteamento como o uso de RL, que visam melhorar o desempenho da rede, mas que para tanto podem requerer um cabeçalho extra o que gera um *overhead* maior ainda. Assim, uma contribuição importante deste trabalho é demonstrar que o *overhead* pode ser reduzida usando a função de ajuste dinâmico da frequência de envio de mensagens *Hello*, proporcionando assim um desempenho melhor ainda ao uso de RL em protocolos de roteamento.

Para trabalhos futuros também vale destacar a possibilidade de usar ajuste dinâmico também na frequência do envio de mensagens ACK e do tempo de espera para gerar um punição. De forma geral, podemos dizer que em uma rede estática a frequência de envio de mensagens ACK pode ser reduzida. Já em um rede com alta mobilidade pressupõe-se que o tempo de espera para gerar uma punição seja menor uma vez que a rede está em constante mudança e aumenta as chances do algoritmo estar esperando receber uma confirmação de um vizinho que não se encontra mais ao seu alcance.

REFERÊNCIAS

3GPP TS 26.171. *Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; General Description (v15.0.0)*. 2018.

AHRENHOLZ, J. et al. Core: A real-time network emulator. In: **MILCOM 2008 - 2008 IEEE Military Communications Conference**. [S.l.: s.n.], 2008. p. 1–7.

ASADI, K.; LITTMAN, M. L. An alternative softmax operator for reinforcement learning. In: **Proceedings of the 34th International Conference on Machine Learning - Volume 70**. [S.l.: s.n.], 2017. (ICML'17), p. 243–252.

AWHEDA, M. D.; SCHWARTZ, H. M. Exponential moving average q-learning algorithm. In: **2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)**. [S.l.: s.n.], 2013. p. 31–38.

BANGOTRA, D. K. et al. An intelligent opportunistic routing algorithm for wireless sensor networks and its application towards e-healthcare. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 14, p. 3887, 2020.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006.

BURKOV, A.; CHAIB-DRAA, B. Effective learning in the presence of adaptive counterparts. **Journal of Algorithms**, Academic Press, v. 64, n. 4, p. 127–138, 2009.

CETIC.BR. **EMPRESAS QUE UTILIZARAM A INTERNET, POR TIPO DE ATIVIDADE NOS ÚLTIMOS 12 MESES**. 2017. Acesso em: 10 de Setembro de 2020. Disponível em: <hhttps://cetic.br/tics/empresas/2017/empresas/B5/>.

CLAUSEN, T. et al. Optimized link state routing protocol (olsr). 2003.

COUTINHO, N. et al. Dynamic dual-reinforcement-learning routing strategies for quality of experience-aware wireless mesh networking. **Computer Networks**, v. 88, p. 269 – 285, 2015. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128615002170>.

COUTO, D. S. D. et al. A high-throughput path metric for multi-hop wireless routing. In: **Proceedings of the 9th annual international conference on Mobile computing and networking**. [S.l.: s.n.], 2003. p. 134–146.

DHOUIB, M. A. et al. Multi-agent system for remote healthcare monitoring. In: SPRINGER. **Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014**. [S.l.], 2014. p. 1–12.

DHURANDHER, S. K.; OBAIDAT, M. S.; Gupta, M. A reactive optimized link state routing protocol for mobile ad hoc networks. In: **2010 17th IEEE International Conference on Electronics, Circuits and Systems**. [S.l.: s.n.], 2010. p. 367–370.

DING, R. et al. Deep reinforcement learning for router selection in network with heavy traffic. **IEEE Access**, v. 7, p. 37109–37120, 2019.

DUGAEV, D. et al. Adaptive reinforcement learning-based routing protocol for wireless multihop networks. **International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering**, p. 209–218, 10 2018.

FAINELLI, F. The openwrt embedded development framework. In: **Proceedings of the Free and Open Source Software Developers European Meeting**. [S.l.: s.n.], 2008. p. 106.

GARCIA, M. N.; RAAKE, A. Parametric packet-layer video quality model for iptv. In: **10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)**. [S.l.: s.n.], 2010. p. 349–352.

GIRUKA, V. C.; SINGHAL, M. Hello protocols for ad-hoc networks: overhead and accuracy tradeoffs. In: **Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks**. [S.l.: s.n.], 2005. p. 354–361.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. book in preparation for mit press. **URL¡ http://www. deeplearningbook. org**, 2016.

GUIMARÃES, R. et al. Recommendation system using sentiment analysis considering the polarity of the adverb. In: IEEE. **2016 IEEE International Symposium on Consumer Electronics (ISCE)**. [S.l.], 2016. p. 71–72.

HANDLEY, M. et al. Sip: session initiation protocol. **IETF RFC 3261**, v. 3261, 01 1999.

HARTPENCE, B. **Packet Guide to Voice over IP: A system administrator's guide to VoIP technologies**. [S.l.]: "O'Reilly Media, Inc.", 2013.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007.

HUANG, H. et al. Three-dimensional geographic routing in wireless mobile ad hoc and sensor networks. **IEEE Network**, v. 30, n. 2, p. 82–90, March 2016. ISSN 1558-156X.

HUNG, L.-L. A smart sensing and routing mechanism for wireless sensor networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 19, p. 5720, 2020.

ITU, T. Recommendation g. 107 the e-model, a computational model for use in transmission planning. 2003.

ITU-T Rec. G.113 - Appendix IV. **Revised Appendix IV - Provisional planning values for the wideband equipment impairment factor and the wideband packet loss robustness factor**. 2009. Disponível em: <https://www.itu.int/rec/T-REC-G.113-200903-I!Amd1/en>.

ITU-T Rec. P.800. **Methods for subjective determination of transmission quality**. 1996. Disponível em: <www.itu.int/rec/T-REC-P.800-199608-I/en.>

ITU-T Rec. P.862.1. **Mapping function for transforming P.862 raw result scores to MOS-LQO**. 2003. Disponível em: <https://www.itu.int/rec/T-REC-P.862.1-200311-I/en.>

JUNG, W.; YIM, J.; KO, Y. Qgeo: Q-learning-based geographic ad hoc routing protocol for unmanned robotic networks. **IEEE Communications Letters**, v. 21, n. 10, p. 2258–2261, 2017.

KIM, H. J. et al. The qoe evaluation method through the qos-qoe correlation model. In: **2008 Fourth International Conference on Networked Computing and Advanced Information Management**. [S.l.: s.n.], 2008. v. 2, p. 719–725.

KOULOURIOTIS, D.; XANTHOPOULOS, A. Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems. **Applied Mathematics and Computation**, v. 196, n. 2, p. 913 – 922, 2008. ISSN 0096-3003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0096300307007448>.

KUIPERS, F. et al. Techniques for measuring quality of experience. In: SPRINGER. **International Conference on Wired/Wireless Internet Communications**. [S.l.], 2010. p. 216–227.

KUMARI, B. et al. An efficient algebraic codebook structure for cs-acelp based speech codecs. In: **2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)**. [S.l.: s.n.], 2017. p. 1–6.

KUROSE, J.; ROSS, K. **Redes de computadores e a internet: uma abordagem top-down**. ADDISON WESLEY BRA, 2010. ISBN 9788588639973. Disponível em: <https://books.google.com.br/books?id=raZtQwAACAAJ>.

LAMPING, U.; WARNICKE, E. Wireshark user's guide. interface. v. 4, 2004.

LASMAR, E. L. et al. Rsrs: Ridesharing recommendation system based on social networks to improve the user's qoe. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 20, n. 12, p. 4728–4740, 2019.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

LI, G.; BOUKHATEM, L.; MARTIN, S. An intersection-based qos routing in vehicular ad hoc networks. **Mobile Networks and Applications**, Springer, v. 20, n. 2, p. 268–284, 2015.

LUO, Y. et al. $\mathscr{G}$ -softmax: Improving intraclass compactness and interclass separability of features. **IEEE Transactions on Neural Networks and Learning Systems**, v. 31, n. 2, p. 685–699, 2020.

MAHMUD, I.; CHO, Y. Adaptive hello interval in fanet routing protocols for green uavs. **IEEE Access**, v. 7, p. 63004–63015, 2019.

MBARUSHIMANA, C.; SHAHRABI, A. Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks. In: **21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)**. [S.l.: s.n.], 2007. v. 2, p. 679–684.

MEMETI, S. et al. A review of machine learning and meta-heuristic methods for scheduling parallel computing systems. In: **Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications**. [S.l.: s.n.], 2018. p. 1–6.

MITCHELL, T. M. Does machine learning really work? **AI magazine**, v. 18, n. 3, p. 11–11, 1997.

MÖLLER, S.; RAAKE, A. **Quality of experience: advanced concepts, applications and methods**. [S.l.]: Springer, 2014.

NAWROCKI, P. et al. Adaptive context-aware energy optimization for services on mobile devices with use of machine learning considering security aspects. In: IEEE. **2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)**. [S.l.], 2020. p. 708–717.

PESHKIN, L.; SAVOVA, V. Reinforcement learning for adaptive routing. In: **Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)**. [S.l.: s.n.], 2002. v. 2, p. 1825–1830 vol.2.

POPA, D. et al. Deep learning model for home automation and energy reduction in a smart home environment platform. **Neural Computing and Applications**, Springer, v. 31, n. 5, p. 1317–1337, 2019.

RAMASUBRAMANIAN, V.; HAAS, Z. J.; SIRER, E. G. Sharp: A hybrid adaptive routing protocol for mobile ad hoc networks. In: **Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing**. [S.l.: s.n.], 2003. (MobiHoc '03), p. 303–314. ISBN 1-58113-684-6.

RECOMMENDATION, E. 800: Definitions of terms related to quality of service. **International Telecommunication Union's Telecommunication Standardization Sector (ITU-T) Std**, 2008.

RODRÍGUEZ, D. Z.; MÖLLER, S. Speech quality parametric model that considers wireless network characteristics. In: **2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)**. [S.l.: s.n.], 2019. p. 1–6. ISSN 2372-7179.

RODRÍGUEZ, D. Z. et al. Speech quality assessment in wireless communications with mimo systems using a parametric model. **IEEE Access**, v. 7, p. 35719–35730, 2019. ISSN 2169-3536.

ROSA, R. L.; Rodriguez, D. Z.; Bressan, G. Sentimeter-br: A social web analysis tool to discover consumers' sentiment. In: **2013 IEEE 14th International Conference on Mobile Data Management**. [S.l.: s.n.], 2013. v. 2, p. 122–124.

ROSA, R. L. et al. A knowledge-based recommendation system that includes sentiment analysis and deep learning. **IEEE Transactions on Industrial Informatics**, IEEE, v. 15, n. 4, p. 2124–2135, 2018.

SALAMI, R. et al. The adaptive multi-rate wideband codec: history and performance. In: **Speech Coding, 2002, IEEE Workshop Proceedings.** [S.l.: s.n.], 2002. p. 144–146.

SANCHEZ-IBORRA, R.; CANO, M.-D.; GARCIA-HARO, J. Performance evaluation of batman routing protocol for voip services: a qoe perspective. **IEEE Transactions on Wireless Communications**, IEEE, v. 13, n. 9, p. 4947–4958, 2014.

SHAIKH, J.; FIEDLER, M.; COLLANGE, D. Quality of experience from user and network perspectives. **Annales des Telecommunications/Annals of Telecommunications**, v. 65, n. 1-2, p. 47–57, 2010.

SHIN, C.; LEE, M. Swarm-intelligence-centric routing algorithm for wireless sensor networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 20, n. 18, p. 5164, 2020.

SUTTON, R.; BARTO, A. **Introduction to Reinforcement Learning. Cambridge, MA**. [S.l.]: USA: MIT Press, 1998.

TANENBAUM, A. **S. Redes de Computadores. São Paulo: Ed**. [S.l.]: Campus, 2003.

TANG, F. et al. On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control. **IEEE Wireless Communications**, v. 25, n. 1, p. 154–160, February 2018.

TANNENBAUM, A.; WETHERALL, D. **Computer Networks,(5-th edition)**. [S.l.]: Prentice-Hall, 2010.

TESAURO, G. Extending q-learning to general adaptive multi-agent systems. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2004. p. 871–878.

THAKKAR, P.; SHETE, P. Ah-aodv: adaptive hello messaging based aodv routing protocol. **International Journal of Computer Applications**, Foundation of Computer Science, v. 124, n. 17, 2015.

THEODORIDIS, S. **Koutroumbas K (2008) Pattern recognition**. [S.l.]: Academic Press, MA, USA, 2008.

TIRUMALA, A. et al. iperf: Tcp/udp bandwidth measurement tool. In: . [S.l.: s.n.], 2005.

VOIP-INFO.ORG. **What is voip**. 2019. Acesso em: 10 de Setembro de 2020. Disponível em: <https://www.voip-info.org/what-is-voip/>.

WANG, S.; SHIN, Y. Efficient routing protocol based on reinforcement learning for magnetic induction underwater sensor networks. **IEEE Access**, IEEE, v. 7, p. 82027–82037, 2019.

WANG, Y.; Lin, X. User-provided networking for qoe provisioning in mobile networks. **IEEE Wireless Communications**, v. 22, n. 4, p. 26–33, August 2015. ISSN 1558-0687.

WU, J. et al. Rsu-assisted traffic-aware routing based on reinforcement learning for urban vanets. **IEEE Access**, v. 8, p. 5733–5748, 2020.

YANG, J. et al. A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 19, n. 4, p. 970, 2019.