INSTITUTO FEDERAL DE SANTA CATARINA

CAMILLA BARRETO DE SOUSA

Um Agente baseado em Aprendizado por Reforço para Ajuste de Parâmetros de um MAC CSMA/CA

São José - SC

Dezembro/2025

UM AGENTE BASEADO EM APRENDIZADO POR REFORÇO PARA AJUSTE DE PARÂMETROS DE UM MAC CSMA/CA

Monografia apresentada ao Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheira de Telecomunicações.

Orientador: Prof. Marcelo Maia Sobral, Dr.

São José - SC

Dezembro/2025

CAMILLA BARRETO DE SOUSA

UM AGENTE BASEADO EM APRENDIZADO POR REFORÇO PARA AJUSTE DE PARÂMETROS DE UM MAC CSMA/CA

Este trabalho foi julgado adequado para obtenção do título de Engenheira de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 23 de Dezembro de 2025:

gov.br Documento assinado digitalmente
MARCELO MAIA SOBRAL
Data: 11/03/2026 16:02:02-0300
Verifique em https://validar.iti.gov.br

Prof. Marcelo Maia Sobral, Dr.
Orientador
Instituto Federal de Santa Catarina

gov.br Documento assinado digitalmente
ODILSON TADEU VALLE
Data: 11/03/2026 16:39:36-0300
Verifique em https://validar.iti.gov.br

Professor Odilson Tadeu Valle, Dr.
Instituto Federal de Santa Catarina

gov.br Documento assinado digitalmente
ARLIONES STEVERT HOELLER JUNIOR
Data: 11/03/2026 16:15:27-0300
Verifique em https://validar.iti.gov.br

Professor Arliones Stevert Hoeller Junior, Dr.
Instituto Federal de Santa Catarina

Este trabalho é dedicado à minha família e amigos,
que me apoiaram e incentivaram desde o início dessa jornada.

A inteligência é a capacidade de se adaptar à mudança.
Stephen Hawking

RESUMO

A tecnologia IEEE 802.11, popularmente conhecida por Wi-Fi, é uma tecnologia fundamental para redes locais sem-fio, proporcionando aceesso ubíquo a Internet. Todavia, aparelhos conectados a uma rede sem-fio precisam dividir o mesmo meio de comunicação para transmitir quadros de dados, o que ocasionalmente causa disputa pelo canal e possivelmente colisões de quadros. O protocolo MAC CSMA/CA usado no Wi-Fi impõe tempos de espera aleatórios antes da transmissão de quadros, para reduzir a chance de sobreposição de transmissões de diferentes dispositivos. No entanto, as durações desses tempos aleatórios podem não resultar num bom desempenho em dado cenário, onde uma certa quantidade de dispositivos disputam o canal para suas transmissões. A proposta desse trabalho é desenvolver um agente que ajuste dinamicamente os parâmetros do protocolo MAC, tais como janela de disputa e AIFS, de acordo com o estado do canal, buscando o melhor desempenho em diferentes cenários. Devido à complexidade da tarefa, uma técnica de aprendizado por reforço será utilizada, capacitando o agente de realizar sua função com a experiência de resultados anteriores.

Palavras-chave: IEEE 802.11. CSMA/CA. Aprendizado por Reforço.

ABSTRACT

The IEEE 802.11 standard, commercially known as Wi-Fi, is the fundamental technology for wireless local area networks, providing ubiquitous Internet access. However, devices connected to a wireless network must share the same communication medium to transmit data frames, which leads to channel contention and potential frame collisions. The CSMA/CA MAC protocol used in Wi-Fi imposes random waiting times before frame transmission in order to reduce the likelihood of overlapping transmissions from different devices. Nonetheless, the duration of these random backoff intervals may not provide optimal performance in a given scenario, where a certain number of devices compete for channel access. The objective of this work is to develop an agent capable of dynamically adjusting MAC parameters, such as the contention window and AIFS, according to channel conditions, aiming to achieve improved performance across different scenarios. Due to the complexity of this task, a reinforcement learning technique will be employed, enabling the agent to perform its function based on experience gained from previous outcomes.

Keywords: IEEE 802.11. CSMA/CA. Reinforcement Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

A Internet é uma rede global de comunicação de dados que conecta bilhões de dispositivos como computadores, *laptops*, *smartphones*, televisores e automóveis. Muitas pessoas estão conectadas quase integralmente a essa rede, utilizando seus recursos no trabalho, para se comunicar ou simplesmente por lazer. Em 2021, segundo a ITU (2021), cerca de 4,9 bilhões de pessoas se conectaram à Internet. Existem diversas tecnologias que possibilitam que aparelhos eletrônicos se conectem à Internet, o popular Wi-Fi é uma delas.

A tecnologia IEEE 802.11 (IEEE, 2016), conhecida como Wi-Fi, proporciona conexões sem-fio em redes locais. O diferencial dessa tecnologia em comparação com as outras é que ela dispensa o uso de cabos, facilitando a conexão dos aparelhos na rede. Como o canal de comunicação sem-fio é compartilhado pelos aparelhos conectados nessas redes, ocasionalmente ocorrem interferências entre as transmissões, causando erros de transmissão. Para minimizar esses efeitos, os dispositivos Wi-Fi disputam o acesso ao meio utilizando o protocolo *Media Access Control* (MAC) do tipo *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA).

Um aspecto importante do CSMA/CA é reduzir a probabilidade de acontecerem transmissões simultâneas em um mesmo canal de comunicação. Esse protocolo MAC impõe tempos de espera aleatórios antes de iniciar uma transmissão, chamados de *backoff*, em situações em que há chance de dois ou mais dispositivos estarem aguardando o meio ficar ocioso para iniciarem suas transmissões. A duração do *backoff* é sorteada dentro de um limite dado pela janela de disputa (*contention window*), a qual é dobrada a cada tentativa de transmissão de um mesmo quadro, havendo valores mínimos e máximos da janela de disputa definidos pelo padrão IEEE 802.11. Além da janela de disputa, o protocolo também aplica espaçamentos entre quadros, chamados de *Inter-Frame Spaces* (IFS), os quais são períodos mínimos em que as estações precisam aguardar antes da transmissão. Os espaçamentos entre quadros possuem valores distintos de tempo, dependendo da finalidade, sendo associados a diferentes prioridades de transmissão (GAST, 2005).

É importante ressaltar que a configuração dos parâmetros da janela de disputa deve proporcionar um acesso justo para as estações, resultando em um melhor desempenho do canal. Porém, como a configuração dos parâmetros é fixa no protocolo, esse desempenho é afetado pela quantidade de estações que estão transmitindo. O autor Garg (2007) explica que quanto maior o número de estações transmitindo, e menor o valor de *backoff*, maior será a ocorrência de colisões. Em contrapartida, quando há poucas estações transmitindo e os valores de *backoff* são grandes, ocorrem atrasos e pouco aproveitamento do canal.

A proposta deste trabalho é desenvolver um agente que ajuste dinamicamente os parâmetros da janela de disputa conforme o estado do canal muda visando o melhor desempenho em todos os diferentes cenários. Além disso, propõem-se ajustar dinamicamente o *Arbitration Inter-Frame Space* (AIFS), que é um dos espaçamentos entre quadros definidos no protocolo. Como o estado do canal depende de diversos fatores, tornando estes ajustes uma tarefa muito complexa, a proposta é desenvolver um agente que altera os parâmetros do CSMA/CA usando uma técnica de Aprendizado por Reforço (AR), que é uma das áreas de estudo de Aprendizado de Máquina. Ressalta-se que a técnica adotada prioriza algoritmos de baixa complexidade computacional, de modo a viabilizar sua implementação em dispositivos embarcados com recursos limitados. Com isso, espera-se que o agente aprenda a configurar os parâmetros do protocolo com base na experiência obtida de resultados anteriores, sem comprometer o desempenho global do sistema. Além do desenvolvimento do agente e da biblioteca de aprendizado por reforço, são realizados experimentos para validar o funcionamento da biblioteca implementada e demonstrar a operação do agente no ajuste dinâmico dos parâmetros do protocolo.

1.1 Objetivo Geral

Desenvolver e validar um agente, baseado em AR, para ajustar dinamicamente os parâmetros de janela de disputa e o espaçamento entre quadros AIFS do protocolo MAC CSMA/CA, visando melhor eficiência do uso do canal. O agente será desenvolvido em linguagem C, de forma a possibilitar sua portabilidade e aplicação em dispositivos embarcados com recursos computacionais limitados.

1.2 Objetivos Específicos

Para alcançar o objetivo central proposto no trabalho será necessário atingir os seguintes objetivos específicos:

1. Desenvolver uma biblioteca parametrizável que implemente os algoritmos de AR escolhidos;
2. Desenvolver um software capaz de implementar um agente para ajuste de parâmetros, utilizando a biblioteca previamente implementada;
3. Modelar o comportamento do agente de forma que o software possa ser utilizado em experimentos futuros para análise de desempenho.
4. Realizar experimentos para validar o funcionamento da biblioteca e demonstrar a operação do agente no ajuste dinâmico dos parâmetros do protocolo.

1.3 Organização do texto

No Capítulo 1 foi contextualizado o tema deste trabalho, apontado o problema encontrado no cenário e a proposta de solução. No Capítulo 2 é apresentada a fundamentação teórica, onde é feito uma revisão dos assuntos que abrangem a proposta do trabalho. No Capítulo 3 é descrito o desenvolvimento da biblioteca de aprendizado por reforço e do agente proposto para ajuste dinâmico dos parâmetros do protocolo. No Capítulo 4 são apresentados os experimentos realizados para validação da biblioteca e para demonstração do funcionamento do agente no ajuste dos parâmetros do protocolo. Por fim, no Capítulo 5, são revisitados os principais resultados obtidos no trabalho e apresentadas as propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados assuntos teóricos que fundamentam o trabalho proposto. Na seção 2.1 são descritos os principais aspectos da tecnologia IEEE 802.11 que são de interesse desse trabalho. A seção 2.2 descreve o funcionamento do protocolo MAC CSMA/CA utilizado no padrão IEEE 802.11, em que se propõe adicionar o agente. Na seção 2.3 serão abordados conceitos de Aprendizado de Máquina, dando destaque aos conceitos de Aprendizado por Reforço e aos Métodos de Diferença Temporal (DT).

2.1 IEEE 802.11

A tecnologia IEEE 802.11 corresponde a um conjunto de padrões definidos pela Institute of Electrical and Eletronics Engineers (IEEE) para conexões sem-fio de dispositivos em redes locais por meio de ondas de rádio. Existem diversas emendas que marcaram a evolução da tecnologia, trazendo melhorias em termos de taxa de dados, alcance do sinal em ambientes internos e externos além de conexões mais seguras (IEEE Standards Association, s.d.).

O Wi-Fi é amplamente utilizado em diferentes lugares, como em ambientes residenciais, corporativos, comerciais, educacionais, pois dispensa a necessidade de cabeamento físico e permite a mobilidade dos dispositivos conectados. Devido a facilidade de conexão e uso, cada vez mais dispositivos se conectam a essas redes, resultando em mais aparelhos compartilhando o mesmo meio de transmissão.

Uma das principais diferenças entre redes sem-fio e redes cabeadas é que o meio de transmissão nas Wireless Local Area Network (WLANs) não possui limites físicos definidos e está sujeito a interferências externas, especialmente em áreas onde redes se sobrepõem. Além disso, a topologia das redes sem-fio é dinâmica, o que pode resultar em Stations (STAs) “ocultas”, que não conseguem detectar a transmissão de outras STAs. Todas essas características e as dificuldades inerentes a um meio de transmissão compartilhado exigem protocolos de controle de acesso ao meio. No caso do Wi-Fi, a norma IEEE 802.11 define que o protocolo MAC para esse tipo de rede é o CSMA/CA (IEEE, 2021).

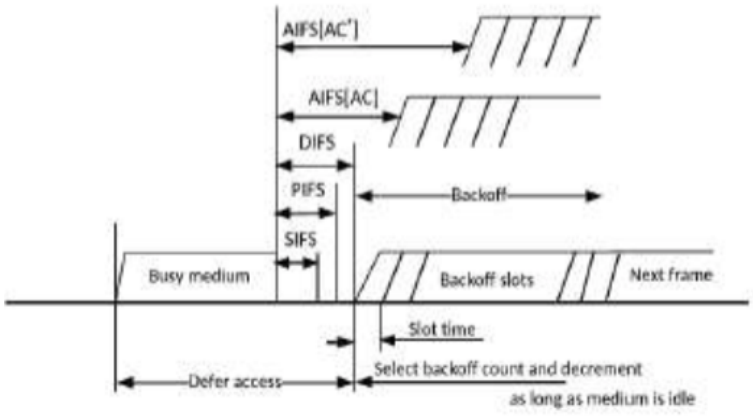
O desempenho das redes IEEE 802.11 é intimamente ligado à eficiência do protocolo MAC, que é responsável por controlar o compartilhamento do canal entre as STAs. A sobrecarga do canal, resultante da crescente quantidade de dispositivos conectados, pode levar a aumentos no tempo de latência, número de colisões e diminuição do throughput, impactando negativamente a qualidade da comunicação. O desempenho do protocolo

também é afetado pela variação da densidade de dispositivos que compartilham o canal.

2.2 MAC CSMA/CA

Em uma rede sem-fio como no padrão IEEE 802.11, o canal de comunicação é compartilhado entre as estações sem-fio. Um quadro transmitido por uma estação é recebido por todas as demais estações que estejam no alcance do sinal. Eventualmente, por conta do compartilhamento do canal, ocorrem colisões entre quadros cujas transmissões se sobreponham. Esses quadros, transmitidos por duas ou mais estações, se interferem no canal, de forma que os receptores não conseguem decodificá-los e recuperar as informações. Para evitar que ocorram colisões, ou reduzir a probabilidade que aconteçam, adota-se nesse tipo de rede um protocolo de controle de acesso ao meio, ou protocolo MAC. Existem diversos tipos de protocolo MAC com diferentes técnicas de controle, os quais podem ser classificados em três tipos: partição de canal, revezamento ou acesso aleatório. As redes sem-fio IEEE 802.11 usam o protocolo MAC CSMA/CA, classificado como um protocolo de acesso aleatório, que busca evitar a ocorrência de colisões entre quadros (KUROSE; ROSS, 2014).

Figura 1 – Protocolo CSMA/CA nas redes IEEE 802.11.



Fonte: (IEEE, 2021)

O MAC CSMA/CA define que, quando uma estação deseja realizar uma transmissão, primeiramente deve aguardar que o canal esteja ocioso por um tempo de espera mínimo chamado de Espaçamento entre Quadros, ou *Inter-Frame Spaces* (IFS). O protocolo define alguns tipos de IFS, como é mostrado na Figura 1, os quais possuem valores distintos de tempo, dependendo da finalidade, sendo associados a diferentes prioridades de transmissão. Se, ao iniciar o procedimento de transmissão, o canal já estiver livre por pelo menos esse tempo IFS, então a transmissão pode iniciar imediatamente. Caso contrário, deve-se aguardar que esse tempo decorra e, em seguida, um tempo adicional, chamado de *backoff*, calculado como uma quantidade aleatória de *slots* (uma constante de tempo

definida no padrão IEEE 802.11). A quantidade de *slots* é sorteada dentro de uma faixa de valores entre 0 (*zero*) e o valor da janela de disputa, ou Contention Window (CW). O valor de *backoff* é decrementado a cada *slot*, e, quando chega a zero, a estação pode então realizar sua transmissão. Durante essa espera, o decremento do *backoff* é congelado sempre que for detectada uma transmissão no canal. (KLOIBER et al., 2012).

2.2.1 Parâmetros

O protocolo possui alguns parâmetros fixos que configuram alguns aspectos de funcionamento. Esses parâmetros são predefinidos no padrão IEEE 802.11, de forma a possibilitarem o funcionamento do MAC CSMA/CA em redes sem-fio com quantidades e densidades de estações arbitrárias, dentro de certos limites. Nessa subseção são apresentados os parâmetros de interesse neste trabalho.

2.2.1.1 Janela de Disputa

Como mencionado anteriormente, o valor de *backoff* define um tempo de espera antes de transmitir, dado por uma quantidade aleatória de *slots* de tempo, gerada dentro de uma faixa de valores. Essa faixa tem como limite inferior o valor 0 (*zero*), e CW como limite superior. Inicialmente, o valor de CW é dado pelo parâmetro CW_{min}. A cada tentativa de transmissão de um mesmo quadro, o valor de CW é recalculado da seguinte forma:

$$CW_n = 1 + 2 \cdot CW_{n-1} \tag{2.1}$$

O valor máximo de CW é dado pelo parâmetro CW_{max}. Ambos parâmetros são predefinidos pelo padrão IEEE 802.11, com CW_{min} tendo valor 15, e CW_{max} tendo valor 1023.

O cálculo da janela de disputa faz com que, a cada tentativa de transmissão de um mesmo quadro, o tempo de *backoff* seja, em média, o dobro do tempo usado na tentativa anterior.

2.2.1.2 Espaçamento entre Quadros

No intuito de implementar uma forma de priorização estatística para os diferentes tipos de transmissões, o protocolo define alguns tempos de intervalos mínimos entre quadros, ou IFS. No caso de quadros comuns, seja de dados ou de gerenciamento, o tempo de intervalo entre quadros se chama Distributed Inter-Frame Space (DIFS), e tem duração de $34\mu s$. No caso de quadros Quality of Service (QoS), os quais são associados a classes de acesso para diferenciar suas prioridades de acesso ao meio, usa-se um espaçamento entre quadros denominado AIFS. Quanto mais prioritária uma classe de acesso, mais curto

seu intervalo AIFS. Os valores de AIFS para cada classe de acesso são predefinidos pelo padrão IEEE 802.11, de acordo com a equação 2.2:

$$AIFS[AC] = AIFSN[AC] * aSlotTime + aSIFSTime \tag{2.2}$$

... sendo que:

- **AC**: classe de acesso, que pode ser VO (voz), VI (vídeo), BE (melhor-esforço) ou BK (segundo plano)
- **AIFSN**: tabela com quantidades de *slots* para cada AC
- **aSlotTime**: duração de um *slot*, que é de $9\mu s$ no padrão IEEE 802.11
- **aSIFSTime**: duração do intervalo do tipo Short Inter-Frame Space (SIFS), que é o menor intervalo entre quadros possível, sendo usado somente antes da transmissão de quadros de controle (ex: Acknowledge (ACK) ou Clear-To-Send (CTS)).

2.3 Aprendizado de Máquina

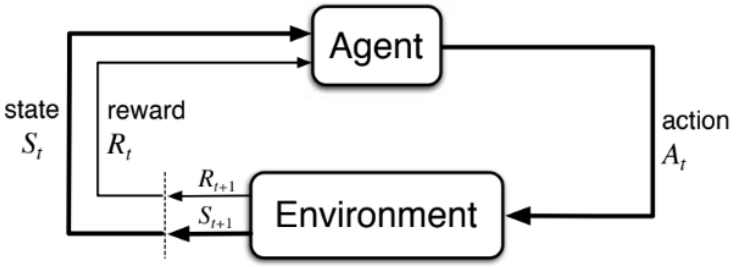
O Aprendizado de Máquina é definido pela IBM (2021) como uma "...tecnologia onde os computadores tem a capacidade de aprender de acordo com as respostas esperadas por meio associações de diferentes dados, os quais podem ser imagens, números e tudo que essa tecnologia possa identificar". Existem três abordagens de Aprendizado de Máquina: o Aprendizado Supervisionado, Aprendizado Não Supervisionado e o Aprendizado por Reforço. O Aprendizado Supervisionado consiste em treinar um algoritmo para tomar decisões corretas por meio de um conjunto de dados que contém os resultados verdadeiros sobre a tarefa de interesse. O Aprendizado Não Supervisionado busca encontrar padrões que são complexos de serem previstos dentro de uma base de dados que contém informações de um determinado problema. Por fim, o Aprendizado por Reforço tem por objetivo aprender a tomar boas decisões sobre uma determinada tarefa com base na experiência de escolhas anteriores.

2.3.1 Aprendizado por Reforço

O Aprendizado por Reforço é um dos paradigmas de Aprendizado de Máquina e este "usa a estrutura formal dos processos de decisão de Markov para definir a interação entre um agente de aprendizado e seu ambiente em termos de estados, ações e recompensas" (SUTTON; BARTO, 2018). Uma particularidade do AR em relação as outras abordagens de Aprendizado de Máquina é que não se faz necessário uma base de dados prévios para que o algoritmo realize uma tarefa. No diagrama da Figura 2 é possível visualizar a dinâmica

entre agente e ambiente. Esses dois blocos interagem periodicamente seguindo a sequência: i) o agente recebe o estado do ambiente (S_t) e toma uma ação (A_t), ii) ambiente reage e retorna ao agente o seu novo estado (S_{t+1}) junto com a recompensa (R_{t+1}).

Figura 2 – Interação agente-ambiente.



Fonte: (SUTTON; BARTO, 2018)

Um Processo de Decisão de Markov, ou Markov Decision Process (MDP), dispõe de um conjunto de ações e um conjunto de estados. As ações são as formas como o agente poderá agir nas tomadas de decisão. Os estados representam todas as configurações possíveis do ambiente durante o processo. A recompensa é um número inteiro que orienta o agente ao objetivo, indicando se a decisão tomada foi boa ou não. A tarefa do agente é maximizar as recompensas, ou seja, em cada estado deverá encontrar a ação que gera resultados com maior valor. O aprendizado ocorre à medida que o agente associa valores a cada ação ou estado, os quais informam o ganho potencial proporcionado por sua escolha. No caso de sistemas com número de estados tratáveis, algoritmos de aprendizado tabulares, tais como *SARSA* e *Q-Learning*, podem apresentar bons resultados com poucos recursos computacionais (SUTTON; BARTO, 2018).

2.3.1.1 Métodos de Diferenças Temporais

Os algoritmos *SARSA* e *Q-Learning* são classificados como métodos de DT, porque não precisam que a tarefa chegue ao fim para que o algoritmo atualize (aprenda), diferente de outros métodos de Aprendizado por Reforço. Os métodos de DT são úteis para resolução de problemas que precisam ser aprendidos *online*, ou seja, quando as tarefas são contínuas ou quando demora para chegar em um estado final.

2.3.1.1.1 SARSA

O algoritmo *SARSA* é utilizado para aprender uma política de ação a partir da interação agente-ambiente. O objetivo do algoritmo é estimar a função de ação, $Q(S, A)$, que representa o valor esperado ao executar a ação A no estado S e, a partir disso, seguir a politica atual.

A regra de atualização do valor da função $Q(S, A)$ no algoritmo *SARSA* é definido da seguinte maneira:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big] \tag{2.3}$$

... sendo que:

- $Q(S_t, A_t)$: é o valor estimado da ação escolhida no estado anterior.
- α: é a taxa de aprendizado
- R_{t+1}: é a recompensa imediata.
- γ: é o fator de desconto.
- $Q(S_{t+1}, A_{t+1})$: é o valor da próxima ação escolhida.

A atualização do valor da ação é realizado na transição para um estado não-terminal. Na Figura 3 é apresentada a forma geral do algoritmo *SARSA*.

Figura 3 – *SARSA*: sob a política de controle de DT.

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Fonte: (SUTTON; BARTO, 2018)

2.3.1.1.2 Q-Learning

O algoritmo *Q-Learning* é utilizado para aprender a função de valor independente da política que está sendo seguida. O objetivo do algoritmo é aprender uma política ótima, mesmo que o agente esteja explorando ou executando ações aleatórias.

A regra de atualização do valor da função $Q(S, A)$ no algoritmo *Q-Learning* é definido da seguinte maneira:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big] \tag{2.4}$$

. . . sendo que:

- $Q(S_t, A_t)$: é o valor estimado da ação escolhida no estado anterior.
- α: é a taxa de aprendizado
- R_{t+1}: é a recompensa imediata.
- γ: é o fator de desconto.
- $max_a Q(S_{t+1}, A_{t+1})$: é o valor máximo das ações no estado atual.

A atualização do valor da ação é realizado na transição para um estado não-terminal. Na Figura 4 é apresentada a forma geral do algoritmo *Q-Learning*.

Figura 4 – Q-Learning: fora da política de controle de DT.

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Fonte: (SUTTON; BARTO, 2018)

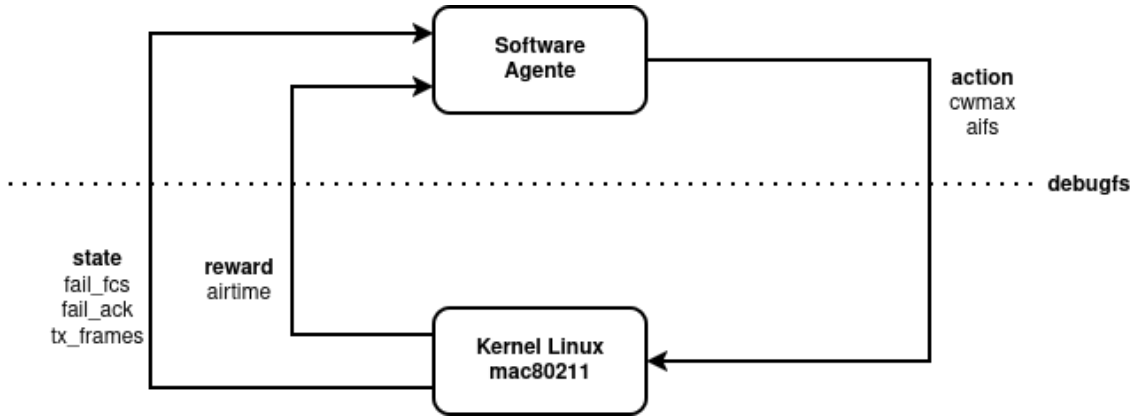
3 AGENTE PARA AJUSTE DE PARÂMETROS CSMA/CA

Nesse capítulo será apresentada uma visão geral do agente proposto, seguido de um detalhadamento da modelagem dos elementos do AR, tais como estado, ação e recompensa, também são apresentadas as implementações da biblioteca TD_algorithm e do software responsável pelo papel de agente do ambiente. O código-fonte desenvolvido nesse trabalho está disponível em (SOUSA, 2025).

3.1 Visão geral

O software-agente foi modelado com o objetivo de proporcionar um melhor aproveitamento do canal de comunicação em redes IEEE 802.11 através do ajuste dinâmico de parâmetros do protocolo CSMA/CA. A Figura 5 ilustra a interação entre o agente e o ambiente adotada neste trabalho, onde o agente tem a responsabilidade de ajustar dinamicamente os parâmetros CW_{max} e $AIFS$ com base nas informações obtidas e aprendidas relacionadas ao estado da rede WiFi.

Figura 5 – Diagrama de interação agente-ambiente modelado para o ajuste de parâmetros do MAC CSMA/CA.



Fonte: Adaptado de (SUTTON; BARTO, 2018)

A cada ciclo de interação, o agente observa o estado do ambiente composto por informações que caracterizam a ocupação do canal e o desempenho da comunicação. Com base nessas informações, o agente seleciona uma ação à ser tomada, que consiste na modificação de parâmetros do protocolo CSMA/CA. Após esse processo, o ambiente vai retornar ao agente uma recompensa, que reflete o impacto da ação tomada em termos de eficiência do uso do canal.

3.2 Modelagem

Para modelar o AR com o objetivo de resolver o problema de ajuste dinâmico dos parâmetros do CSMA/CA, é necessária a definição dos elementos: estado, ação e recompensa. Nessa seção é apresentada a modelagem proposta para cada um dos elementos.

3.2.1 Estado

O estado foi definido a partir da combinação das métricas FAIL_FCS, FAIL_ACK e TX_FRAMES. A relação entre essas métricas é apresentada pela equação a seguir, na qual a soma de FAIL_FCS e FAIL_ACK define a coordenada no eixo x e TX_FRAMES define a coordenada no eixo y:

$$(x, y) = (\text{FAIL\_FCS} + \text{FAIL\_ACK},\ \text{TX\_FRAMES}) \tag{3.1}$$

A métrica FAIL_FCS representa a somatória da quantidade de quadros recebidos com erro de FCS dentro do período de observação. A métrica FAIL_ACK representa a somatória da quantidade de quadros que falharam na transmissão por ausência de confirmação dentro do período de observação. O resultado da soma das métricas será mapeado para um nível entre 0 e 10. O mapeamento dos 11 níveis foi definido assim:

$$\text{Nível X(FAIL\_FCS + FAIL\_ACK)} = \begin{cases} 0, & \text{FAIL\_FCS + FAIL\_ACK} = 0 \\ 1, & \text{FAIL\_FCS + FAIL\_ACK} = 1 \\ 2, & \text{FAIL\_FCS + FAIL\_ACK} = 2 \\ 3, & \text{FAIL\_FCS + FAIL\_ACK} = 3 \\ 4, & \text{FAIL\_FCS + FAIL\_ACK} = 4 \\ 5, & \text{FAIL\_FCS + FAIL\_ACK} = 5 \\ 6, & \text{FAIL\_FCS + FAIL\_ACK} = 6 \\ 7, & \text{FAIL\_FCS + FAIL\_ACK} = 7 \\ 8, & \text{FAIL\_FCS + FAIL\_ACK} = 8 \\ 9, & \text{FAIL\_FCS + FAIL\_ACK} = 9 \\ 10, & \text{FAIL\_FCS + FAIL\_ACK} \geq 10 \end{cases} \tag{3.2}$$

A métrica TX_FRAMES representa a somatória da quantidade de quadros transmitidos com sucesso dentro do período de observação. O valor da somatória será mapeado para um nível entre 0 e 10. O mapeamento dos 11 níveis foi definido assim:

$$\text{Nível Y(TX\_FRAMES)} = \begin{cases} 0, & 0 \leq \text{TX\_FRAMES} < 100 \\ 1, & 100 \leq \text{TX\_FRAMES} < 200 \\ 2, & 200 \leq \text{TX\_FRAMES} < 300 \\ 3, & 300 \leq \text{TX\_FRAMES} < 400 \\ 4, & 400 \leq \text{TX\_FRAMES} < 500 \\ 5, & 500 \leq \text{TX\_FRAMES} < 600 \\ 6, & 600 \leq \text{TX\_FRAMES} < 700 \\ 7, & 700 \leq \text{TX\_FRAMES} < 800 \\ 8, & 800 \leq \text{TX\_FRAMES} < 900 \\ 9, & 900 \leq \text{TX\_FRAMES} < 1000 \\ 10, & \text{TX\_FRAMES} \geq 1000 \end{cases} \tag{3.3}$$

A combinação dos níveis de FAIL_FCS + FAIL_ACK e TX_FRAMES resulta em um total de 121 arranjos de estados. Vale ressaltar que os limiares adotados em ambos os eixos foram definidos empiricamente, buscando um compromisso entre a redução do custo computacional e a preservação da granularidade do estado.

3.2.2 Ação

A ação foi definida como a manipulação dos parâmetros CW_MAX e AIFS, os quais podem assumir qualquer valor pertencente aos respectivos conjuntos:

$$\text{CW\_MAX} \in \{31, 63, 127, 255, 511, 1023\} \tag{3.4}$$

$$\text{AIFS} \in \{1, 2, 3, \ldots, 31\} \tag{3.5}$$

A combinação dos conjuntos de valores de CW_MAX e AIFS resulta em um total de 186 arranjos de ações. Destaca-se que os valores considerados para os parâmetros CW_MAX e AIFS foram limitados de modo a respeitar as restrições e recomendações estabelecidas pelo padrão IEEE 802.11.

3.2.3 Recompensa

A recompensa foi definida a partir da métrica AIRTIME, que representa o tempo médio necessário para a transmissão dos quadros dentro do período observado. O valor medido será multiplicado por -1 para que a recompensa indique ao agente que valores próximos de 0 representam melhor aproveitamento do canal, enquanto valores distantes de 0 representam o contrário. Portanto, a recompensa ficou definida como:

$$\text{RECOMPENSA} = -(\text{AIRTIME}) \tag{3.6}$$

3.3 Biblioteca

A biblioteca TD_algorithm implementa, em suas funções internas, os algoritmos SARSA e Q-Learning, ambos utilizando a política ϵ-greedy para possibilitar que o agente se adapte a mudanças no ambiente. São disponibilizas duas funções externas que permitem a interação do módulo com o agente. Também são disponibilizas estruturas responsáveis por modelar e armazenar os recursos necessários para o funcionamento dos algoritmos. A biblioteca exige que o módulo externo implemente três funções, chamadas de funções compute, que são utilizadas pela biblioteca.

3.3.1 Funções externas

Nesta subseção, serão apresentadas as funções externas que permitem ao módulo interagir com o agente, incluindo suas assinaturas e uma descrição de seus parâmetros e valores de retorno.

3.3.1.1 TDA_initialize

A função `TDA_initialize` tem a tarefa de inicializar o agente, realizar o primeiro ciclo de execução do algoritmo e retornar a ação escolhida por ele. A função recebe como parâmetro a estrutura `TDA_config_t` e o retorno da função é um ponteiro para uma estrutura que armazena os parâmetros da ação. A definição dessa estrutura retornada é de responsabilidade do módulo. Abaixo é apresentada a assinatura da função:

```
1 void * TDA_initialize(TDA_config_t *conf);
```

3.3.1.2 TDA_next

A função `TDA_next` tem a tarefa de realizar os próximos ciclos do algoritmo. A função recebe como parâmetro a estrutura `TDA_config_t` e o retorno da função é um ponteiro para uma estrutura que armazena os parâmetros da ação. A definição dessa estrutura retornada é de responsabilidade do módulo. Abaixo é apresentada a assinatura da função:

```
1 void * TDA_next(TDA_config_t *conf);
```

3.3.2 Estruturas

Nessa subseção serão apresentadas as estruturas que modelam e armazenam os recursos necessários para o funcionamento do algoritmo, incluindo suas assinaturas e uma descrição de cada um dos campos.

3.3.2.1 TDA_action_t

A estrutura `TDA_action_t` representa uma ação possível em um estado.

```
typedef struct {
    int calls;
    float value;
    void *paremeters;
} TDA_action_t;
```

- **calls**: quantidade de vezes que a ação foi escolhida.
- **value**: valor estimado da ação.
- **parameters**: ponteiro para a estrutura que armazena os parâmetros que definem a ação. A definição dessa estrutura fica a cargo da aplicação.

3.3.2.2 TDA_state_info_t

A estrutura `TDA_state_info_t` representa um estado do ambiente.

```
typedef struct {
    void * descr;
    int calls;
    int actions_len;
    TDA_action_t *actions;
} TDA_state_info_t;
```

- **descr**: ponteiro para a estrutura que descreve o estado. A definição dessa estrutura fica a cargo da aplicação.
- **calls**: quantidade de vezes que o estado foi visitado.
- **actions_len**: número de ações disponíveis no estado.
- **actions**: vetor de ações possíveis no estado.

3.3.2.3 TDA_status_t

A estrutura `TDA_status_t` define o status atual que o algoritmo se encontra.

```
typedef struct {
    TDA_state_info_t *current_state;
    int current_action;
} TDA_status_t;
```

- **current_state**: ponteiro que indica o estado atual.
- **current_action**: indice da ação que foi escolhida.

3.3.2.4 TDA_config_t

A estrutura `TDA_config_t` armazena as configurações do algoritmo, o contexto ao qual o algoritmo se encontra e os recursos necessários para o seu funcionamento.

```
typedef struct {
    TDA_status_t status;
    HashTable *ht_states;
    int algorithm;
    float alpha;
    float gama;
    float explore;
    int seed;
    void *private;
    int (*compute_reward)(void * priv);
    void (*compute_actions)(TDA_state_info_t * state, void * priv);
    int (*compute_state)(void * priv);
} TDA_config_t;
```

- **status**: status corrente.
- **ht_states**: tabela de estados.
- **algorithm**: algoritmo (SARSA ou QLEARNING).
- **alpha**: taxa de aprendizagem ($0 < \alpha \leq 1$).
- **gama**: fator de desconto ($0 \leq \gamma \leq 1$).
- **explore**: taxa de exploração ($0 \leq \varepsilon \leq 1$).
- **seed**: semente RNG (*Random Number Generator seed*).
- **private**: ponteiro para uma estrutura que armazena informações do ambiente. A definição da estrutura fica a cargo da aplicação.
- **compute_reward**: ponteiro para a função `compute_reward`. A implementação da função fica a carga da aplicação.
- **compute_actions**: ponteiro para a função `compute_actions`. A implementação da função fica a carga da aplicação.
- **compute_state**: ponteiro para a função `compute_state`. A implementação da função fica a carga da aplicação.

3.3.3 Funções Compute

Nessa subseção serão apresentadas as funções compute. Essas funções são utilizadas pela biblioteca para calcular o estado em que o ambiente se encontra, as ações possíveis do estado atual e a recompensa da última ação. A implementação das funções são de responsabilidade da aplicação, pois dependem da modelagem do problema.

3.3.3.1 Função compute_state

A função `compute_state` é responsável por definir em qual estado o ambiente se encontra. O parâmetro dessa função é um ponteiro para a estrutura definida pela aplicação (mesma estrutura apontada por `TDA_config_t::private`). O retorno da função deve ser um inteiro que represente o estado, servindo como um identificador único. Abaixo é apresentada a assinatura da função:

```
1 int (*compute_state)(void * priv);
```

3.3.3.2 Função compute_actions

A função `compute_actions` é responsável por definir as ações possíveis em um determinado estado. Os parâmetros dessa função são um ponteiro para o estado atual do ambiente e um ponteiro para a estrutura definida pela aplicação (mesma estrutura apontada por `TDA_config_t::private`). A função não tem valor de retorno. Abaixo é apresentada a assinatura da função:

```
1 void (*compute_actions)(TDA_state_info_t * state, void * priv);
```

3.3.3.3 Função compute_reward

A função `compute_reward` é responsável por calcular a recompensa da última ação tomada. O parâmetro dessa função é um ponteiro para a estrutura definida pela aplicação (mesma estrutura apontada por *TDA_config_t::private*). O retorno da função deve ser um inteiro que represente a recompensa da última ação. Abaixo é apresentada a assinatura da função:

```
1 int (*compute_reward)(void * priv);
```

3.4 Software

O software foi implementado utilizando a biblioteca TD_algorithm como agente adaptativo dos parâmetros do MAC CSMA/CA. Para fazer consultas as métricas do

ambiente e realizar modificações dos parâmetros do protocolo MAC CSMA/CA, o software realiza operações de leitura e escrita em arquivos debugfs.

3.4.1 Interface entre o software e o kernel

A interface do software com o kernel Linux é por meio de operações de leitura e escrita em arquivos disponibilizados pelo sistema de arquivos debugfs. A coleta das métricas do ambiente de rede, bem como as modificações dos parâmetros do protocolo CSMA/CA, são implementadas diretamente no código-fonte do kernel Linux, mais especificamente no subsistema mac80211. A árvore de arquivos exposta via debugfs foi organizada conforme descrito a seguir:

```
/sys/kernel/debug/tda/
├── ack_fail
├── be/
│   ├── action
│   └── stats
├── bk/
│   ├── action
│   └── stats
├── fcs_fail
├── vi/
│   ├── action
│   └── stats
└── vo/
    ├── action
    └── stats
```

... onde:

- **ack_fail**: arquivo para leitura da métrica ACK_FAIL.
- **fcs_fail**: arquivo para leitura da métrica FCS_FAIL.
- **action**: são arquivos para escrita das ações AIFS e CW_MAX nas diferentes categorias de tráfego.
- **stats**: são arquivos para leitura das métricas TX_FRAMES e AIRTIME nas diferentes categorias de tráfego.

3.4.2 Configurações da biblioteca

Nessa subseção serão apresentadas as configurações setadas na biblioteca TD_algorithm, além das implementações das funções compute e estruturas dependentes da aplicação.

3.4.2.1 TDA_config_t

Na estrutura TDA_config_t foram setados os seguintes valores:

- **algorithm**: `SARSA`
- **alpha**: `0.5`
- **gama**: `1`
- **explore**: `0.1`
- **seed**: `time(NULL)`
- **private**: ponteiro para a struct `mac_factors_t`
- **compute_reward**: ponteiro para a função `compute_reward`
- **compute_actions**: ponteiro para a função `compute_actions`
- **compute_state**: ponteiro para a função `compute_state`

3.4.2.2 Implementação da struct mac_factors_t

Na aplicação foi implementada a estrutura `mac_factors_t` para armazenar os dados coletados do ambiente que alimentarão o agente (estado e recompensa) e dados que serão injetados no ambiente (ação). Abaixo são apresentadas as assinaturas da estrutura principal, estruturas auxiliares e também o significado de cada um dos respectivos campos:

```
1 typedef struct {
2     state_factors_t state;
3     action_factors_t action;
4 } mac_factors_t;
```

- **state**: dados coletados do ambiente.
- **action**: dados escolhidos para serem injetados no ambiente.

3.4.2.3 Implementação da struct state_factors_t

Na aplicação foi implementada a estrutura `state_factors_t` para armazenar os dados coletados do ambiente que alimentarão o agente (estado e recompensa). Abaixo são apresentados os campos que compõem a estrutura e seus significados:

```
typedef struct {
    int tx_frames;
    int fail_fcs;
    int fail_ack;
    int airtime;
} state_factors_t;
```

- **tx_frames**: número de quadros transmitidos com sucesso no último intervalo.
- **fail_fcs**: número de quadros recebidos com erro FCS no último intervalo.
- **fail_ack**: número de quadros que falharam na transmissão por falta de confirmação no último intervalo.
- **airtime**: tempo médio de transmissão de quadros no último intervalo.

3.4.2.4 Implementação da struct action_factors_t

Na aplicação foi implementada a estrutura `action_factors_t` para armazenar os dados que serão injetados no ambiente (ação). Abaixo são apresentados os campos que compõem a estrutura e seus significados:

```
typedef struct {
    int aifs;
    int cwmax;
} action_factors_t;
```

- **aifs**: valor de aifs escolhido pelo algoritmo no último intervalo.
- **cwmax**: valor de cwmax escolhido pelo algoritmo no último intervalo.

3.4.2.5 Implementação da função compute_actions

Na implementação da função `compute_actions` é necessário definir quais ações serão possíveis no estado passado como parâmetro. Foi definido que todos os arranjos de ações seriam possíveis independente do estado. Portanto, a implementação da função consistiu em declarar as 186 ações no vetor `TDA_state_info_t::actions` do estado requisitado.

3.4.2.6 Implementação da função compute_state

Na implementação da função `compute_state` é necessário retornar um inteiro que identifique o estado em que o ambiente se encontra com base nos dados coletados no

último intervalo. Portanto, a implementação da função consistiu em encontrar em quais níveis os fatores FAIL_FC + FAIL_ACK e TX_FRAMES se encontravam. Além disso foi necessário implementar uma lógica para que cada estado fosse representado por um identificador único. Abaixo é apresentada a lógica para gerar o identificador:

$$\text{STATE} = 100 \times \text{Nível\_X}(\text{FAIL\_FCS} + \text{FAIL\_ACK}) + \text{Nível\_Y}(\text{TX\_FRAMES}) \quad (3.7)$$

3.4.2.7 Implementação da função compute_reward

Na implementação da função `compute_reward` é necessário retornar um inteiro que represente a recompensa da última ação escolhida. A função foi implementada para buscar o valor de AIRTIME coletado no último intervalo e retornar o seu valor negativo:

$$\text{REWARD} = -1 \times \text{AIRTIME} \quad (3.8)$$

4 RESULTADOS

Nesse capítulo serão apresentadas as validações da biblioteca TD_algorithm e do agente para o ajuste de parâmetros CSMA/CA.

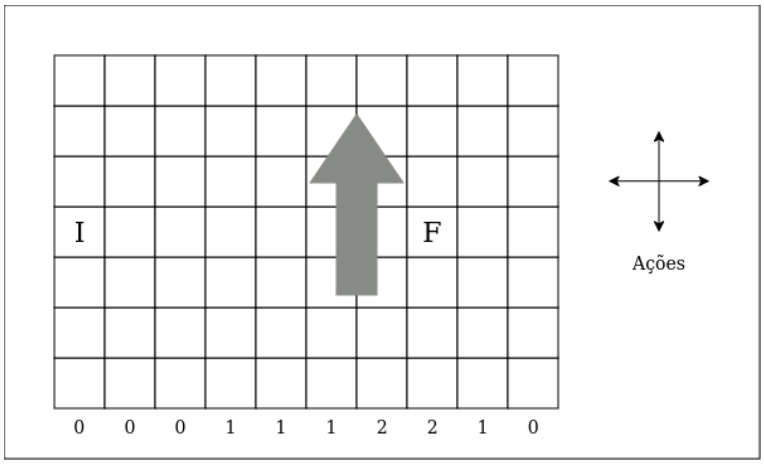
4.1 Validação da Biblioteca

Para validar a biblioteca TD_algorithm, foram realizados 2 experimentos reproduzindo exemplos descritos no Capítulo 6: *Temporal-Difference Learning* do livro *Reinforcement Learning: An Introduction*, de (SUTTON; BARTO, 2018).

4.1.1 Experimento 1

O primeiro experimento reproduz o Exemplo 6.5 chamado *Windy Gridworld* que demonstra a aplicação do algoritmo Sarsa. O ambiente consiste em um conjunto finito de estados organizados em forma de grade. Dentre esses estados, há um estado inicial (I), a partir do qual o agente começa cada episódio, e um estado final (F), que encerra o episódio quando alcançado. O agente pode se mover para os estados localizados acima, abaixo, à direita ou à esquerda. No entanto, os movimentos realizados em algumas regiões da grade são influenciados por um "vento"que desloca o agente para uma quantidade de estados para cima. A quantidade de estados pelas quais o agente é deslocado é indicada nas colunas apresentadas na Figura 6. Todos os movimentos do agente são recompensados com valor -1 até que o objetivo seja alcançado.

Figura 6 – Ambiente Windy Gridworld.

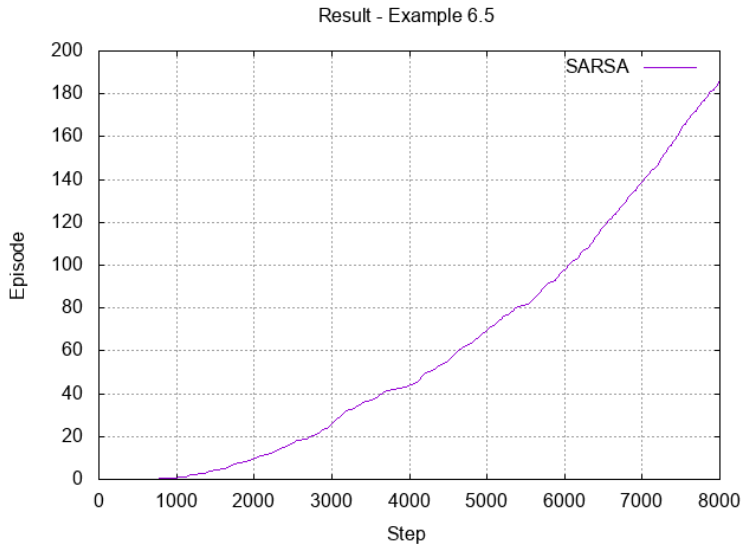


Fonte: Adaptado de (SUTTON; BARTO, 2018)

Os elementos de estado, ação e recompensa foram modelados de acordo com a descrição apresentada no exemplo. Utilizando a biblioteca TD_algorithm, foram realizados

200 episódios. Em cada episódio, foi registrada a quantidade de passos (steps) necessários para que o agente alcançasse o estado final e concluísse a tarefa. A Figura 7 apresenta os resultados obtidos no experimento. Observa-se que, à medida que o agente acumula experiência a partir dos episódios anteriores, ocorre uma redução na quantidade de passos necessários para alcançar o objetivo.

Figura 7 – Resultados obtidos no Exemplo 6.5 - Windy Gridworld.



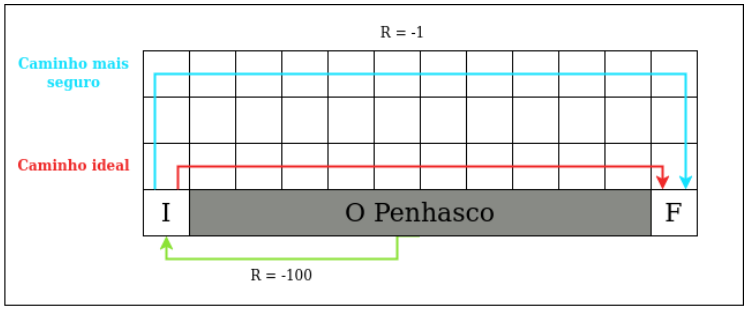
Fonte: Adaptado de (SUTTON; BARTO, 2018)

4.1.2 Experimento 2

O segundo experimento reproduz o Exemplo 6.6 chamado *Cliff Walking*, cujo objetivo é comparar o desempenho dos algoritmos Sarsa e Q-Learning. O ambiente consiste em um conjunto finito de estados organizados em forma de grade. Dentre esses estados, há um estado inicial (I), a partir do qual o agente começa cada episódio, e um estado final (F), que encerra o episódio quando alcançado. O agente pode se mover para os estados localizados acima, abaixo, à direita ou à esquerda. No entanto, caso o agente entre na região nomeada "O Penhasco", ele retornará para o estado inicial. O agente recebe recompensa constante de -1 em todas as transições, exceto quando acessa o penhasco, nesse caso a recompensa é -100.

Os elementos de estado, ação e recompensa foram modelados de acordo com a descrição apresentada no exemplo. Utilizando a biblioteca TD_algorithm, foram realizados 500 episódios para cada algoritmo. Em cada episódio, foi registrada a soma das recompensas até que o agente alcançasse o objetivo. A Figura 9 apresenta os resultados obtidos no experimento. Observa-se que para ambos os algoritmos, à medida que o agente acumula experiência, as penalidades diminuem. Quando aplicado o algoritmo Q-learning, o agente aprendeu o caminho ideal, mas ocasionalmente isso o levou a cair no penhasco gerando

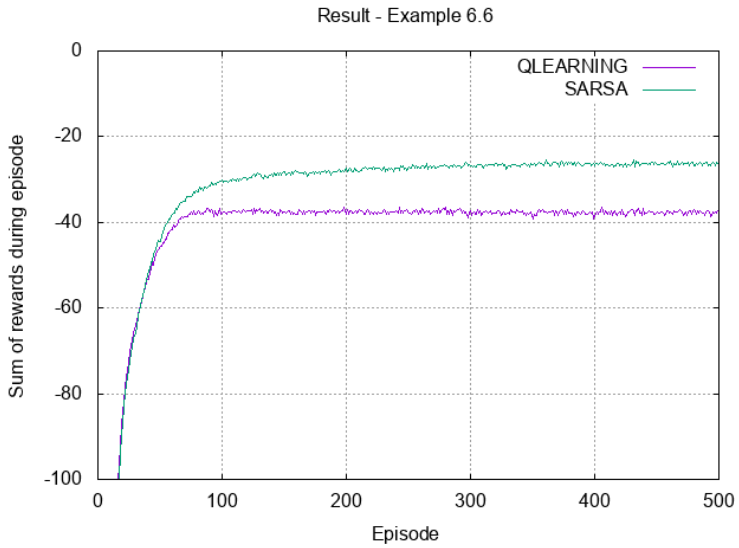
Figura 8 – Ambiente Cliff Walking.



Fonte: Adaptado de (SUTTON; BARTO, 2018)

penalidades altas. Em contra partida, quando aplicado o algoritmo Sarsa, o agente escolheu percorrer o caminho mais longo, porem mais seguro, o que resultou em penalidades menores.

Figura 9 – Resultados obtidos no Exemplo 6.6 - Cliff Walking



Fonte: Adaptado de (SUTTON; BARTO, 2018)

4.2 Validação do Agente

A validação do agente teve como objetivo verificar o correto funcionamento do processo de percepção do estado do ambiente e seleção de ações. Os experimentos demonstraram que o agente foi capaz de observar as métricas da rede e ajustar dinamicamente o valor do parâmetro CW_MAX e AIFS de acordo com os estados identificados, além de obter a recompensa da ação anterior.

Embora a modelagem inicial previsse a utilização de métricas adicionais da camada MAC, dificuldades na coleta das métricas ACK_FAIL e FCS_FAIL impediram sua utilização no experimento final. Dessa forma, o modelo foi simplificado para considerar apenas as métricas disponíveis durante a execução do sistema. Essa simplificação não

compromete o objetivo desta etapa do trabalho, que é demonstrar o funcionamento do agente e sua capacidade de observar o estado da rede e ajustar parâmetros do protocolo IEEE 802.11.

O experimento foi executado com os seguintes parâmetros:

- **Duração:** 225 segundos
- **Intervalo de iterações do agente:** 5 segundos
- **Estado simplificado:** TX_FRAMES

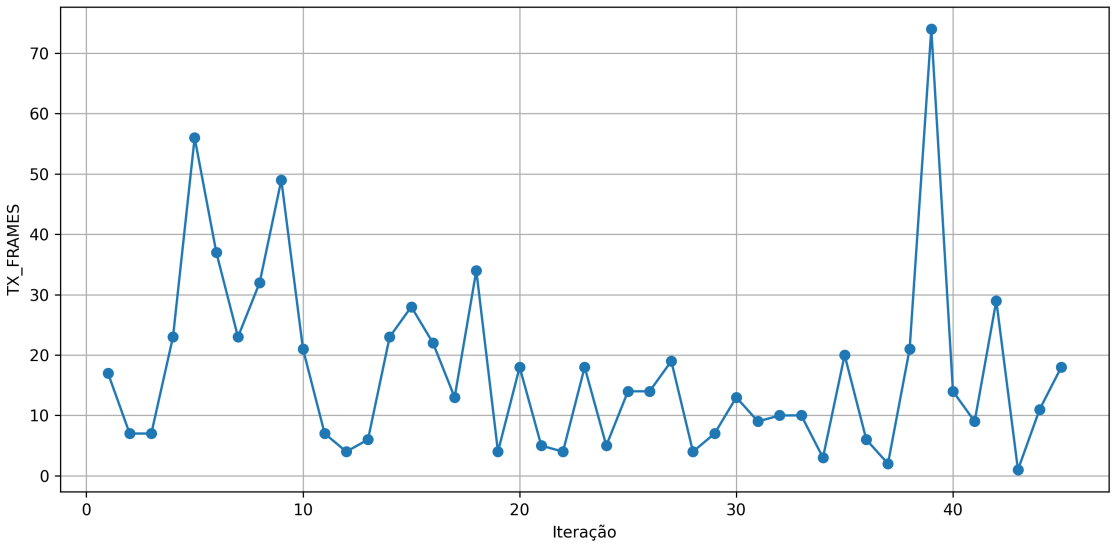
A Figura 10a e a Figura 10b mostram as variações das métricas coletadas pelo agente ao longo das iterações do experimento. Observa-se que os valores de TX_FRAMES variaram significativamente entre as iterações. De forma semelhante, as médias de AIRTIME também apresentam variações durante o experimento, indicando diferentes níveis de ocupação do meio de comunicação.

A Figura 11a e a Figura 11b mostram as tomadas de decisão do agente durante a execução do experimento. Observa-se que ambos os parâmetros foram ajustados diversas vezes durante a execução do experimento, indicando que o agente está explorando o conjunto de ações disponíveis.

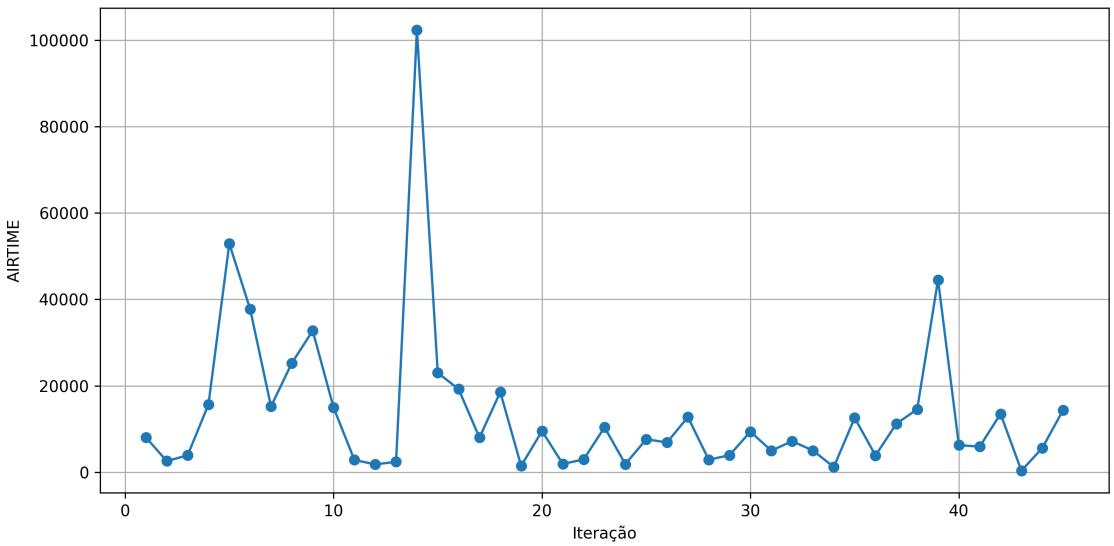
Os resultados demonstram que o agente é capaz de monitorar o ambiente, compondo os estados à partir das métricas e selecionar ações que modificam os parâmetros do protocolo MAC. Ainda que o experimento não tenha como objetivo avaliar ganhos de desempenho da rede, os resultados confirmam o funcionamento do mecanismo de decisão do agente e sua capacidade de atuar dinamicamente sobre os parâmetros de controle do acesso ao meio.

Figura 10 – Métricas coletadas durante a execução do agente.

(a) TX_FRAMES por iteração.



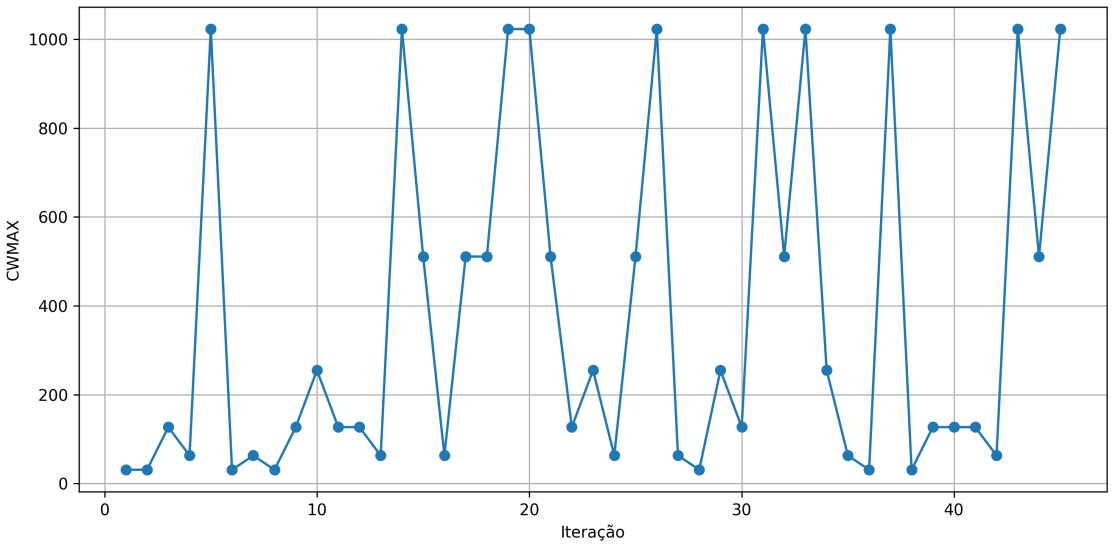
(b) AIRTIME por iteração.



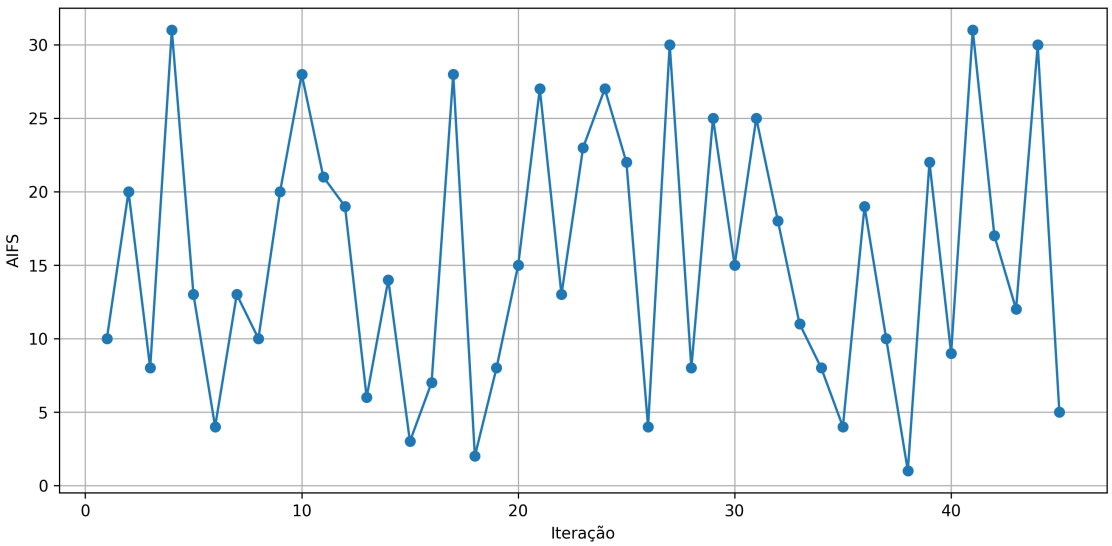
Fonte: Elaborado pelo autor

Figura 11 – Ações tomadas pelo agente durante a execução do experimento.

(a) CWMAX por iteração.



(b) AIFS por iteração.



Fonte: Elaborado pelo autor

5 CONCLUSÕES

Este trabalho teve como objetivo o desenvolvimento e avaliação de um agente baseado em técnicas de AR para o ajuste dinâmico de parâmetros do protocolo MAC CSMA/CA utilizado em redes IEEE 802.11. Esse agente busca minimizar a ociosidade do canal de comunicação por meio do ajuste do tempo de impedimento preventivo para transmissão, expressado pela janela de disputa. Com isso, o protocolo MAC pode ser ajustado em função do nível de disputa presente na rede sem-fio, o qual depende da quantidade de estações ativas e de seus padrões de geração de tráfego.

Um primeiro resultado deste trabalho foi a definição de um modelo para o agente de AR, o qual é formado por estado, ações e recompensa. O estado é composto por informações sobre quadros transmitidos e eventuais erros. As ações envolvem alterar a duração do tamanho máximo da janela de disputa. Por fim, a recompensa foi definida em função do airtime, de forma que menor airtime implica maior recompensa. Com esse modelo, o agente deve aprender que ações tomar em cada estado para minimizar o airtime e, com isso, o tempo médio gasto para enviar quadros.

Outro resultado desse trabalho foi o desenvolvimento de uma biblioteca em linguagem C que implementa os algoritmos de AR junto com a implementação do agente voltado ao subsistema mac80211 do kernel Linux. Os algoritmos implementados na biblioteca foram o SARSA e Q-Learning que possuem baixa complexidade computacional, possibilitando seu uso em sistemas embarcados sem comprometer o restante do sistema. O software foi desenvolvido de maneira que possibilita a integração com o subsistema mac80211, sugere uma modelagem para os elementos do AR e possibilita a investigação do comportamento do protocolo CSMA/CA sob diferentes políticas adaptativas, viabilizando análises comparativas de desempenho em diferentes cenários.

Para verificar o funcionamento da biblioteca desenvolvida, foram realizados experimentos baseados em exemplos clássicos da literatura de aprendizado por reforço, como os ambientes Windy Gridworld e Cliff Walking. Esses experimentos permitiram validar a implementação dos algoritmos SARSA e Q-Learning, demonstrando a capacidade da biblioteca em aprender políticas de ação adequadas ao longo dos episódios.

Posteriormente, foi realizada a integração do agente com o subsistema mac80211 do kernel Linux. Durante a execução, o agente coletou métricas do protocolo MAC, como quantidade de quadros transmitidos e airtime, utilizando essas informações para compor o estado do ambiente e a recompensa. A partir desses estados, o agente passou a selecionar ações que modificam parâmetros do protocolo, como o tamanho máximo da janela de disputa (CWMAX) e o valor de AIFS. Os resultados obtidos demonstraram que o agente

foi capaz de adaptar dinamicamente esses parâmetros ao longo das iterações, evidenciando o funcionamento do mecanismo de decisão baseado em aprendizado por reforço.

Como trabalhos futuros, destaca-se a realização de experimentos mais extensivos para avaliar o impacto do agente em métricas de desempenho, como vazão, latência, taxa de colisões e justiça entre estações. Além disso, podem ser exploradas extensões do modelo para considerar diferentes categorias de tráfego definidas pelo IEEE 802.11, bem como a comparação entre os diferentes algoritmos de AR quanto à convergencia e estabilidade em ambientes dinâmicos.

REFERÊNCIAS

GARG, V. K. Chapter 21 - wireless local area networks. In: GARG, V. K. (Ed.). *Wireless Communications Networking*. Burlington: Morgan Kaufmann, 2007, (The Morgan Kaufmann Series in Networking). p. 713–776. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780123735805500557>. Citado na página 19.

GAST, M. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, 2005. (A Nutshell handbook). ISBN 9780596100520. Disponível em: <https://books.google.com.br/books?id=9rHnRzzMHLIC>. Citado na página 19.

IBM. *Machine Learning e Ciência de dados com IBM Watson*. 2021. Disponível em: <https://www.ibm.com/br-pt/analytics/machine-learning>. Acesso em: 01 mar 2022. Citado na página 26.

IEEE. Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, p. 1–3534, 2016. Citado na página 19.

IEEE. Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications - redline. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) - Redline*, p. 1–7524, 2021. Citado 2 vezes nas páginas 23 e 24.

IEEE Standards Association. *The Evolution of Wi-Fi Technology and Standards*. s.d. Disponível em: <https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/>. Acesso em: 13 dez 2025. Citado na página 23.

ITU. *Statistics*. 2021. Disponível em: <https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx>. Acesso em: 18 feb 2022. Citado na página 19.

KLOIBER, B. et al. Improving information dissemination in sparse vehicular networks by adding satellite communication. In: *2012 IEEE Intelligent Vehicles Symposium*. [S.l.: s.n.], 2012. p. 611–617. Citado na página 25.

KUROSE, J. F.; ROSS, K. W. Computer networking: A top down approach sixth edition. 2014. Citado na página 24.

SOUSA, C. B. de. *Um Agente baseado em Aprendizado por Reforço para Ajuste de Parâmetros de um MAC CSMA/CA*. 2025. <https://github.com/camillabarreto/agente-csma-ca>. Acesso em: 15 dez. 2025. Citado na página 31.

SUTTON, R.; BARTO, A. *Reinforcement Learning, second edition: An Introduction*. MIT Press, 2018. (Adaptive Computation and Machine Learning series). ISBN 9780262352703.

Disponível em: <https://books.google.com.br/books?id=uWV0DwAAQBAJ>. Citado 8 vezes nas páginas 26, 27, 28, 29, 31, 43, 44 e 45.