Uma abordagem Q-Learning para escalonamento dinâmico de comunicação do TSCH

Victor S. Cardel[1], **Paulo H. L. Rettore**[2], **Bruno P. Santos**[1]

[1]Instituto de Computação
Universidade Federal da Bahia (UFBA) – BA – Brasil.

[2]Communication Systems Department (KOM)
Fraunhofer FKIE, Bonn, Germany.

{victor.cardel, bruno.ps}@ufba.br

paulo.lopes.rettore@fkie.fraunhofer.de

Resumo. *Uma rede mesh 6TiSCH provê conectividade IPv6 usando enlaces IEEE 802.15.4 governados pelo Time Slotted Channel Hopping (TSCH). Essencialmente, o TSCH promete baixo consumo de energia e alta confiabilidade através do escalonamento de tempo e salto de canais de comunicação, respectivamente. Entretanto, o 6TiSCH não define as políticas para construir e manter o cronograma de comunicação. Este trabalho propõe uma nova função de escalonamento de comunicação que utiliza Q-Learning, que leva em consideração a variação no tráfego da rede, o consumo de energia e o tamanho da fila de mensagens a serem enviadas pelo dispositivo. Comparamos a abordagem proposta com Minimal Scheduling Function (MSF), o escalonador de facto usada na literatura. Os experimentos mostram que a abordagem proposta reduz a latência da comunicação, enquanto mantém a confiabilidade alta, o consumo de energia e tempo de junção da rede baixos, mostrando que a abordagem é promissora.*

Abstract. *A 6TiSCH network provides IPv6 connectivity through links IEEE 802.15.4 governed by Time Slotted Channel Hopping (TSCH). Essentially, TSCH intends to provide low energy consumption and high reliability by scheduling communication resources and the channel hopping mechanism. However, 6TiSCH does not define the policies to build and maintain the communication schedule. This work proposes a new scheduling function that utilizes Q-Learning, considering the network traffic variation, the energy consumption, and the message queue size to be sent by the device. We compared the proposed method with Minimal Scheduling Function (MSF), the de facto scheduler used in the literature. The experiments show that the proposed method reduces the communication latency while maintaining high reliability and low energy consumption and join times. This indicates the potential of the approach.*

1. Introdução

A Internet das Coisas (do inglês Internet of Things (IoT)) intensificou a interconectividade entre dispositivos na última década [Santos et al. 2016]. Mais recentemente, uma das áreas que mais se beneficia é a indústrial, e a integração de tecnologias IoT neste cenário é denominada Industrial Internet Of Things (IIoT) [Hazra et al. 2021].

Diante deste cenário complexo, as Low Power and Lossy Networks (LLN), utilizadas na IIoT, podem promover o aumento da produtividade e da segurança, além da redução dos custos nas plantas industriais, desde que os requisitos de confiabilidade e economia no consumo de energia sejam alcançados [Peter et al. 2023]. Dito isto, há certas características de redes LLN que tornam essa tarefa desafiadora. Entre elas, o fato de que os dispositivos devem compartilhar o meio de transmissão implica em problemas como interferência no sinal, *multi path fading* e aumento no consumo de energia, o que pode reduzir a confiabilidade da rede de forma considerável [Vilajosana et al. 2019].

Os esforços para conectar redes industriais à Internet culminaram no desenvolvimento do IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) [Thubert 2021]. O Time Slotted Channel Hopping (TSCH) foca somente no controle de acesso ao meio, isto permite que o protocolo seja subjacente a pilhas de protocolos habilitadas para LLNs. Podendo executar abaixo do IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN)[1], o qual habilita o uso do IPv6 em LLNs; e do IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)[2], responsável pelo roteamento. Uma típica organização lógica de rede 6TiSCH abarca uma estrutura chamada Destination-Oriented Directed Acyclic Graph (DODAG), em que a raiz é um dispositivo conectado à Internet. Neste tipo de rede, o tráfego flui tipicamente de baixo para cima, tendo a raiz como seu destino final.

Na arquitetura 6TiSCH, a função de escalonamento de intervalos de tempo de envio de quadros do TSCH é um componente fundamental. Entretanto, o TSCH apenas especifica como a comunicação ocorre, mas não diz nada sobre como os intervalos de tempo para comunicação serão distribuídos entre os dispositivos. Esta tarefa fundamental é deixada para o escalonador, cuja implementação é deixada em aberto pelo grupo de trabalho 6TiSCH. Existem diversas implementações possíveis para um escalonador, porém o grupo de trabalho 6TiSCH formalizou a chamada Minimal Scheduling Function (MSF) [Hauweele et al. 2020]. MSF é um escalonador reativo (aqueles que reagem ao estado da rede), que tenta predizer se uma célula (intervalo de tempo de comunicação) deve ser inserida ou removida do *slotframe* (janela de tempo que contém diversas células de comunicação) ao se basear em uma estimativa do tráfego da rede.

Diversos trabalhos propuseram modelos de escalonadores, seguindo diferentes estratégias. Enquanto alguns trabalhos optam por utilizar abordagens distribuídas ou probabilísticas para construir seus escalonadores [Ha and Chung 2022], outros optam pelo método baseados em aprendizagem de máquina, mais especificamente, aprendizado por reforço não supervisionado através do algoritmo Q-Learning [Kherbache et al. 2023]. Q-Learning é atrativo no cenário do 6TiSCH por permitir alta adaptabilidade a um custo relativamente baixo de processamento e memória, uma propriedade importante para dispositivos potencialmente restritos em termos de recursos.

Este trabalho, inspirado nas ideas discutidas em [Kherbache et al. 2023, Santos et al. 2019], propõe um escalonador que explora o potencial da Inteligência Artificial (IA) e do Q-Learning para resolver a tarefa de escalonamento, e tenta alcançar este objetivo ao escolher a ação que representa o melhor momento para inserir ou remover uma célula no *slotframe*, computada pelo Q-Learning. De maneira complementar, o número de células a serem inseridas ou removidas é estimado em um passo separado, o

[1]RFC 6282
[2]RFC 6550

que permite uma maior economia no número de colunas da Q-Table. O cálculo da ação e do número de células são baseados em características da rede, como tráfego, utilização do *buffer* e consumo de energia dos dispositivos. Finalmente, mostramos os resultados de nossa abordagem com experimentos comparando-a com a MSF. Em resumo, este estudo apresenta as seguintes contribuições:

- Metodologia Q-learning para escalonamento dinâmico de comunicação TSCH usando metricas como o tráfego, o tamanho do *buffer* e o consumo de energia do dispositivo para montar o escalonamento.
- Metodologia de código aberto[3].
- Um comparativo qualitativo entre as abordagens de escalonamento de comunicação TSCH disponíveis na literatura.
- Resultados experimentais quantificando a eficácia da abordagem Q-Learning na redução da latência fim-a-fim da rede enquanto mantém o tempo de vida da rede compatível com a MSF.

O restante do artigo está estruturado da seguinte forma: A Seção 2 aborda o referencial teórico e trabalhos relacionados. A Seção 3 apresenta o método Q-learning proposto, enquanto a Seção 4 descreve os cenários de avaliação e apresenta os resultados experimentais. Concluímos com discussões sobre os resultados na Seção 5.

2. Referencial Teórico e Trabalhos Relacionados

2.1. Arquitetura 6TiSCH

Uma das características chave do 6TiSCH é o Time Slotted Channel Hopping (TSCH). TSCH combina dois fatores em um protocolo de acesso ao meio: alocação de tempo para comunicação e mudança de canais de comunicação. Cada fator age de forma a mitigar um problema diferente de redes LLNs, a saber consumo de energia e confiabilidade.

O TSCH delimita um instante de tempo e um canal, na forma de uma tupla $(timeslot, channelOffset)$, a qual é chamado de célula de comunicação. Assim, os dispositivos podem ligar ou desligar seus rádios em períodos predeterminados, consequentemente reduzindo o seu consumo de energia. Cada célula possui tempo pré-determinado, designado *timeslot*. O valor de um *timeslot* é usualmente definido como sendo $10\,\mathrm{ms}$, que é o tempo necessário para que um dispositivo envie uma mensagem e receba uma confirmação. Os *timeslots* e canais são organizados em uma matriz, chamada *slotframe*. O comprimento usual de um *slotframe* é 101 *timeslots*, os quais se repetemcanal periodicamente durante o tempo de vida da rede.

O segundo fator é a mudança do canal de comunicação. A cada troca de quadros entre dispositivos, uma frequência de comunicação diferente é utilizada. O processo de escolha dos canais (frequência) de comunicação, ao longo da execução do protocolo, é chamado de *Channel Hopping*. Esse processo tende a resultar em maior confiabilidade de entrega de mensagens (evitando, por exemplo, colisões no canal de comunicação). Isso significa que se a transmissão de um pacote falha por alguma razão, sua retransmissão provavelmente acontecerá em um canal diferente, com uma probabilidade maior de sucesso no envio [Duquennoy et al. 2017].

[3]Link para o código fonte no Github.

6TiSCH usa o RPL [Brandt et al. 2012] como procolo de roteamento. Isto resulta em uma topologia lógica da rede configurada como um DODAG, tendo como destino final das mensagens o dispositivo raiz. Pelas regras do protocolo RPL, dispositivos podem ter somente um pai efetivo, entretanto é mantida uma lista de potenciais pais na árvore de roteamento RPL. Desta forma, dispositivos 6TiSCH se comunicam em um padrão *botton-up*, com cada filho comunicando-se com o seu pai, por um processo de negociação de células mediado por um protocolo chamado 6P [Wang et al. 2018]. Consequentemente, é usual considerar nas análises somente o tráfego na direção filho-pai, apesar da ocorrência necessária de tráfego em outras direções, como na ocasião de uma confirmação, por exemplo. A Figura 1(a) exibe a estrutura de um *slotframe* com 5 *timeslots* e 4 canais, e um DODAG (ou aŕvore) RPL para uma rede com 5 dispositivos.

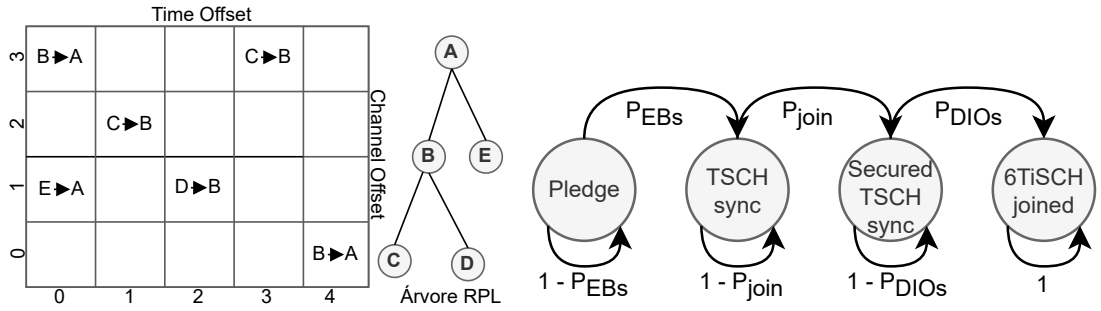
Cada dispositivo decide o instante e o canal que irá enviar uma mensagem ao utilizar uma célula em seu *slotframe*. Para que esse esquema seja bem sucedido, os dispositivos precisam informar o seu remetente, usualmente o seu pai RPL ou um de seus vizinhos, acerca das células que pretendem utilizar. Para isso, os dispositivos precisam estar sincronizados, o que implica em troca de mensagens de controle para que informações da rede sejam compartilhadas entre todos. Uma dessas informações é um canal comum para a troca deste tipo de informação, representado por uma célula chamada *minimal cell*. A *minimal cell* localiza-se na posição reservada (0,0) de um *slotframe*, e todos os pacotes de controle são enviados por este canal.

A sequência de etapas que um dispositivo deve realizar até que esteja devidamente sincronizado, autenticado e inserido no DODAG é denominado *join process*. Um dispositivo que passou por todas as etapas do *join process* é dito um dispositivo 6TiSCH [Kalita and Khatua 2022]. As etapas são, respectivamente, sincronização com a rede, autenticação segura e obtenção de um pai RPL através de recebimento de um pacote DODAG Information Object (DIO). A Figura 1(b) mostra os passos que um dispositivo deve completar para se tornar-se um dispositivo 6TiSCH, juntamente com os pacotes de controle necessários para completar cada transação. As transições indicam que o dispositivo prossegue para outro estado de acordo com uma probabilidade de receber um pacote de controle, e.g.: P_{DIOs} é a probabilidade de se receber um DIO, um pacote de controle do RPL. O primeiro estado, denominado *pledge*, indica um dispositivo que não passou por nenhuma etapa do *join process*, e está buscando um pacote de controle para se sincronizar.

Um dispositivo que ainda não se tornou um dispositivo 6TiSCH ainda não está sincronizado, e portanto não tem como saber a localização da célula mínima. As informações necessárias para realizar a sincronização são enviadas em um pacote de controle chamado Enhanced Beacon (EB). Para que um dispositivo receba um EB, ele deve escanear todas as frequências disponíveis, o que gasta uma quantidade significativa de tempo e energia. Portanto, é importante para a performance da rede como um todo que o tempo para que um dispositivo se conecte deva ser minimizado [Kalita and Khatua 2022].

2.2. Aprendizado por Reforço e o algoritmo Q-Learning

Aprendizado por reforço [Wiering and Van Otterlo 2012] é uma subárea de aprendizado de máquina que modela o processo de aprendizado de um agente através dos conceitos de estados, ações e recompensas. Nesta abordagem, um agente encontra-se em um estado, e pode tomar uma ação para transicionar para um outro estado subsequente. A ação tomada



(a) TSCH slotframe (b) Join *process* [Kalita and Khatua 2022]

Q-table

	a_1	a_2	a_3
s_1	5	3.1	-1
s_2	0	8	1
s_3	3	4.4	5

(c) Q-Table

Figura 1. Slotframe com árvore RPL, *join process* e Q-Table

está sujeita a uma recompensa, que informa ao agente o benefício de se realizar esta ação. Em termos matemáticos, um algoritmo de aprendizado por reforço possui um conjunto de estados S, um conjunto de ações A, e a função de recompensa $r_a(s, s')$ [Ullah et al. 2020].

O Q-learning é um algoritmo de aprendizado por reforço que vem tendo destaque em diversas aplicações na área de rede de computadores e afins [Clifton and Laber 2020]. Exemplos da aplicação no contexto de redes sem fio de baixa potência incluem estratégias de escalonamento de recursos, otimização de funções objetivo roteamento (ex.: RPL) e em temporizadores de mensagens de controle [Fawwaz and Chung 2023, Bekar et al. 2023, Bekar et al. 2023]. Suas características explicam o motivo do seu uso na área como, por exemplo, o fato do algoritmo não necessitar de nenhum conhecimento prévio acerca do ambiente para derivar a política (sequência de ações) ótima. Apesar disso, Q-Learning pode apresentar um custo elevado para o agente nos estados iniciais do treinamento, pois para derivar a política ótima, todas as ações tem que ser realizadas diversas vezes, mesmo que algumas dessas ações não gerem um resultado desejado para o agente. Em adição, existe o compromisso entre *exploration* e *exploitation* que deve ser equilibrado adequadamente. *Exploration* é a etapa em que o agente escolhe uma ação aleatória para realizar. Por outro lado, *exploitation* é a etapa em que o agente escolhe a melhor ação a ser executada com base na política derivada durante o treinamento.

O Q-Learning é implementado através de uma tabela de estados-ações, a Q-table (vide Figura 1(c)). O agente utiliza a Q-Table para escolher a melhor ação tendo em vista o estado em que ele se encontra, e a cada execução de uma ação, o agente muda para um novo estado. Na Figura 1(c)), as linhas da tabela representam os estados, enquanto

as colunas as possíveis ações. Cada entrada $Q(s, a)$ dessa tabela indica a recompensa esperada para o par estado-ação. Por exemplo, a entrada $Q(s_1, a_2)$ retorna a recompensa esperada com valor 3.1. Assim, um agente em fase de *exploitation* escolherá a ação que produza maior recompensa. A Equação (1) reflete este comportamento. Por outro lado, se o agente estiver em fase *exploration*, uma ação é escolhida aleatoriamente, o que se traduz na Equação (11). Como indicado na Equação (5), o parâmetro ϵ define a probabilidade de um agente estar em fase de *exploitation* ou *exploration*. Os parâmetros ϵ_{min}, ϵ_{max} e $decayRate$ são parâmetros do modelo, e indicam, respectivamente, o valor mínimo e máximo de ϵ e a taxa com que ϵ decai ao longo do tempo. O valor de ϵ diminui ao longo dos episódios de treino para que o agente utilize cada vez mais a política estimada para tomar decisões. No início, o agente possui mais liberdade para explorar, evitando que o mesmo fique preso em uma estratégia sub-ótima.

Treinar o agente significa atualizar os valores da Q-table. A tabela é atualizada de acordo com a Q-function, expressa pelas Equações (3) e (4). As fórmulas dependem dos parâmetros α e γ, obtidos experimentalmente, e cujos valores encontram-se entre 0 e 1. A Equação (3) calcula a recompensa estimada de uma ação com base em sua recompensa, adicionada a ação que maximiza a recompensa do próximo estado (escolha gulosa).

$$a_{opt} = \arg\max_{a} Q(s, a) \tag{1}$$

$$a_t = \begin{cases} a_{opt}, & 1 - \epsilon \\ a, & \epsilon \end{cases} \tag{2}$$

$$\Delta Q(s, a) = r(s, a) + \gamma \times \max_{a} Q(s_{next}, a) \tag{3}$$

$$Q(s, a)_{new} = (1 - \alpha) \times Q(s, a)_{old} + \alpha \times \Delta Q(s, a) \tag{4}$$

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \times e^{-decayRate \times episode} \tag{5}$$

2.3. Trabalhos relacionados

Como já comentado, o grupo de trabalho 6TiSCH deixou em aberto a especificação do escalonador que determina a distribuição de células no *slotframe* de um dispositivo. Por consequência, diversos trabalhos propuseram um conjunto variável de estratégias para resolver este problema. Por exemplo, em [Hauweele et al. 2020], os autores proveem uma descrição detalhada da MSF, além da descrição de experimentos realizados com o intuito de verificar até que ponto este método consegue adaptar-se a variação do tráfego na rede. A conclusão obtida é que MSF consegue adaptar-se ao tráfego, apesar de alocar mais células do que seria estritamente necessário para atender a demanda a qual está sujeita. Além disso, a taxa de alocação de novas células depende do número de células previamente alocadas no *slotframe*, o que pode acarretar em uma adaptação mais demorada caso nenhuma célula esteja alocada e o tráfego aumente rapidamente.

De modo semelhante, em [Hamza and Kaddoum 2019] os autores procuram estender a MSF, derivando uma distribuição de probabilidade de *Poisson* a qual procura

Tabela 1. Comparativo entre trabalhos relacionados.

Referências	Estratégia	Otimização	Q-Learning	Tráfego	Buffer	Energia
[Hamza and Kaddoum 2019]	Previsão do tráfego	Escalonamento	✗	✓	✗	✗
[Hauweele et al. 2020]	Adaptação ao tráfego	Escalonamento	✗	✓	✗	✗
[Domingo-Prieto et al. 2016]	Controlador PID	Escalonamento	✗	✓	✓	✗
[Palattella et al. 2012]	Coloração/Emparelhamento	Escalonamento	✗	✗	✗	✗
[Fawwaz and Chung 2023]	Q-Learning	Envio/Supressão DIOs	✓	✗	✓	✗
[Nguyen-Duy et al. 2019]	Q-Learning	Escalonamento	✓	✗	✓	✓
[Bekar et al. 2023]	Q-Learning	OFs para RPL	✓	✓	✓	✗
[Pratama and Chung 2022]	Q-Learning	Escalonamento	✓	✗	✓	✗
Este trabalho	**Q-Learning**	**Escalonamento**	✓	✓	✓	✓

prever o número de quadros que serão enviados no próximo *slotframe*. Assim, o algoritmo pode escolher alocar ou liberar células de acordo com o número de quadros estimados. [Domingo-Prieto et al. 2016] investigaram o problema de escalonamento modelando-o como um problema de controle. A estratégia consistiu em desenvolver um controlar PID que remove e insere células automaticamente do *slotframe* de acordos com parâmetros como tamanho do *buffer* e utilização da *minimal cell*. Os resultados, os quais comparam o controlador PID com o escalonador Orchestra [Duquennoy et al. 2015], indicam que o controlador apresenta um tempo ligeiramente maior para alocar ou desalocar células, apresentando um *overhead* no processo de adaptação.

Enquanto as metodologias citadas até então são distrubuídas, existe a opção de se construir um escalonador centralizado. O algoritmo TASA (Traffic Aware Scheduling Algorithm) [Palattella et al. 2012] é um exemplo de tal abordagem. Nesse contexto, a rede é=foi modelada como um grafo, e o escalonamento é construído de maneira centralizada utilizando técnicas de emparelhamento e coloração.

Outras metodologias utilizam aprendizado de máquina, e mais especificamente aprendizado por reforço através do algoritmo Q-Learning, para resolver problemas em arquiteturas 6TiSCH. Em [Fawwaz and Chung 2023], os autores utilizaram Q-Learning para otimizar o temporizador *trickle-timer* do RPL, utilizando a metodologia para tentar derivar uma sequência ótima de envios e supressões de pacotes DIO com base em características da rede, como utilização da célula mínima e tamanho da fila. Além disso, os autores propuseram fórmulas para tornar os intervalos dos valores de parâmetros do *trickle-timer* variáveis, ao invés de fixos. Os resultados indicam que o Q-Learning foi promissor em sua aplicação, reduzindo o tempo do *join time* da rede como um todo.

Já em [Bekar et al. 2023], os autores também utilizaram Q-Learning para otimizar o RPL, mas focando em obter a função de objetivo para tomar melhores decisões de roteamento. No contexto de uma rede 6TiSCH, a troca frequente de pais RPL por um dispositivo pode acarretar em perda de performance, pois todas as células previamente alocadas para o antigo pai devem ser transferidas para o novo. Os autores abordaram esse desafio utilizando uma Q-Table em que cada coluna representava um vizinho do dispositivo, e cada ação representava a escolha daquele vizinho. Outros trabalhos utilizaram Q-Learning para tentar otimizar a construção do escalonamento de um dispositivo. O trabalho de [Nguyen-Duy et al. 2019] utilizou Q-Learning para decidir quantas células consecutivas devem ser ativadas com base no tamanho da fila e o nível de energia dos dispositivos, mas não forneceu uma análise da influência de tráfego variável.

O trabalho de [Pratama and Chung 2022] também utilizou Q-Learning para o pro-

blema de escalonamento, mas não considerou apenas células consecutivas, nem a energia restante no dispositivo. A função de recompensa levou em consideração a taxa de utilização do *buffer* juntamente com o número de células negociadas não utilizadas. O número de células a serem inseridas ou removidas do *slotframe* foi fixado nas colunas da Q-Table. É importante notar que essa decisão pode não escalar de maneira satisfatória, podendo haver desperdício de recursos caso o dispositivo não tenha necessidade do número máximo de células, que delimita o número de colunas, ou falta de recursos caso contrário.

É possivel notar que entre os trabalhos que empregaram Q-Learning para otimizar o escalonamento, nenhum avaliou a influência do tráfego. Além disso, não foi feita uma avaliação de variáveis fundamentais de uma arquitetura 6TiSCH, como a influência do algoritmo no *join process* da rede. A Tabela 1 apresenta o contraste entre nossa abordagem e os trabalhos relacionados, comparando-os de acordo com a estratégia utilizada para abordar o problema que se propunham a resolver.

3. Metodologia

3.1. Tomada de decisão acerca da inserção e remoção de células

Para aplicar Q-Learning no problema de escalonamento é necessário modelar um dispositivo como um agente. Tal agente encontra-se em um estado s, podendo tomar uma ação a, pela qual recebe uma recompensa. Na modelagem proposta, o estado consiste em uma tupla com três variáveis: i) t_b sendo a taxa de utilização do *buffer*; ii) t_{ack} sendo a taxa do número de pacotes os quais o dispositivo envia e recebe uma confirmação como resposta; iii) e_t, medido em mA, representando a energia residual no dispositivo.

As Equações (6) e (7) demonstram os cálculos de t_b, t_{ack}, respectivamente. O número k encontrado nas fórmulas é o número de *slotframes* anteriores considerados para o cálculo, sendo S o *slotframe* atual. O número Q é o número de pacotes no *buffer* no *slotframe* $S-k$, enquanto o número $RxAck$ é o número de pacotes rx confirmados pelo dispositivo no *slotframe* $S-k$. Usamos $k=10$ em nossos experimentos. A carga e_l consumida pelo dispositivo usou o modelo descrito em [Vilajosana et al. 2013].

Para que possamos limitar o número de estados possíveis em que um dispositivo pode se encontrar, é necessário computar estas medidas e categorizá-las de acordo com alguma função que as converta em valores inteiros. As Equações (8) (9) e (10) descrevem essa categorização das variáveis. Os valores dessas fórmulas foram obtidos através de experimentos, e são considerados parâmetros do algoritmo. A partir disso, um estado pode ser definido como uma tupla $(c_b(t_b), c_{Rx}(t_{ack}), c_e(e_l))$. A intuição para o uso do t_{ack} é que se um dispositivo possui o número de confirmações de pacotes recebidos mais alto do que o computado anteriormente, o tráfego provavelmente aumentou.

$$t_b = \left(\sum_{i=S-k}^{S} Q \right) / k \qquad (6) \qquad t_{ack} = \left(\sum_{i=S-k}^{S} RxAck \right) / k \qquad (7)$$

$$c_{Rx}(t_{ack}) = \begin{cases} 0, t_{ack} \leqslant 0.068 \\ 1, \text{caso contrário} \end{cases} \qquad (8) \qquad c_e(e_l) = \begin{cases} 0, e_l \leq 500 \\ 1, \text{caso contrário} \end{cases} \qquad (9)$$

$$c_b\left(t_b\right) = \begin{cases} 0, t_b \leqslant 0.118 \\ 1, \text{caso contrário} \end{cases} \quad (10)$$

Seguindo na modelagem, foi considerado que o agente possui três possíveis ações: inserir, remover ou manter células do seu *slotframe*. Por consequência, as possíveis ações podem ser descritas pela Equação (11):

$$A = \begin{cases} 0, \text{ Remove células} \\ 1, \text{ Insere células} \\ 2, \text{ Mantém o estado} \end{cases} \quad (11)$$

A última etapa da descrição do modelo é a função de recompensa, a qual atribui um número real a um par estado-ação. Assume-se que o agente deseja alcançar um estado com baixo tráfego, baixa ocupação de *buffer* e alta energia remanescente, denominado s_f, de forma que o agente será altamente recompensado por atingir tal configuração. Caso contrário, o agente será recompensado de acordo com o estado em que se encontra: alta ocupação de *buffer* e/ou tráfego são penalizados, enquanto uma alta energia remanescente é recompensada. A Equação (12) captura este conceito:

$$r(s,a) = \begin{cases} 3, \text{se } s = s_f \\ 1 - c_b\left(t_b\right) + 1 - c_{Rx}\left(t_{ack}\right) + c_e\left(e_l\right) \end{cases} \quad (12)$$

O momento da tomada de decisão pelo agente ocorre da mesma forma que na MSF. Após a passagem de 100 células negociadas com o pai na árvore de roteamento RPL. Após esse período, o agente pode tomar a decisão de escolher uma ação aleatória entre as 3 decisões possíveis (*exploration*), ou invocar a Equação (1) para decidir a melhor ação a se tomar com base no Q-Learning (*exploitation*). A proporção entre *exploration* e *exploitation* é dada por ϵ, que varia ao longo da execução do algoritmo pela Equação (5) .

3.2. Cálculo do número de células

Outra etapa importante é a escolha do número de células a serem inseridas ou removidas do *slotframe*. A inserção ou remoção de uma célula exige a invocação do protocolo 6P, e esse processo acarreta em trocas de mensagem de controle. Ao se inserir ou remover células em grupos, ao invés de individualmente, se minimiza o *overhead* da invocação do 6P diversas vezes. Caso a decisão do agente seja a de inserir células, o número de células C_i a serem inseridas é dado por pela Equação (13). Enquanto o número C_r de células a serem removidas, caso a decisão seja a de remover células, é dado por (14):

$$C_i = c_b\left(t_b\right) + c_{Rx}\left(t_{ack}\right) + c_e\left(e_l\right) \quad (13) \qquad C_r = 3 - C_i \quad (14)$$

4. Avaliação

O algoritmo proposto, de código aberto[3], foi implementado no 6TiSCH Simulator [Municio et al. 2019], um simulador de redes 6TiSCH implementado em Python.

Tabela 2. Parâmetros dos experimentos.

Parâmetros	Valores
Tamanho do *buffer*	5
Duração do Experimento	3750 slotframes
Tamanho do Timeslot	10 ms
Tamanho Slotframe	101
Canais Disponíveis	1-16
α (Q-Learning)	0.7
γ (Q-Learning)	0.3

A Minimal Scheduling Function foi utilizada como *baseline* comparativo, pois representa uma referência para funções de escalonamento formalizada pelo grupo de trabalho 6TiSCH. Utilizamos como critérios de comparação: i) o tempo de sincronização médio dos dispositivos com a rede em segundos (s); ii) a latência fim-a-fim em segundos (s); e iii) o tempo de vida em anos e a taxa de pacotes entregues ao longo da simulação. A Tabela 2 exibe as variáveis experimentais utilizadas nas simulações.

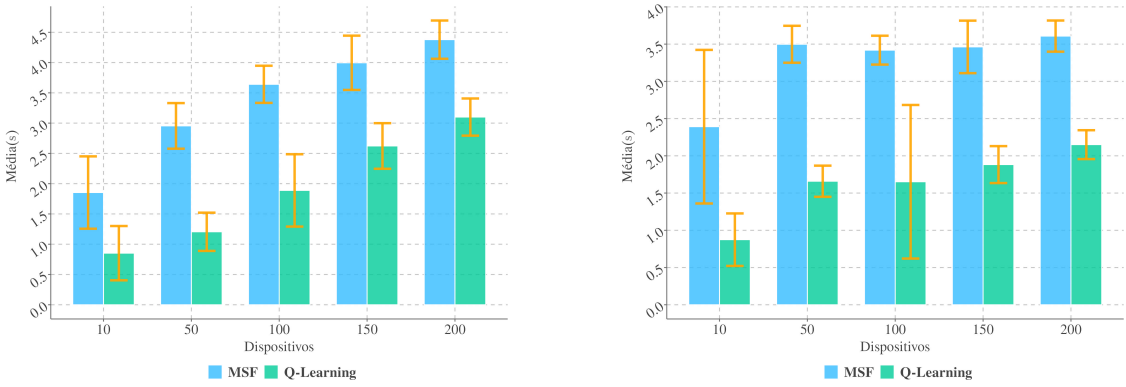
4.1. Cenários de avaliação

A Tabela 3 resume os cenários de avaliação do esquema de alocação. Duas aplicações e duas topologias foram utilizadas pelos experimentos com o intuito de analisar ambas as funções de escalonamento consideradas. A primeira aplicação, denominada inundação, possui o seguinte comportamento: cada dispositivo pode se comportar de duas maneiras, sendo elas enviar um único pacote periodicamente de $0.05\,\mathrm{s}$ em $0.05\,\mathrm{s}$ (modo periódico), ou enviar 5 pacotes em sequência em um período de $0.025\,\mathrm{s}$ (modo inundação). Somente $25\,\%$ dos dispositivos, escolhidos aleatoriamente a cada rodada do experimento, se comportam no modo inundação, com os demais utilizando o modo periódico. Essa aplicação possui a finalidade de testar as funções de escalonamento em cenários onde o tráfego apresenta alta variabilidade. A segunda aplicação enviava somente um único pacote em um período de $0.05\,\mathrm{s}$, e é denominada periódica. O intuito dessa aplicação era avaliar as funções de escalonamento em um cenário de tráfego estável.

Topologias em malha e aleatórias foram utilizadas para gerar cenários de avaliação. As topologias em malha representam cenários ideais em que não há perda de pacotes devido a boa qualidade de conexão entre os dispositivos. Nessa configuração, o dispositivo raiz encontrava-se na posição mais central da malha. Já as topologias aleatórias representam um cenário mais desafiador, em que a posição e o número de conexões entre os dispositivos eram gerados de maneira aleatória. Além disso, a qualidade das conexões entre dispositivos vizinhos também podia variar, havendo a possibilidade de perda de pacotes entre dispositivos por este motivo. A qualidade de conexão é indicada

Tabela 3. Cenários dos experimentos.

Cenários	
Aplicação	Periódica, Inundação
Topologia	Malha, Aleatória
Número de dispositivos	10, 50, 100, 150, 200



(a) Topologia malha com inundação

(b) Topologia randômica com inundação

Figura 2. Comparação das latências.

pela Received Signal Strength Indication (RSSI), a qual é implementada pelo 6TiSCH *Simulator* utilizando o modelo Pister-Hack [Le et al. 2009].

4.2. Experimentos

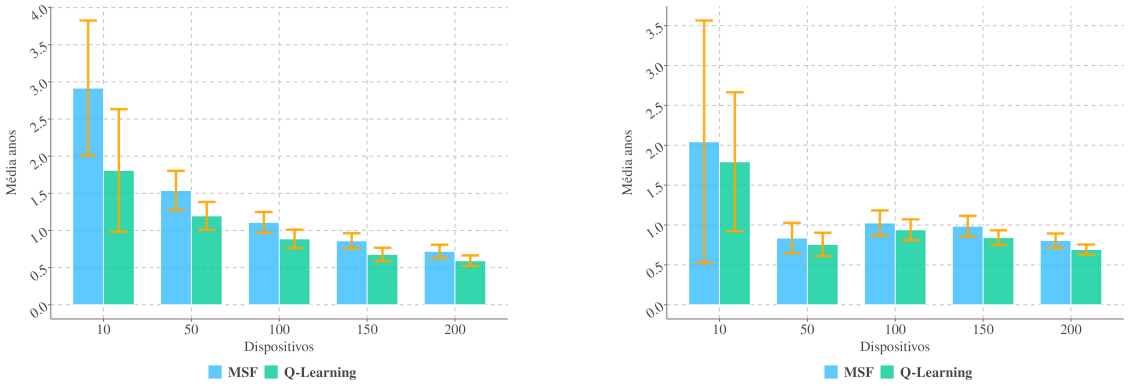
Para coletar os resultados, o mesmo conjunto de experimentos foi utilizado tanto para avaliar a MSF quanto nossa abordagem. Cada experimento consistiu em executar uma das combinações exibidas na Tabela 3 e coletar as métricas resultantes. Cada combinação foi executada 10 vezes, e os resultados derivados dessas execuções foram agregados utilizando a média, com intervalo de confiança de $99\,\%$.

4.3. Resultados

Consideramos quatro métricas de desempenho para avaliar as funções de escalonamento, a saber: i) latência; ii) tempo de vida da rede; iii) taxa de entrega de pacotes; e iv) tempo de adesão à rede (*Join Time*). A Figura 2 apresenta as latências observadas. É possível notar que houve uma redução de ao menos $2\,\mathrm{s}$ para as latências da abordagem que usa Q-Learning quando comparadas com as obtidas pela MSF. Isso provavelmente ocorre pois a abordagem com Q-Learning aloca uma quantidade maior de células de comunicação do que a MSF, o que aumenta a frequência de envio dos pacotes e reduz a latência.

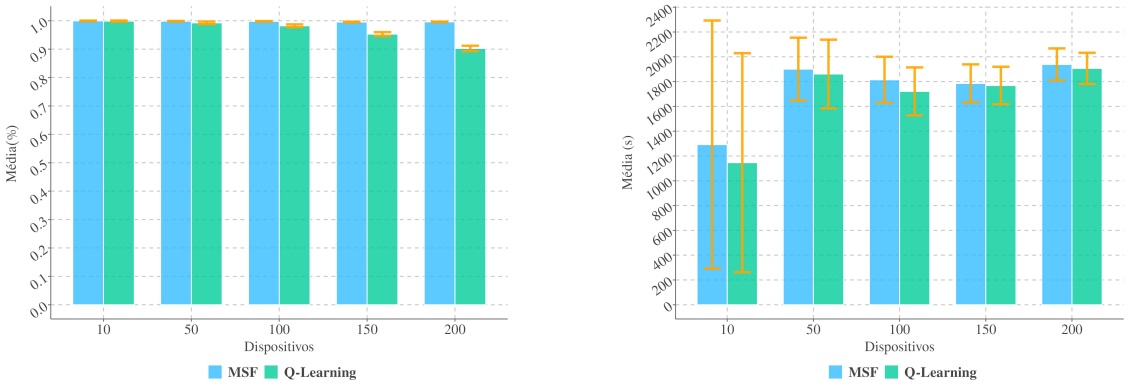
O tempo de vida da rede pode ser observado na Figura 3 para as duas topologias usando a aplicação inundação. É possível observar que ambos os métodos obtiveram tempos de vida comparáveis dentro do intervalo de confiança, o que demonstra que apesar da abordagem com Q-learning alocar mais células em média, ainda é capaz de manter um consumo de energia comparável com a MSF.

A Figura 4 mostra os resultados para as métricas restantes considerando a topologia de malha com aplicação periódica. A Figura 4(a) reproduz um padrão encontrado em todos os experimentos, porém mais acentuado na aplicação periódica, em que a taxa de entrega começa equiparada entre ambos os métodos, porém apresentando uma ligeira queda por parte da abordagem que usa Q-Learning a medida que o número de dispositivos avaliados aumenta. Uma explicação para esse comportamento é que como na aplicação periódica menos pacotes são enviados, a função de adaptação é invocada menos vezes, já que ela depende do critério da MSF de que 100 células decorram antes de ser chamada.



(a) Topologia malha com inundação (b) Topologia randômica com inundação

Figura 3. Comparação dos tempos de vida.



(a) Taxas de entrega (b) *Join time*

Figura 4. Taxas de entrega e *join time* em topologia malha e aplicação periódica

Isso faz com que o algoritmo Q-Learning tenha menos oportunidades de aprender a partir de observações, e consequentemente gera uma queda mínima de desempenho. Para o caso da aplicação de inundação, uma possível explicação é que a abordagem Q-Learning acaba perdendo uma quantidade maior de pacotes na fase inicial em que o modelo ainda encontra-se na fase de exploração.

É importante notar, porém, que a taxa de entrega se manteve muito próxima ou superior a $90\,\%$ em todos os casos considerados. Por fim, a Figura 4(b) mostra os *join times* na mesma configuração, demonstrando que esta métrica localiza-se comparável com a MSF dentro do intervalo de confiança.

5. Conclusão

Neste trabalho, foi proposto um escalonador Q-Learning de células de comunicação com base em características da rede, como a variação do tráfego, consumo de energia e taxa de ocupação da fila de mensagens. A quantidade de células a serem adicionadas ou removidas do *slotframe* 6TiSCH é estimada separadamente para reduzir o *overhead* da invocação do protocolo 6P. Os resultados experimentais, comparando a abordagem Q-Learning com a MSF, indicam uma significativa redução da latência, mantendo métricas como tempo de

vida, taxa de entrega e *join time* comparáveis à função MSF, considerada um importante *baseline* para funções de escalonamento 6TiSCH . Apesar dos resultados promissores na redução da latência, trabalhos futuros na metodologia envolverão a tentativa de melhorar a performance das outras métricas de interesse. É também um objetivo futuro automatizar a escolha dos parâmetros do algoritmo que atualmente foram definidos empiricamente.

Agradecimentos

Agradecemos o apoio da CAPES, CNPq, e FAPESB.

Referências

[Bekar et al. 2023] Bekar, T., Görmüş, S., Aydın, B., and Aydın, H. (2023). Q-Learning Algorithm Inspired Objective Function Optimization For IETF 6TiSCH Networks. In *SmartNets*. IEEE.

[Brandt et al. 2012] Brandt, A., Designs, S., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). Internet engineering task force (ietf) t. winter, ed. request for comments: 6550 category: Standards track p. thubert, ed.

[Clifton and Laber 2020] Clifton, J. and Laber, E. (2020). Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, 7:279–301.

[Domingo-Prieto et al. 2016] Domingo-Prieto, M., Chang, T., Vilajosana, X., and Watteyne, T. (2016). Distributed pid-based scheduling for 6tisch networks. *IEEE Communications Letters*, 20(5):1006–1009.

[Duquennoy et al. 2015] Duquennoy, S., Al Nahas, B., Landsiedel, O., and Watteyne, T. (2015). Orchestra: Robust mesh networks through autonomously scheduled TSCH. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*.

[Duquennoy et al. 2017] Duquennoy, S., Elsts, A., Al Nahas, B., and Oikonomo, G. (2017). Tsch and 6tisch for contiki: Challenges, design and evaluation. In *13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE.

[Fawwaz and Chung 2023] Fawwaz, D. Z. and Chung, S.-H. (2023). Adaptive Trickle Timer for Efficient 6TiSCH Network Formation using Q-Learning. *IEEE Access*.

[Ha and Chung 2022] Ha, Y. and Chung, S.-H. (2022). Traffic-Aware 6TiSCH Routing Method for IIoT Wireless Networks. *IEEE Internet of Things Journal*.

[Hamza and Kaddoum 2019] Hamza, T. and Kaddoum, G. (2019). Enhanced minimal scheduling function for IEEE 802.15. 4e TSCH networks. In *WCNC*. IEEE.

[Hauweele et al. 2020] Hauweele, D., Koutsiamanis, R.-A., Quoitin, B., and Papadopoulos, G. Z. (2020). Pushing 6TiSCH minimal scheduling function (MSF) to the limits. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE.

[Hazra et al. 2021] Hazra, A., Adhikari, M., Amgoth, T., and Srirama, S. N. (2021). A comprehensive survey on interoperability for IIoT: Taxonomy, standards, and future directions. *ACM Computing Surveys (CSUR)*, 55(1):1–35.

[Kalita and Khatua 2022] Kalita, A. and Khatua, M. (2022). 6tisch–ipv6 enabled open stack iot network formation: A review. *ACM Transactions on Internet of Things*, 3(3):1–36.

[Kherbache et al. 2023] Kherbache, M., Sobirov, O., Maimour, M., Rondeau, E., and Benyahia, A. (2023). Decentralized TSCH scheduling protocols and heterogeneous traffic: Overview and performance evaluation. *Internet of Things*, page 100696.

[Le et al. 2009] Le, H., John, M., and Pister, K. (2009). Energy-Aware Routing in Wireless Sensor Networks with Adaptive Energy-Slope Control; EE290Q-2 Spring. *IEEE: Piscataway, NJ, USA*.

[Municio et al. 2019] Municio, E., Daneels, G., Vučinić, M., Latré, S., Famaey, J., Tanaka, Y., Brun, K., Muraoka, K., Vilajosana, X., and Watteyne, T. (2019). Simulating 6TiSCH networks. *Transactions on Emerging Telecommunications Technologies*.

[Nguyen-Duy et al. 2019] Nguyen-Duy, H., Ngo-Quynh, T., Kojima, F., Pham-Van, T., Nguyen-Duc, T., and Luongoudon, S. (2019). RL-TSCH: A Reinforcement Learning Algorithm for Radio Scheduling in TSCH 802.15.4e. In *ICTC*, pages 227–231. IEEE.

[Palattella et al. 2012] Palattella, M. R., Accettura, N., Dohler, M., Grieco, L. A., and Boggia, G. (2012). Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. In *PIMRC*, pages 327–332. IEEE.

[Peter et al. 2023] Peter, O., Pradhan, A., and Mbohwa, C. (2023). Industrial internet of things (iiot): opportunities, challenges, and requirements in manufacturing businesses in emerging economies. *Procedia Computer Science*, 217:856–865.

[Pratama and Chung 2022] Pratama, Y. H. and Chung, S. (2022). RL-SF: Reinforcement Learning based Scheduling Function for Distributed TSCH Networks. In *IEEE ICEIEC*, pages 5–8.

[Santos et al. 2019] Santos, B. P., Rettore, P. H., Vieira, L. F. M., and Loureiro, A. A. F. (2019). Dribble: A learn-based timer scheme selector for mobility management in IoT. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*.

[Santos et al. 2016] Santos, B. P., Silva, L. A., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Minicursos SBRC*, 31:16.

[Thubert 2021] Thubert, P. (2021). RFC 9030: An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15. 4 (6TiSCH).

[Ullah et al. 2020] Ullah, Z., Al-Turjman, F., Mostarda, L., and Gagliardi, R. (2020). Applications of artificial intelligence and machine learning in smart cities. *Computer Communications*, 154:313–323.

[Vilajosana et al. 2013] Vilajosana, X., Wang, Q., Chraim, F., Watteyne, T., Chang, T., and Pister, K. S. (2013). A realistic energy consumption model for TSCH networks. *IEEE Sensors Journal*, 14(2):482–489.

[Vilajosana et al. 2019] Vilajosana, X., Watteyne, T., Chang, T., Vučinić, M., Duquennoy, S., and Thubert, P. (2019). Ietf 6tisch: A tutorial. *IEEE Communications Surveys & Tutorials*, 22(1):595–615.

[Wang et al. 2018] Wang, Q., Vilajosana, X., and Watteyne, T. (2018). 6TiSCH Operation Sublayer (6top) Protocol (6P)-RFC8480. *Internet Engineering Task Force RFC series*.

[Wiering and Van Otterlo 2012] Wiering, M. A. and Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729.